Return of Pseudo Ransomware

trellix.com/en-us/about/newsroom/stories/threat-labs/return-of-pseudo-ransomware.html

By Christiaan Beek, Max Kersten and Raj Samani · January 20, 2022

Arnab Roy, Filippo Sitzia and Mo Cashman contributed to the research supporting this blog

Recent news reports of a “ransomware” campaign targeting Ukraine has resulted in significant press coverage regarding not only attribution but also possible motive. Unlike traditional ransomware campaigns where the motive is obvious, this particular campaign is believed to be pseudo in nature[1]. In other words, its intention is likely to cause destruction of infected systems since the wiper at Stage 4 simply overwrites data on the victim’s system, meaning no decryption is possible. Whilst the campaign is targeting largely one country, the Trellix Advanced Threat Research team have published an MVISION Insights campaign to track the threat which highlights what indicators have been found in other countries. (Additional details)

Source: Insights Preview

Subsequently we would strongly recommend organizations ensure that their security posture has the necessary controls to protect and detect against the threat. Indicators of Compromise for active hunting are available within the preview, and we have incorporated all known indicators into our products. Furthermore, work continues to ensure that we can identify any new elements to this attack as they arise.

Campaign details

First samples and indicators were reported on Thursday the 13rd of January in the late evening UTC time, which is around 1AM in the morning and onwards in the Ukraine.

The attack consists of three stages, as can be seen below.

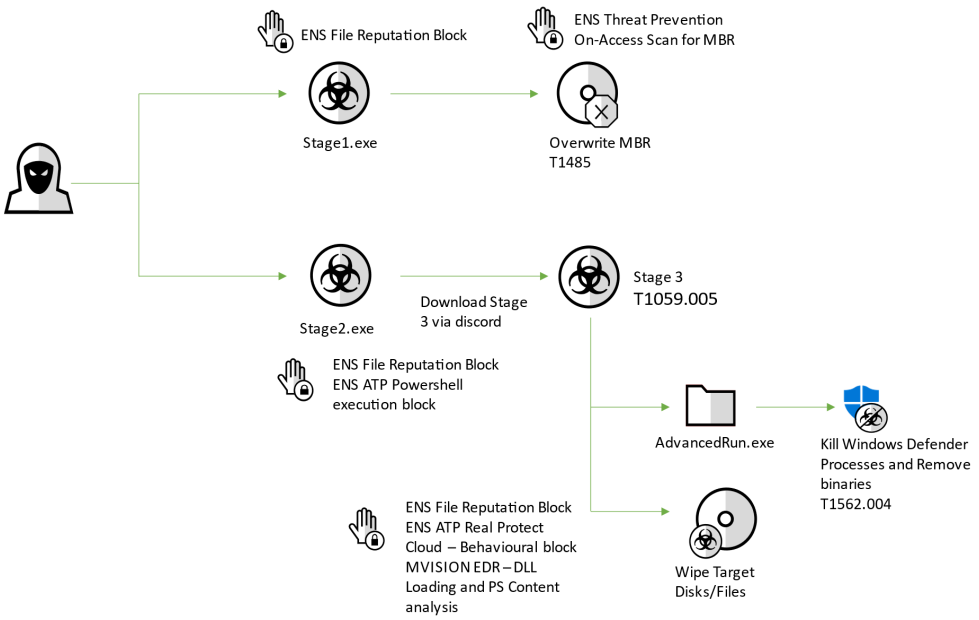


Figure 1: Attack Flow

Stage 1: Overwrites the MBR and destroys all partitions Initial files that were discovered were:

Filename	Stage1.exe
Compile time	01-10-2022 10:37:18
SHA1	189166d382c73c242ba45889d57980548d4ba37e
SHA256	a196c6b8ffcb97ffb276d04f354696e2391311db3841ae16c8c9f56f36a38e92
MD5	5d5c99a08a7d927346ca2dafa7973fc1

Stage 2: Downloads Stage 3 via CDN hosted by discord

Filename	Stage2.exe (originally named Tbopbh.exe)
Compile time	01-10-2022 14:39:54
SHA1	16525cb2fd86dce842107eb1ba6174b23f188537
SHA256	dcbbae5a1c61dbbbb7dcd6dc5dd1eb1169f5329958d38b58c3fd9384081c9b78
MD5	14c8482f302b5e81e3fa1b18a509289d

Stage 3: Executes the file wiper

Filename	Frkmlkdkdubkznbkmcf.dll

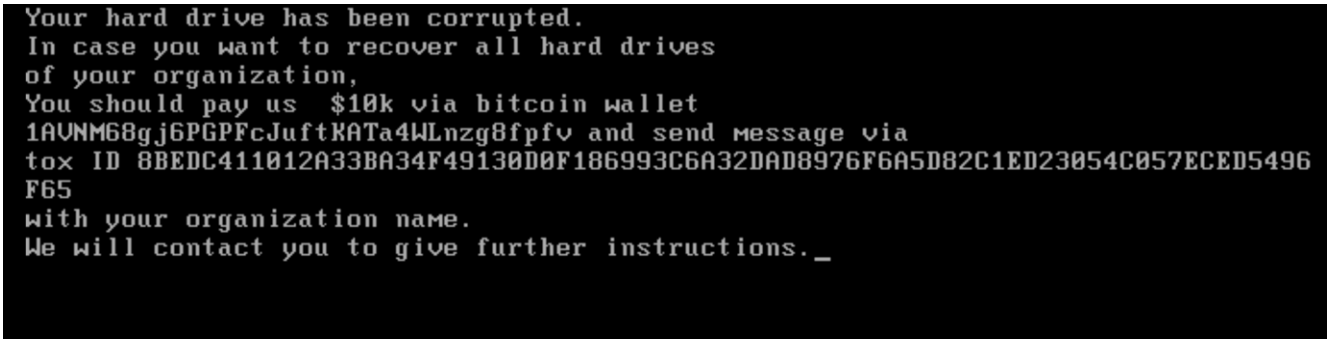
SHA1	82d29b52e35e7938e7ee610c04ea9daaf5e08e90
SHA256	9ef7dbd3da51332a78eff19146d21c82957821e464e8133e9594a07d716d892d
MD5	e61518ae9454a563b8f842286bbdb87b

The file path of both executables was “C:\”. All jokes aside, the files were actually named like this.

Both files have a destructive character as we observed during our analysis of the samples.

Stage 1 – The Master Boot Record rewrite

When starting with the first stage, “Stage1.exe”, it imposes itself as ransomware after execution, where it overwrites the Master Boot Record (MBR). After running the malware in a VM and rebooting the machine, the following note appears:



As with most ransomware notes, the usual language is in there: a notification that the device has been encrypted, and that files are being held hostage, together with a payment address to send the demanded amount to, in the requested currency. In this case, which is rare, a TOX ID is included. TOX is used to chat end-to-end encrypted via peer-to-peer connections.

Although different variants were analyzed, the amount and BTC address stayed the same, which in most Ransomware-as-a-Service operations changes, using bit mixers to obfuscate transactions.

Analysing the code, no code was observed to delete volume shadow copies or block the recovery mode boot process, which is often used to remove malware and/or restore operations. These steps are very common in most ransomware samples.

In this below snippet of code, we observe the malware accessing the physical drive, which is where the MBR resides, after which it will overwrite said MBR.

```
mov     [esp+2040h+dwDesiredAccess], 10000000h ; dwDesiredAccess
mov     [esp+2040h+lpFileName], offset FileName ; "\\\\.\\PhysicalDrive0"
call    CreateFileW
mov     esi, eax
lea     eax, [ebp-2018h]
sub     esp, 1Ch
mov     [esp+2040h+lpFileName], esi ; hFile
mov     [esp+2040h+dwCreationDisposition], 0 ; lpOverlapped
mov     [esp+2040h+lpSecurityAttributes], 0 ; lpNumberOfBytesWritten
mov     [esp+2040h+dwShareMode], 200h ; nNumberOfBytesToWrite
mov     [esp+2040h+dwDesiredAccess], eax ; lpBuffer
call    WriteFile
```

Stage 2 – The Discord Downloader

The ‘Stage2’ file is using an icon that resembles the icon of the proxy client “Proxifier”. The authors tried to evade detection by signing the sample with a certificate and mimicking itself as a Microsoft binary belonging to the operating system. The meta-data in that is attached to the file is Russian, as can be seen in the screenshot below. The assembly description and title both equal “Проводник” (or “Conductor” in English), as can be seen in line 24 and 25 in the screenshot below.

```
13  [assembly: AssemblyVersion("10.0.18362.1500")]
14  [assembly: AssemblyTrademark("")]
15  [assembly: AssemblyFileVersion("10.0.18362.1500")]
16  [assembly: AssemblyCompany("Microsoft Corporation")]
17  [assembly: AssemblyCopyright("© Корпорация Майкрософт. Все права защищены.")]
18  [assembly: Guid("f27071ad-742b-4416-aac2-f9626c7709d2")]
19  [assembly: ComVisible(false)]
20  [assembly: AssemblyConfiguration("")]
21  [assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
22  [assembly: CompilationRelaxations(8)]
23  [assembly: AssemblyProduct("Операционная система Microsoft® Windows®")]
24  [assembly: AssemblyDescription("Проводник")]
25  [assembly: AssemblyTitle("Проводник")]
26  [assembly: TargetFramework(".NETFramework,Version=v4.0", FrameworkDisplayName = ".NET Framework 4")]
```

When executed, stage2 executes an encoded PowerShell command, which is given below.

powershell.exe" -enc UwB0AGEAcgB0AC0AUwBsAGUAZQBwACAALQBzACAAMQAwAA==

Once the base64 encoded command is decoded, the actual command appears: a ten second sleep to delay the execution:

Powershell.exe Start-Sleep -s 10

The malware then moves on to download a file, named “Tbopbh.jpg” from a Discord CDN server.

hxxps://cdn[.]discordapp[.]com/attachments/928503440139771947/930108637681184768/Tbopbh[.]jpg

Filename	Tbobph.jpg
SHA1	b2d863fc444b99c479859ad7f012b840f896172e

SHA256	923eb77b3c9e11d6c56052318c119c1a22d11ab71675e6b95d05eeb73d1accd6
MD5	b3370eb3c5ef6c536195b3bea0120929

This file is not an image, but rather a reversed PE file. Prior to its invocation, the data is reversed in order. The file is a DotNet Framework based DLL, where the starting point of stage 3 is reflectively invoked, as can be seen in the snippet below.

```
flag = Manager.ReflectItem(methodInfo2.Name, "Ylfwdwgmpilzyaph");
 methodInfo2.Invoke(null, null);
```

The static function in the third stage is named “Ylfwdwgmpilzyaph” and resides in “ClassLibrary1.Main” as a namespace and class respectively. The function takes no arguments, nor does it return any value.

Stage 3 – A re-used loader

Filename	Frkmlkdkdubkznbkmcf.dll
SHA1	82d29b52e35e7938e7ee610c04ea9daaf5e08e90
SHA256	9ef7dbd3da51332a78eff19146d21c82957821e464e8133e9594a07d716d892d
MD5	e61518ae9454a563b8f842286bbdb87b

This stage is also a DotNet Framework binary, as can be derived from the reflective invocation in the previous step. The used loader is more common, and is generally used to spread commodity malware. As described in this recent Twitter thread, the xClient RAT was also distributed via this loader family.

This loader starts several processes during its execution, after which it uses process hollowing to inject and run stage 4. At first, the loader ensures it has administrative privileges. If not, the process is started again while requesting said privileges.

The first process it starts, is a file it drops to “%temp%\Nmddfrqqrbyjeygggda.vbs”. The content of this file is rather small, as can be seen below.

```
CreateObject("WScript.Shell").Run "powershell Set-MpPreference -ExclusionPath 'C:\'", 0, False
```

This excludes “C:\” from Windows Defender’s prying eyes. As stated above, the first two stages were both located in this exact location, meaning their stay will not be detected by Defender’s scans once the exclusion is in-place.

The two processes that are started after that, utilise the same tool to execute commands: AdvancedRun by Nir Sofer. The used version (1.2.2.6, signed on Monday the third of August 2020, at 5:45:51 AM by Nir Sofer), is a legitimate executable that is abused by this malware.

The first execution of AdvancedRun is used to stop Defender, using an invisible window.

```
%temp%\AdvancedRun.exe /EXEFilename "C:\Windows\System32\sc.exe" /WindowState 0
/CommandLine "stop WinDefend" /StartDirectory "" /RunAs 8 /Run
```

The second execution of AdvancedRun is used to completely remove the files of Defender from the system.

```
%temp%\AdvancedRun.exe /EXEFilename
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" /WindowState 0 /CommandLine
"rmdir 'C:\ProgramData\Microsoft\Windows Defender' -Recurse" /StartDirectory "" /RunAs 8 /Run
```

The loader then uses the interoperability functionality within the DotNet Framework to execute unmanaged code from a managed context. The list of functions below shows which unmanaged functions were used, in no apparent order. The names of these files are base64 encoded in the loader.

- ResumeThread
- Wow64SetThreadContext
- SetThreadContext
- GetThreadContext
- VirtualAllocEx
- WriteProcessMemory
- ZwUnmapViewOfSection
- CreateProcessA
- CloseHandle
- ReadProcessMemory

These function calls indicate the usage of process injection. Based on this, the fourth stage can be dumped, of which the analysis can be found below.

Stage 4 – The wiper

Filename	-
SHA1	8be3c66aecd425f1f123aadc95830de49d1851b5
SHA256	191ca4833351e2e82cb080a42c4848cfbc4b1f3e97250f2700eff4e97cf72019
MD5	343fcded2aaf874342c557d3d5e5870d

The wiper is written in C, and does not contain symbols nor other debug information. The main function is given below, together with some notes.

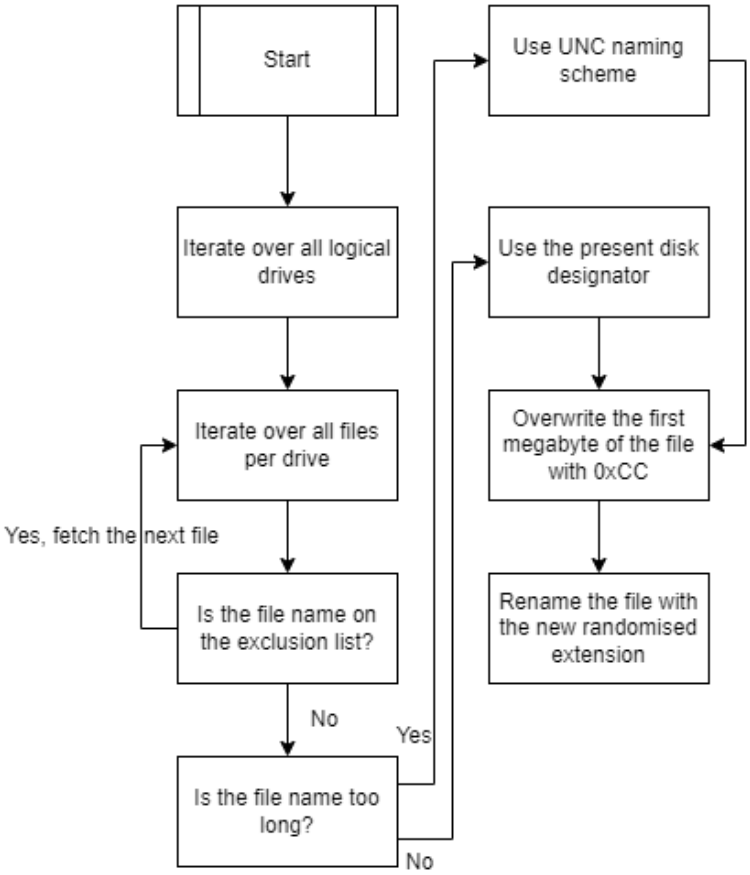
```
int main(void)

{
                    /* Iterates over all drives, starting at A:, where all files are then
                       recursively wiped */
  iterate_drives_and_wipe_files();
                    /* Runs "ping" to delay the delete command of its own location, which is not
                       running anymore due to the ExitWindowsEx call below. */
  sleep_and_delete_self();
                    /* Per MSDN: "Shuts down the system to a point at which it is safe to turn off
                       the power. All file buffers have been flushed to disk, and all running
                       processes have stopped." */
  ExitWindowsEx(EWX_SHUTDOWN,SHTDN_REASON_MINOR_NETWORK_CONNECTIVITY);
  return 0;
}
```

The malware iterates over all drives, where it wipes all files that match any of the following 191 extensions.

.HTML, .HTM, .SHTML, .XHTML, .PHTML, .PHP, .JSP, .ASP, .PHPS, .PHP5, .ASPX, .PHP4, .PHP6, .PHP7, .PHP3, .DOC, .DOCX, .XLS, .XLSX, .PPT, PPTX, .PST, .OST, .MSG, .EML, .VSD, .VSDX, .TXT, .CSV, .RTF, .WKS, .WK1, .PDF, .DWG, .ONETOC2, .SNT, .JPEG, .JPG, .DOCB, .DOCM, .DOT, .DOTM, .DOTX, .XLSM, .XLSB, .XLW, .XLT, .XLM, .XLC, .XLTX, .XLTM, .PPTM, .POT, .PPS, .PPSM, .PPSX, .PPAM, .POTM, .EDB, .HWP, .602, .SXI, .STI, .SLDX, .SLDM, .BMP, .PNG, .GIF, .RAW, .CGM, .SLN, .TIF, .TIFF, .NEF, .PSD, .AI, .SVG, .DJVU, .SH, .CLASS, .JAR, .BRD, .SCH, .DCH, .DIP, .PL, .VB, .VBS, .PS1, .BAT, .CMD, .JS, .ASM, .H, .PAS, .CPP, .C, .CS, .SUO, .ASC, .LAY6, .LAY, .MML, .SXM, .OTG, .ODG, .UOP, .STD, .SXD, .OTP, .ODP, .WB2, .SLK, .DIF, .STC, .SXC, .OTS, .ODS, .3DM, .MAX, .3DS, .UOT, .STW, .SXW, .OTT, .ODT, .PEM, .P12, .CSR, .CRT, .KEY, .PFX, .DER, .OGG, .RB, .GO, .JAVA, .INC, .WAR, .PY, .KDBX, .INI, .YML, .PPK, .LOG, .VDI, .VMDK, .VHD, .HDD, .NVRAM, .VMSD, .VMSN, .VMSS, .VMTM, .VMX, .VMXF, .VSWP, .VMTX, .VMEM, .MDF, .IBD, .MYI, .MYD, .FRM, .SAV, .ODB, .DBF, .DB, .MDB, .ACCDB, .SQL, .SQLITEDB, .SQLITE3, .LDF, .SQ3, .ARC, .PAQ, .BZ2, .TBK, .BAK, .TAR, .TGZ, .GZ, .7Z, .RAR, .ZIP, .BACKUP, .ISO, .VCD, .BZ, .CONFIG

The wiping process is shown in the diagram below.



The name exclusion list contains the following file names: ".", "..", "$RECYCLE.BIN", or the value of the environment variable named "HOMEDRIVE".

Files names which are too long cannot be handled properly using the disk designator, which is why a different approach must be taken there. Microsoft's documentation describes all scenarios clearly, including the how and why.

Once the wiping has finished, the second function within the main function is called, which ensures the asynchronous execution of the given command and the termination of its own process will lead to the deletion of the wiper's file, which is not in use anymore at that point in time. The code is given below.

```
void sleep_and_delete_self(void)

{
  CHAR self [260];
  char command [524];

  GetModuleFileNameA((HMODULE)0x0,self,0x104);
  sprintf(command,"cmd.exe /min /C ping 111.111.111.111 -n 5 -w 10 > Nul & Del /f /q \"%s\"",self);
  run_given_command_line(command);
  return;
}
```

The module name is copied into the command, which runs a minimalised window where a ping request is issued 5 times to "111.111.111.111", with a 10 millisecond wait time between each request. The output of the command is sent to "Nul", ensuring the output is hidden. Additionally, the path to the

wiper is used as an argument to be deleted forcefully and quietly from the disk. This would fail if the program was still running, which is why the ping command is executed first. The function in the screenshot below executes the given command, and closes all handles.

```
void __cdecl run_given_command_line(LPSTR command)

{
  int i;
  _STARTUPINFOA *ptr_startup_info;
  _PROCESS_INFORMATION *ptr_process_info;
  _PROCESS_INFORMATION process_info;
  _STARTUPINFOA startup_info;

  ptr_startup_info = &startup_info;
  for (i = 0x11; i != 0; i = i + -1) {
    ptr_startup_info->cb = 0;
    ptr_startup_info = (_STARTUPINFOA *)&ptr_startup_info->lpReserved;
  }
  ptr_process_info = &process_info;
  for (i = 4; i != 0; i = i + -1) {
    ptr_process_info->hProcess = (HANDLE)0x0;
    ptr_process_info = (_PROCESS_INFORMATION *)&ptr_process_info->hThread;
  }
  CreateProcessA((LPCSTR)0x0,command,(LPSECURITY_ATTRIBUTES)0x0,(LPSECURITY_ATTRIBUTES)0x0,0,
                 0x8000000,(LPVOID)0x0,(LPCSTR)0x0,&startup_info,&process_info);
  CloseHandle(process_info.hThread);
  CloseHandle(process_info.hProcess);
  return;
}
```

At last, ExitWindowsEx is called, using EWX_SHUTDOWN as the flag, which ensures that all file buffers have been flushed to the disk. The given shutdown reason is network connectivity related.

MITRE Techniques

T1059.001	PowerShell	Usage of PowerShell to download file from Discord and execute.
T1485	Data Destruction	MBR wiping and/or file overwriting to corrupt them
T1059.005	Visual Basic	Stage3 is running vbs with Wscript init.
T1562.004	Disable or Modify System firewall	Stage3 is disabling the Defender
T1112	Modify Registry	Registry settings changed to disable tooling
T1105	Ingress Tool Transfer	Files like Stage1.exe and stage2.exe were transferred into and through the network.

Defensive Guidance

Threat Intelligence

MVISION Insights provides early visibility into the IOC's related to this campaign and the associated detections if any in your environment

MVISION Insights also provides signs of prevalence within your environment by matching the ENS detections with the campaign and also providing the process trace information to show the flow of execution

MVISION Insights – stage01/02.exe Process Trace

Ensure your ENS AMCore definitions are upto date and GTI and real protect is enabled

Threat Event Log

View threat and security events from your managed systems.

Hide Filter

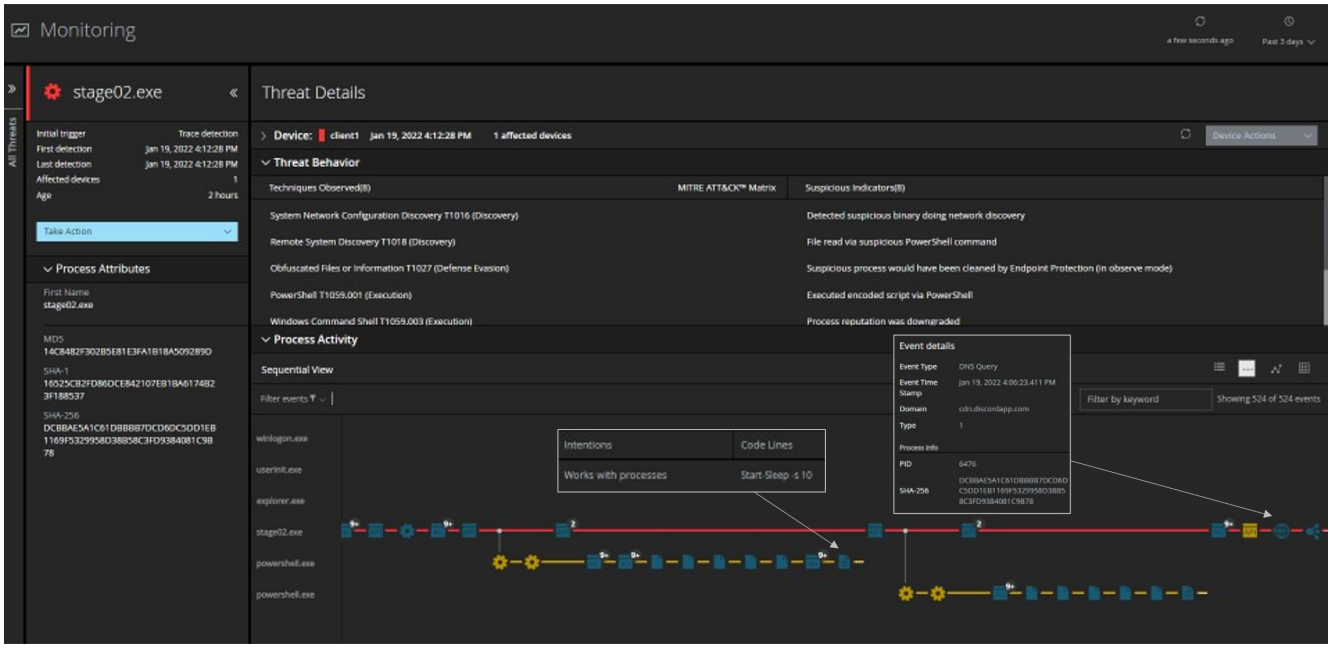
Preset: Last day | Custom: None | Quick find: | Apply | Clear | Show selected rows

Event Received Time	Target Hash	Target Path	Event Description	Event Category	Threat Source Process Nam	Threat Type	Action Taken	Analyzer Detection Method	Event ID	Threat Name
1/19/22 3:07:06 PM UTC	65d86c34814c02569e2ad53fd24e7f61	C:\Windows\SysWOW64\WindowsPowerShell\v1.0	Adaptive Threat Protection Would Clean	Reputation	stage02.exe	Trojan	Adaptive Threat Protection Would Clean	On-Execute Scan	35106	ATP/Suspect!21d5224e20a4
1/19/22 3:07:06 PM UTC	14c8482f302b5e81e3fa1b18a509289d	C:\Users\john\Downloads\stage2	Adaptive Threat Protection Would Clean	Malware detected using heuristics	explorer.exe	Trojan	Adaptive Threat Protection Would Clean	Real Protect Cloud	35106	Real Protect-PENG!14C8482F30
1/19/22 3:07:06 PM UTC		C:\Windows\System32	Adaptive Threat Protection Would Clean	Reputation	powershell.exe	Trojan	Adaptive Threat Protection Would Clean	On-Execute Scan	35106	ATP/Suspect!00667a0f0c0d
1/19/22 3:07:06 PM UTC	65d86c34814c02569e2ad53fd24e7f61	C:\Windows\SysWOW64\WindowsPowerShell\v1.0	Adaptive Threat Protection Would Clean	Reputation	stage02.exe	Trojan	Adaptive Threat Protection Would Clean	On-Execute Scan	35106	ATP/Suspect!21d5224e20a4
1/19/22 3:07:06 PM UTC	14c8482f302b5e81e3fa1b18a509289d	C:\Users\john\Downloads\stage2	Adaptive Threat Protection Would Clean	Reputation	explorer.exe	Trojan	Adaptive Threat Protection Would Clean	On-Execute Scan	35106	ATP/Suspect!16525cb2fd86
1/19/22 3:07:06 PM UTC	5d5c99a08a7d927346ca2dafa7973fc1	C:\Users\john\Downloads\stage1	Adaptive Threat Protection Would Clean	Malware detected using heuristics	explorer.exe	Trojan	Adaptive Threat Protection Would Clean	Real Protect Cloud	35106	Real Protect-PEE!5D5C99A08A7D
1/19/22 3:07:06 PM UTC		C:\Windows\System32	Adaptive Threat Protection Would Clean	Reputation	powershell.exe	Trojan	Adaptive Threat Protection Would Clean	On-Execute Scan	35106	ATP/Suspect!00667a0f0c0d
1/19/22 2:59:03 PM UTC	5d5c99a08a7d927346ca2dafa7973fc1	C:\Users\john\Downloads\stage1	Adaptive Threat Protection Would Clean	Reputation	explorer.exe	Trojan	Adaptive Threat Protection Would Clean	On-Execute Scan	35106	ATP/Suspect!189166d382c7
1/19/22 2:53:33 PM UTC	14c8482f302b5e81e3fa1b18a509289d	C:\Users\john\Downloads\stage2	Infected file deleted.	Malware detected	C:\Windows\explorer.exe	Trojan	Delete	On-Access Scan	1027	RDN/Generic Downloader.x
1/19/22 2:53:33 PM UTC	5d5c99a08a7d927346ca2dafa7973fc1	C:\Users\john\Downloads\stage1	Infected file deleted.	Malware detected	C:\Windows\explorer.exe	Trojan	Delete	On-Access Scan	1027	RDN/Generic.dx

Figure 2: Following are the triggered ENS detections for stage1 and 2

MVISION EDR provides complete visibility into the execution of the processes as follows:

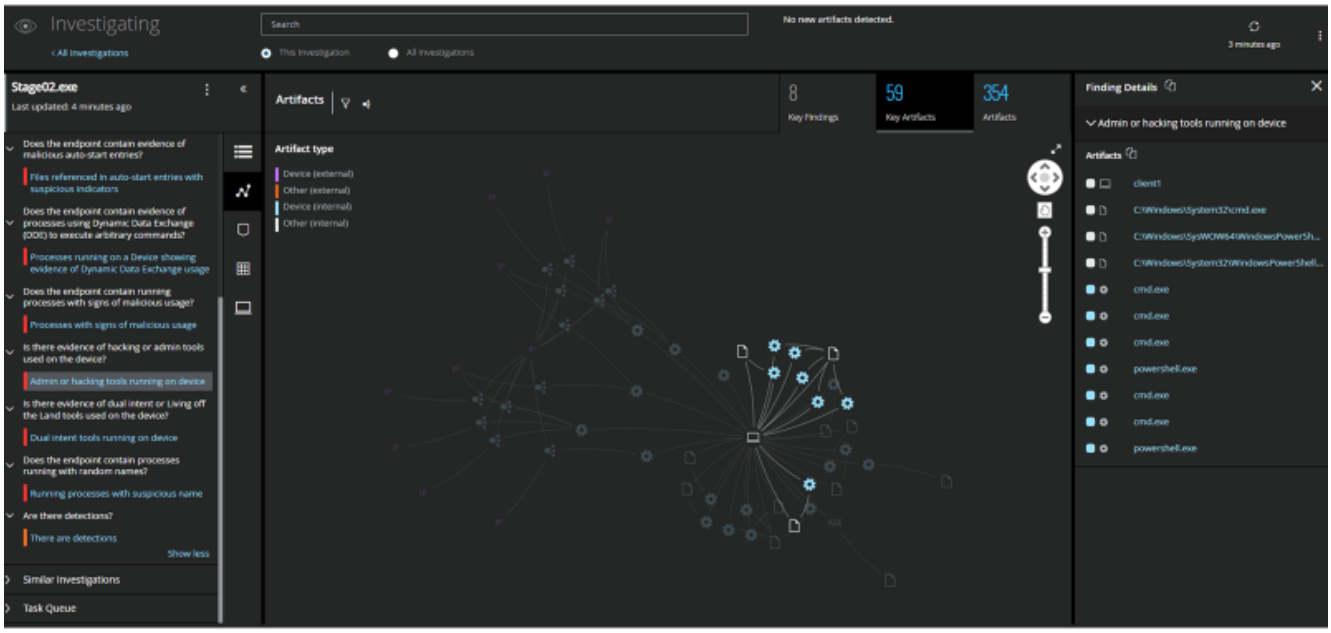
MVISION EDR



Here we can see the communication flow for the second stage payloads and the intent of the powershell script executions

Creating an investigation from the threat detections we can see the use of dual intel tools such as powershell, cmd to execute os commands and make changes to the system and download additional payloads.

MVISION EDR



[1] https://www.helpnetsecurity.com/2017/08/14/pseudo-ransomware/