Underground Ransomware deployed by Storm-0978 that exploited CVE-2023-36884

Prepared by: Vlad Pasca, Senior Malware & Threat Analyst

SecurityScorecard

SecurityScorecard.com
info@securityscorecard.com

1140 6th Avenue
Floor 19
New York, NY 10036
1.800.682.1707

Table of contents

Executive summary

The Underground ransomware is the successor of the Industrial Spy ransomware and was deployed by a threat actor called Storm-0978. The malware stops a target service, deletes the Volume Shadow Copies, and clears all Windows event logs.

The files are encrypted using the 3DES algorithm, with the key and IV being encrypted using an RSA public key. The ransomware deletes itself after the file encryption is complete. The extension of the encrypted files isn't changed, but four specific bytes are added at the end of them.

Analysis and findings

SHA256: d4a847fa9c4c7130a852a2e197b205493170a8b44426d9ec481fc4b285a92666

The malware retrieves the command line arguments using the GetCommandLineW and CommandLineToArgvW functions:

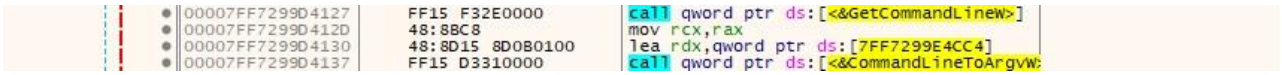


Figure 1

The ransomware can run with a single parameter, which represents a single directory to be encrypted.

The GetWindowsDirectoryW API is used to obtain the path of the Windows directory (see Figure 2).

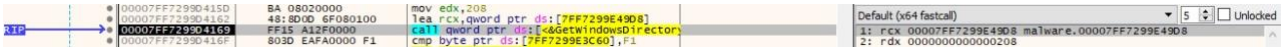


Figure 2

The malware implements the obfuscation technique called "stack strings", which splits plaintext strings and constructs them at runtime:

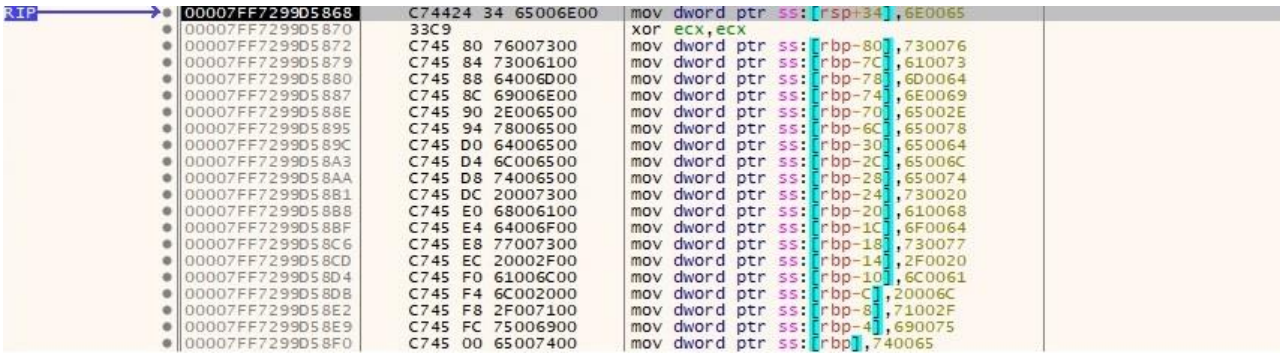


Figure 3

The malicious process deletes all Volume Shadow Copies using vssadmin.exe tool:

Figure 4

The time limit for disconnected RDP sessions is modified using the ShellExecuteW method, as highlighted in Figure 5.

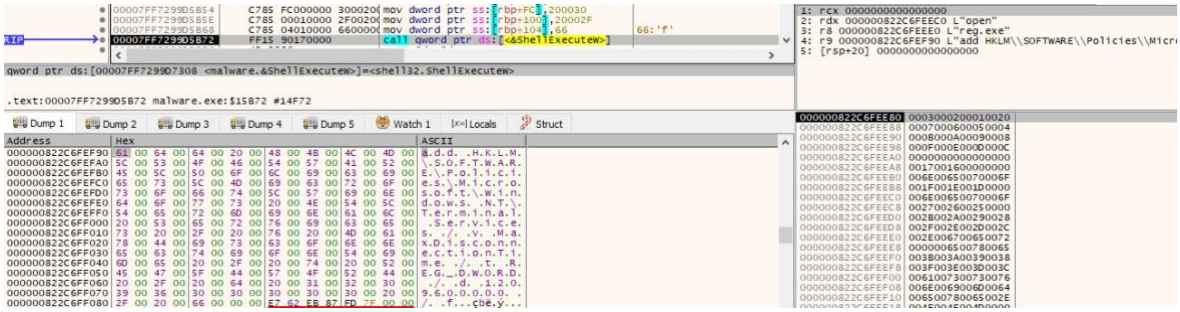


Figure 5

Finally, the binary stops the MSSQLSERVER service:

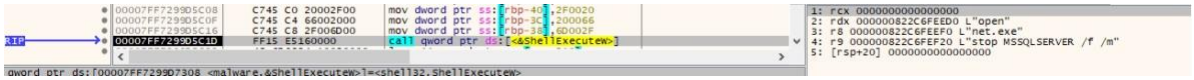


Figure 6

The process creates a new thread that executes the StartAddress function:

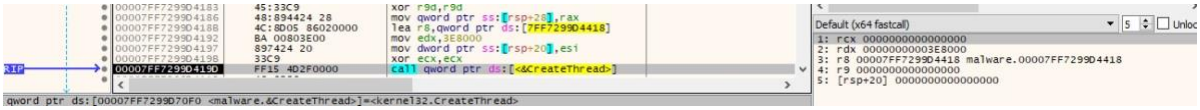


Figure 7

Thread activity – StartAddress function

The ransomware embedded an RSA public key that will be used to encrypt the 3DES key and IV:

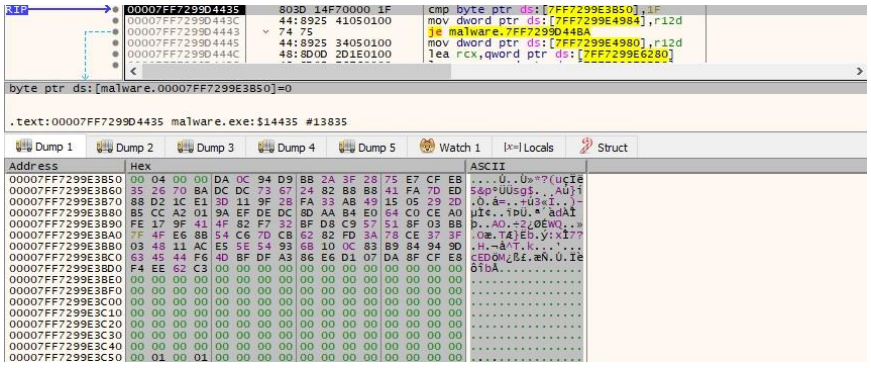
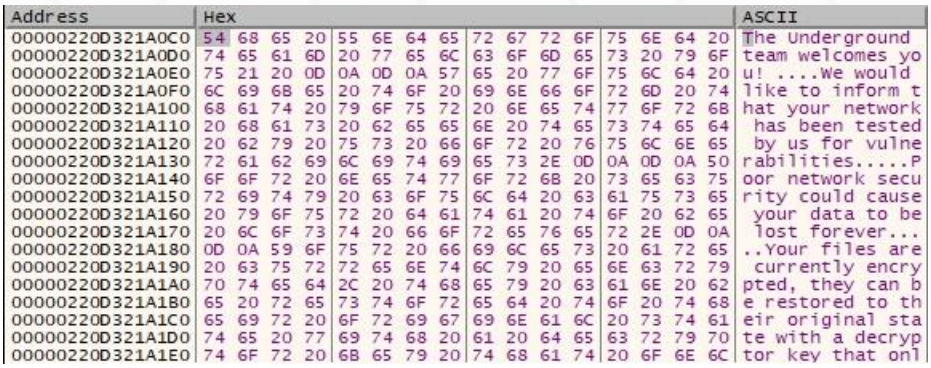


Figure 8

The ransom note content can be also identified in the malware's body:



Address	Hex	ASCII
00000220D321A0C0	54 68 65 20 55 6E 64 65 72 67 72 6F 75 6E 64 20	The Underground
00000220D321A0D0	74 65 61 6D 20 77 65 6C 63 6F 6D 65 73 20 79 6F	team welcomes yo
00000220D321A0E0	75 21 20 0D 0A 0D 0A 57 65 20 77 6F 75 6C 64 20	u! ....We would
00000220D321A0F0	6C 69 6B 65 20 74 6F 20 69 6E 66 6F 72 6D 20 74	like to inform t
00000220D321A100	68 61 74 20 79 6F 75 72 20 6E 65 74 77 6F 72 6B	hat your network
00000220D321A110	20 68 61 73 20 62 65 65 6E 20 74 65 73 74 65 64	has been tested
00000220D321A120	20 62 79 20 75 73 20 66 6F 72 20 76 75 6C 6E 65	by us for vulne
00000220D321A130	72 61 62 69 6C 69 74 69 65 73 2E 0D 0A 0D 0A 50	rabilities.....P
00000220D321A140	6F 6F 72 20 6E 65 74 77 6F 72 6B 20 73 65 63 75	oor network secu
00000220D321A150	72 69 74 79 20 63 6F 75 6C 64 20 63 61 75 73 65	rity could cause
00000220D321A160	20 79 6F 75 72 20 64 61 74 61 20 74 6F 20 62 65	your data to be
00000220D321A170	20 6C 6F 73 74 20 66 6F 72 65 76 65 72 2E 0D 0A	lost forever...
00000220D321A180	0D 0A 59 6F 75 72 20 66 69 6C 65 73 20 61 72 65	..Your files are
00000220D321A190	20 63 75 72 72 65 6E 74 6C 79 20 65 6E 63 72 79	currently encry
00000220D321A1A0	70 74 65 64 2C 20 74 68 65 79 20 63 61 6E 20 62	pted, they can b
00000220D321A1B0	65 20 72 65 73 74 6F 72 65 64 20 74 6F 20 74 68	e restored to th
00000220D321A1C0	65 69 72 20 6F 72 69 67 69 6E 61 6C 20 73 74 61	eir original sta
00000220D321A1D0	74 65 20 77 69 74 68 20 61 20 64 65 63 72 79 70	te with a decryp
00000220D321A1E0	74 6F 72 20 6B 65 79 20 74 68 61 74 20 6F 6E 6C	tor key that onl

Figure 9

The executable computes the MD5 hash of the RSA modulus, which represents the victim's ID:

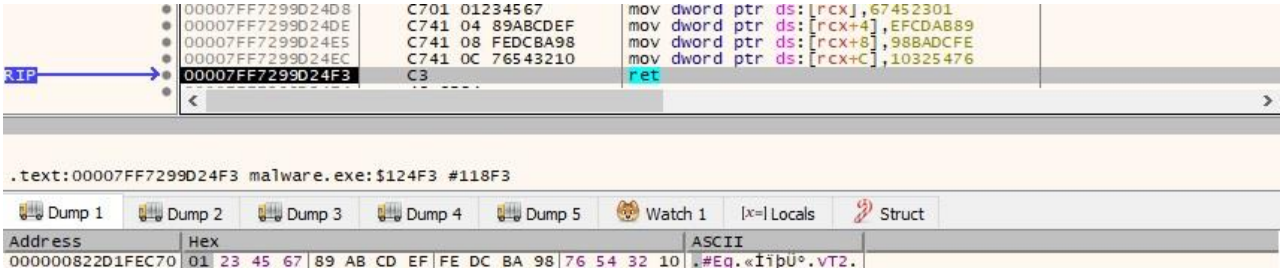


Figure 10

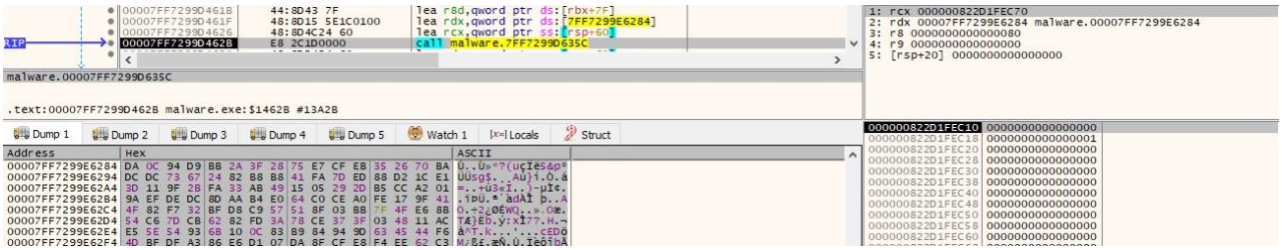
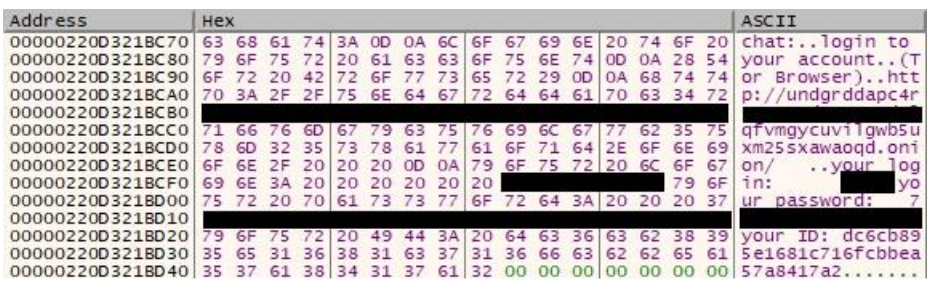


Figure 11

The ransom note containing a TOR link and credentials is modified to include the above victim's ID:



Address	Hex	ASCII
00000220D321BC70	63 68 61 74 3A 0D 0A 6C 6F 67 69 6E 20 74 6F 20	chat:..login to
00000220D321BC80	79 6F 75 72 20 61 63 63 6F 75 6E 74 0D 0A 28 54	your account..(T
00000220D321BC90	6F 72 20 42 72 6F 77 73 65 72 29 0D 0A 68 74 74	or Browser)..htt
00000220D321BCA0	70 3A 2F 2F 75 6E 64 67 72 64 64 61 70 63 34 72	p://undgrddapc4r
00000220D321BCB0	[redacted]	[redacted]
00000220D321BCC0	71 66 76 6D 67 79 63 75 76 69 6C 67 77 62 35 75	qfvmgycuvilgwb5u
00000220D321BCD0	78 6D 32 35 73 78 61 77 61 6F 71 64 2E 6F 6E 69	xm25sxawaoqd.oni
00000220D321BCE0	6F 6E 2F 20 20 20 0D 0A 79 6F 75 72 20 6C 6F 67	on/ ..your log
00000220D321BCF0	69 6E 3A 20 20 20 20 20 [redacted] 79 6F	in: yo
00000220D321BD00	75 72 20 70 61 73 73 77 6F 72 64 3A 20 20 20 37	ur password: 7
00000220D321BD10	[redacted]	[redacted]
00000220D321BD20	79 6F 75 72 20 49 44 3A 20 64 63 36 63 62 38 39	your ID: dc6cb89
00000220D321BD30	35 65 31 36 38 31 63 37 31 36 66 63 62 62 65 61	5e1681c716fcbbea
00000220D321BD40	35 37 61 38 34 31 37 61 32 00 00 00 00 00 00 00	57a8417a2.....

Figure 12

The FindFirstVolumeW API is utilized to begin scanning the volumes on the machine:

Figure 13

The malicious process obtains a list of drive letters for the specified volume via a function call to GetVolumePathNamesForVolumeNameW:

Figure 14

GetVolumeInformationW is used to extract information about the file system and volume associated with the drives:

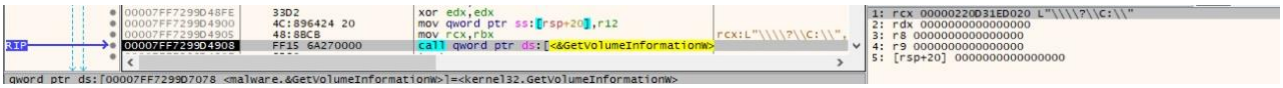


Figure 15

The ransomware continues the volume search by calling the FindNextVolumeW method:

Figure 16

For each of the drives to be encrypted, the binary creates a thread that handles the files encryption (Figure 17).

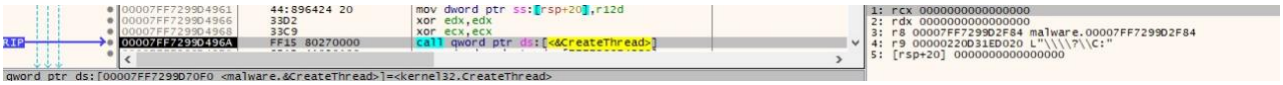


Figure 17

Thread activity – Files encryption

The process enumerates the files found in a directory using the FindFirstFileW and FindNextFileW functions:

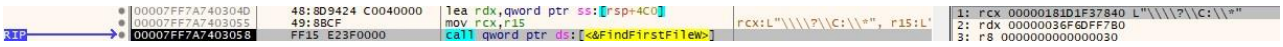


Figure 18

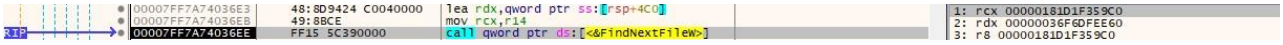


Figure 19

The malware doesn't encrypt a file called "VIPinfo.txt" and the folders containing the following strings:

- Windows
- Microsoft
- google\chrome
- mozilla\firefox
- \opera\

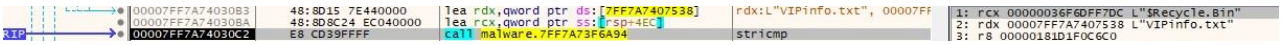


Figure 20

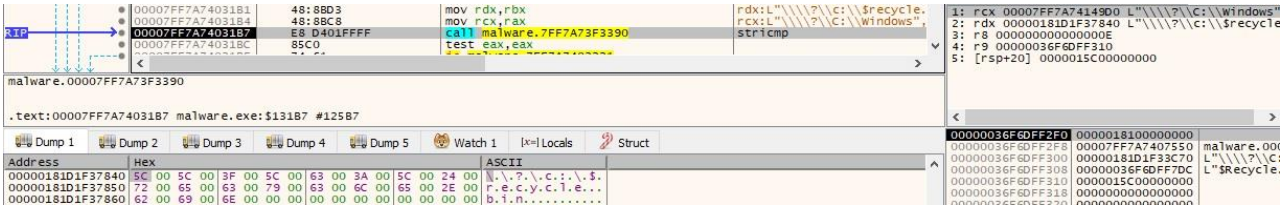


Figure 21

The file's extension is obtained via a function call to PathFindExtensionW (see Figure 22).

Figure 22

The following file's extensions will be skipped by the ransomware:

Address	Hex	ASCII
00000036F6DFE0C0	2E 00 73 00 79 00 73 00 2E 00 65 00 78 00 65 00	..s.y.s...e.x.e.
00000036F6DFE0D0	2E 00 64 00 6C 00 6C 00 2E 00 62 00 61 00 74 00	..d.l.l...b.a.t.
00000036F6DFE0E0	2E 00 62 00 69 00 6E 00 2E 00 63 00 6D 00 64 00	..b.i.n...c.m.d.
00000036F6DFE0F0	2E 00 63 00 6F 00 6D 00 2E 00 63 00 70 00 6C 00	..c.o.m...c.p.l.
00000036F6DFE100	2E 00 67 00 61 00 64 00 67 00 65 00 74 00 2E 00	..g.a.d.g.e.t...
00000036F6DFE110	69 00 6E 00 66 00 31 00 2E 00 69 00 6E 00 73 00	i.n.f.1...i.n.s.
00000036F6DFE120	2E 00 69 00 6E 00 78 00 2E 00 69 00 73 00 75 00	..i.n.x...i.s.u.
00000036F6DFE130	2E 00 6A 00 6F 00 62 00 2E 00 6A 00 73 00 65 00	..j.o.b...j.s.e.
00000036F6DFE140	2E 00 6C 00 6E 00 6B 00 2E 00 6D 00 73 00 63 00	..l.n.k...m.s.c.
00000036F6DFE150	2E 00 6D 00 73 00 69 00 2E 00 6D 00 73 00 74 00	..m.s.i...m.s.t.
00000036F6DFE160	2E 00 70 00 61 00 66 00 2E 00 70 00 69 00 66 00	..p.a.f...p.i.f.
00000036F6DFE170	2E 00 70 00 73 00 31 00 2E 00 72 00 65 00 67 00	..p.s.1...r.e.g.
00000036F6DFE180	2E 00 72 00 67 00 73 00 2E 00 73 00 63 00 72 00	..r.g.s...s.c.r.
00000036F6DFE190	2E 00 73 00 63 00 74 00 2E 00 73 00 68 00 62 00	..s.c.t...s.h.b.
00000036F6DFE1A0	2E 00 73 00 68 00 73 00 2E 00 75 00 33 00 70 00	..s.h.s...u.3.p.
00000036F6DFE1B0	2E 00 76 00 62 00 2E 00 76 00 62 00 65 00 2E 00	..v.b...v.b.e...
00000036F6DFE1C0	76 00 62 00 73 00 2E 00 76 00 62 00 73 00 63 00	v.b.s...v.b.s.c.
00000036F6DFE1D0	72 00 69 00 70 00 74 00 2E 00 77 00 73 00 2E 00	r.i.p.t...w.s...
00000036F6DFE1E0	77 00 73 00 68 00 2E 00 77 00 73 00 66 00 00 00	w.s.h...w.s.f...

Figure 23

The malware also skips the ransom note called "!!readme!!!.txt":

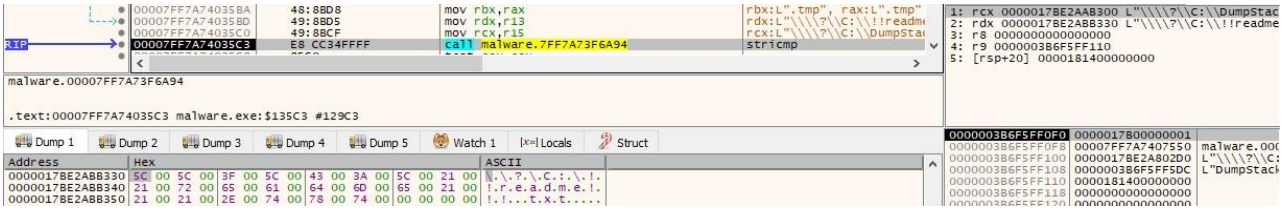


Figure 24

A file is opened using the CreateFileW API (0xc0000000 = **GENERIC_READ** | **GENERIC_WRITE**, 0x1 = **FILE_SHARE_READ**, 0x3 = **OPEN_EXISTING**):

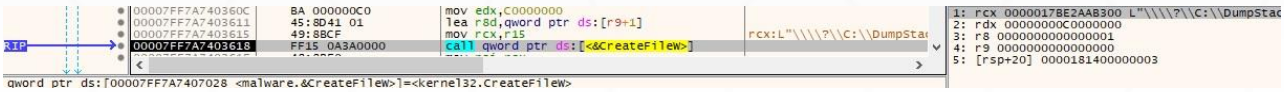


Figure 25

If any of the target files are opened by another process, the binary uses the Restart Manager APIs to kill that process.

The malware starts a new Restart Manager session using the RmStartSession function:

Figure 26

The ransomware determines the process that locked the target file using the RmRegisterResources and RmGetList methods:

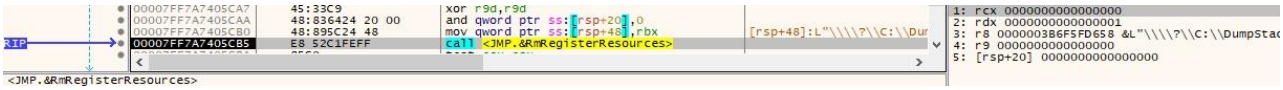


Figure 27

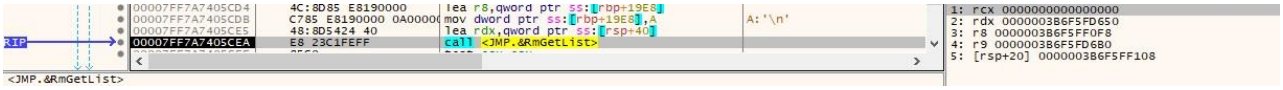


Figure 28

The executable kills the above process using the OpenProcess and TerminateProcess APIs, as shown below:

Figure 29

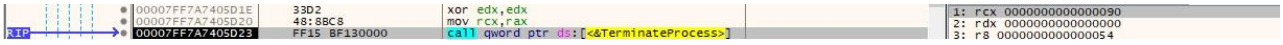


Figure 30

GetSystemTimeAsFileTime is utilized to obtain the system date and time (Figure 31).

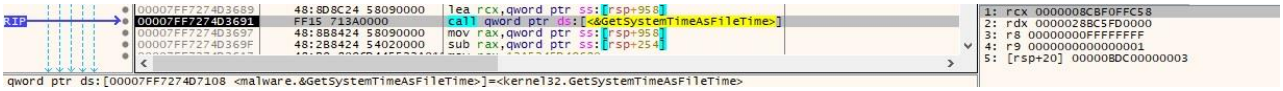


Figure 31

The malicious process retrieves the size of the target file:

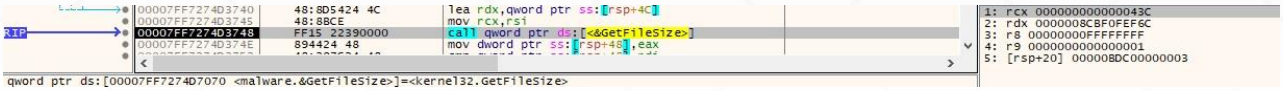


Figure 32

The SetFilePointer API is used to move the file pointer within the target file:

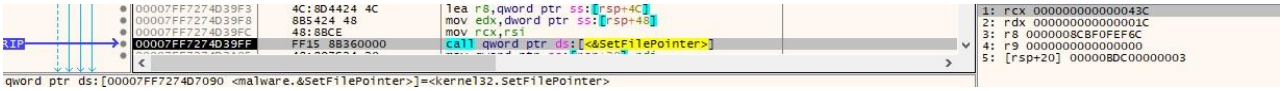


Figure 33

The last four bytes are extracted and compared with “1AY&”, which corresponds to a file that was previously encrypted by Underground ransomware:

```
.text:00007FF7274D3A05 mov     qword ptr [rsp+948h+dwCreationDisposition], rdi ; lpOverlapped
.text:00007FF7274D3A0A lea     r9, [rsp+948h+NumberOfBytesWritten] ; lpNumberOfBytesRead
.text:00007FF7274D3A12 mov     r8d, 4          ; nNumberOfBytesToRead
.text:00007FF7274D3A18 mov     rdx, r14        ; lpBuffer
.text:00007FF7274D3A1B mov     rcx, rsi        ; hFile
.text:00007FF7274D3A1E call    cs:ReadFile
.text:00007FF7274D3A24 add     qword ptr [rsp+948h+FileSizeHigh], 4
.text:00007FF7274D3A2A cmp     dword ptr [r14], '1AY&'
.text:00007FF7274D3A31 jz      loc_7FF7274D3E42
```

Figure 34

The malware creates a ransom note called “!!readme!!!.txt” in every traversed directory:

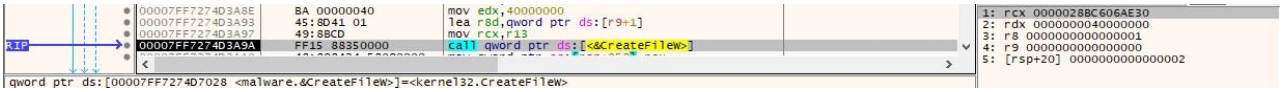


Figure 35

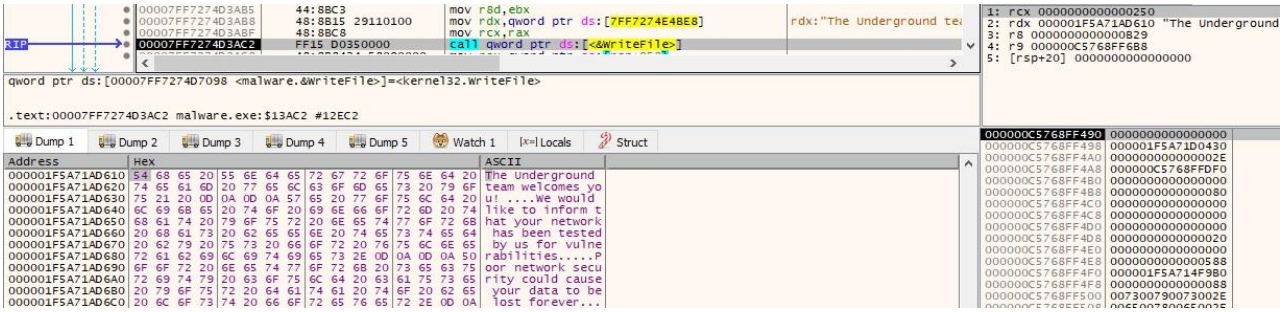


Figure 36

The binary extracts the processor time stamp using the rdtsc instruction and generates 24 pseudo-random bytes twice. The second 24 bytes represent the 3DES key and the first eight bytes from the first iteration represent the IV that will be used for file encryption:

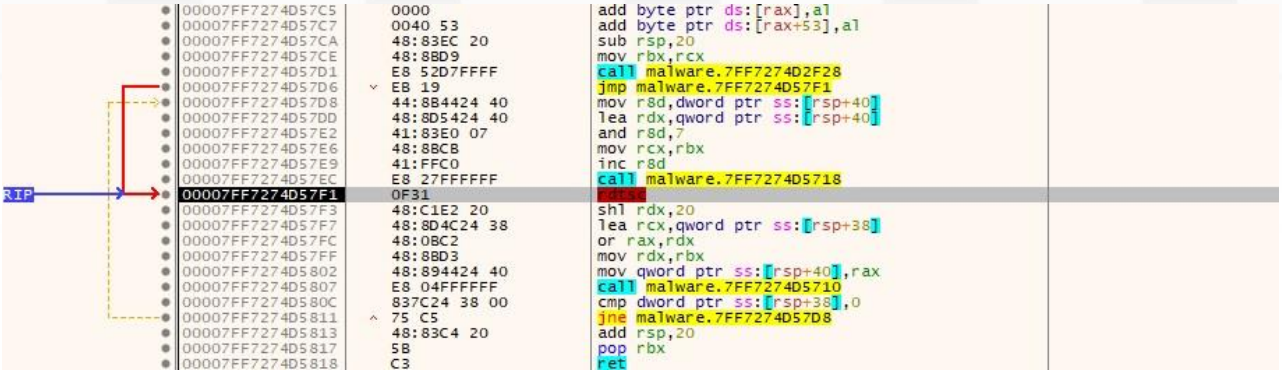


Figure 37

Figure 38

The 3DES key and IV are encrypted using the hard-coded RSA public key, as displayed below:

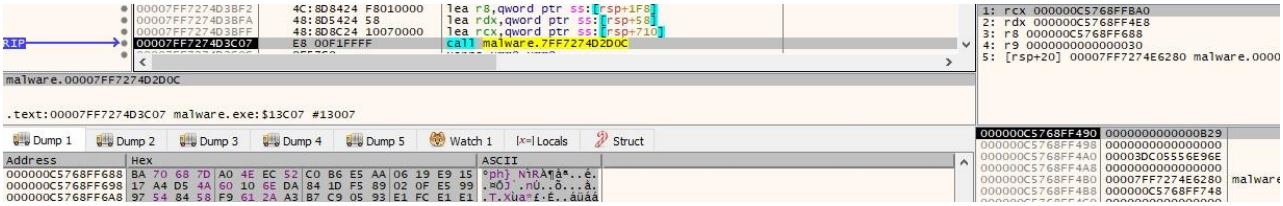
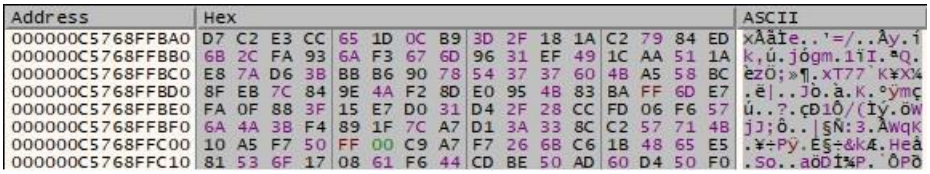


Figure 39



Address	Hex	ASCII
000000C5768FFBA0	D7 C2 E3 CC 65 1D 0C B9 3D 2F 18 1A C2 79 84 ED	xÂãÌe..'=/..Ây.í
000000C5768FFBB0	6B 2C FA 93 6A F3 67 6D 96 31 EF 49 1C AA 51 1A	k,ú.jógm.1ïI.ªQ.
000000C5768FFBC0	E8 7A D6 3B BB B6 90 78 54 37 37 60 4B A5 58 BC	èzÖ;»¶.xT77`K¥X¼
000000C5768FFBD0	8F EB 7C 84 9E 4A F2 8D E0 95 4B 83 BA FF 6D E7	.ë|..Jò.à.K.ºÿmç
000000C5768FFBE0	FA 0F 88 3F 15 E7 D0 31 D4 2F 28 CC FD 06 F6 57	ú..?.çÐ1Ô/(Ìý.öW
000000C5768FFBF0	6A 4A 3B F4 89 1F 7C A7 D1 3A 33 8C C2 57 71 4B	jJ;ô..|§Ñ:3.ÂWqK
000000C5768FFC00	10 A5 F7 50 FF 00 C9 A7 F7 26 6B C6 1B 48 65 E5	.¥÷Pÿ.É§÷&kÆ.Heå
000000C5768FFC10	81 53 6F 17 08 61 F6 44 CD BE 50 AD 60 D4 50 F0	.So..aöDÍ¾P.`ÔPð

Figure 40

The malicious binary reads the file content via a function call to ReadFile:

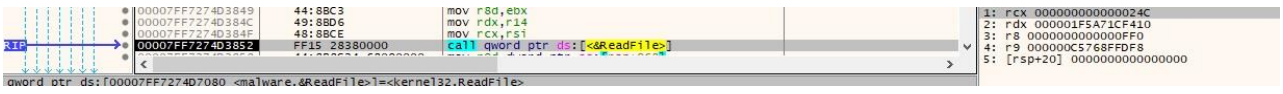


Figure 41

The content is encrypted using the 3DES algorithm. The implementation of the algorithm is

highlighted below:

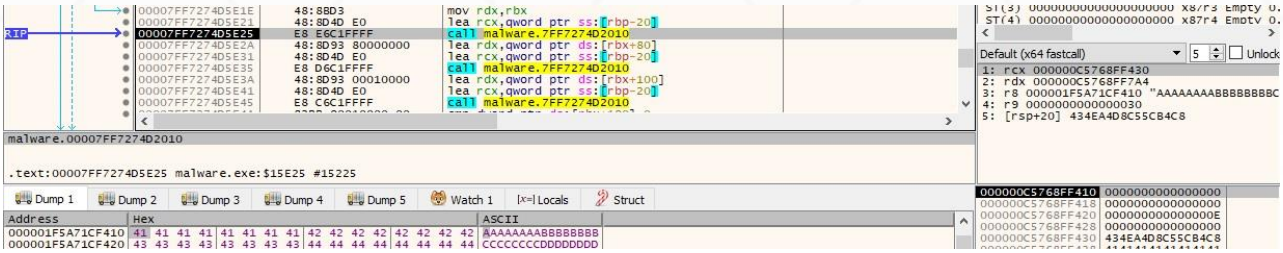


Figure 42

```
.text:00007FF7274D2010 sub_7FF7274D2010 proc near
.text:00007FF7274D2010
.text:00007FF7274D2010 arg_0= qword ptr  8
.text:00007FF7274D2010 arg_8= qword ptr  10h
.text:00007FF7274D2010 arg_10= qword ptr  18h
.text:00007FF7274D2010 arg_18= qword ptr  20h
.text:00007FF7274D2010
.text:00007FF7274D2010 mov     rax, rsp
.text:00007FF7274D2013 mov     [rax+8], rbx
.text:00007FF7274D2017 mov     [rax+10h], rbp
.text:00007FF7274D201B mov     [rax+18h], rsi
.text:00007FF7274D201F mov     [rax+20h], rdi
.text:00007FF7274D2023 mov     r9d, [rcx]
.text:00007FF7274D2026 mov     ebp, 0F0F0F0Fh
.text:00007FF7274D202B mov     r8d, [rcx+4]
.text:00007FF7274D202F mov     eax, r9d
.text:00007FF7274D2032 shr     eax, 4
.text:00007FF7274D2035 mov     rbx, rcx
.text:00007FF7274D2038 xor     eax, r8d
.text:00007FF7274D203B mov     rdi, rdx
.text:00007FF7274D203E and     eax, ebp
.text:00007FF7274D2040 mov     esi, 8
.text:00007FF7274D2045 xor     r8d, eax
.text:00007FF7274D2048 lea     rbp, cs:7FF7274C0000h
.text:00007FF7274D204F shl     eax, 4
.text:00007FF7274D2052 xor     r9d, eax
.text:00007FF7274D2055 mov     eax, r9d
.text:00007FF7274D2058 shr     eax, 10h
.text:00007FF7274D205B xor     eax, r8d
.text:00007FF7274D205E movzx   ecx, ax
.text:00007FF7274D2061 xor     r8d, ecx
.text:00007FF7274D2064 shl     ecx, 10h
.text:00007FF7274D2067 xor     r9d, ecx
.text:00007FF7274D206A mov     eax, r8d
.text:00007FF7274D206D shr     eax, 2
.text:00007FF7274D2070 xor     eax, r9d
.text:00007FF7274D2073 and     eax, 33333333h
.text:00007FF7274D2078 xor     r9d, eax
.text:00007FF7274D207B shl     eax, 2
.text:00007FF7274D207E xor     r8d, eax
.text:00007FF7274D2081 mov     r10d, r8d
.text:00007FF7274D2084 shr     r10d, 8
.text:00007FF7274D2088 xor     r10d, r9d
.text:00007FF7274D208B and     r10d, 0FF00FFh
```

Figure 43

The ransomware writes the encrypted file content, the RSA-encrypted 3DES key and IV, the initial file's length, and 4 bytes indicating that the file was encrypted:

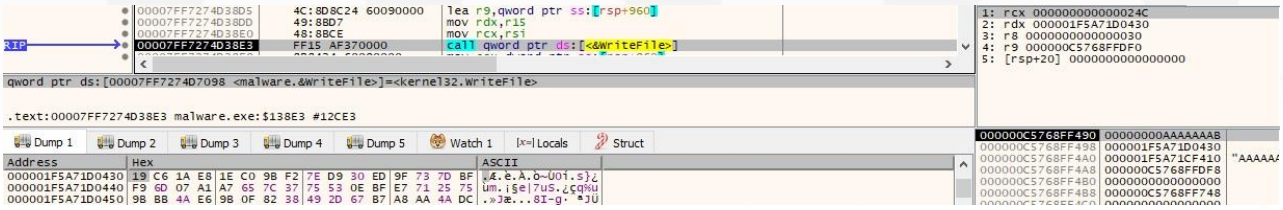


Figure 44

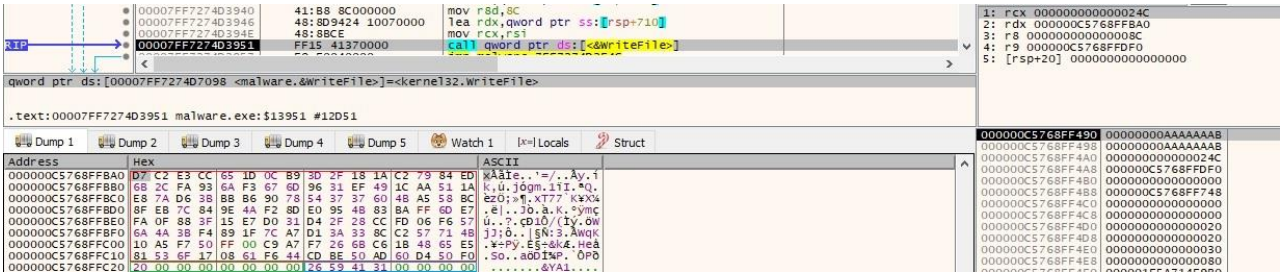


Figure 45

After the encryption is complete, the process creates a CMD file called “temp.cmd” and populates it with the following instructions:

Figure 46

Figure 47

```
temp.cmd
1  @Echo off
2  :rep
3  del %1
4  if not errorlevel 0 goto rep
5  for /F "tokens=*" %%1 in ('wevtutil.exe el') DO wevtutil.exe cl "%%1"
6  del %0
```

Figure 48

The path of the current executable is obtained using the GetModuleHandleA and GetModuleFileNameA APIs:

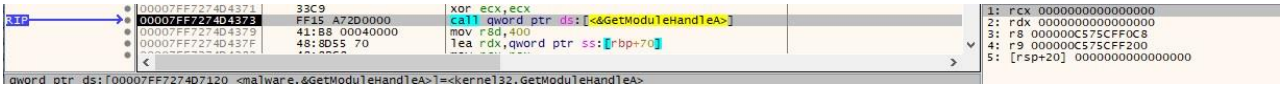


Figure 49

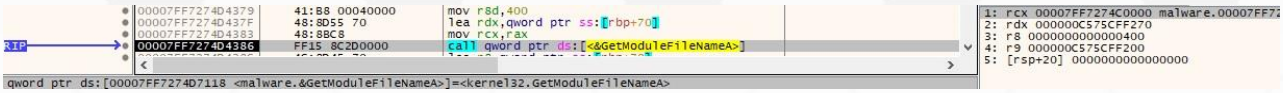


Figure 50

The purpose of the CMD file is to clear all event logs using wevtutil and to delete the initial executable as well as the file itself afterwards:

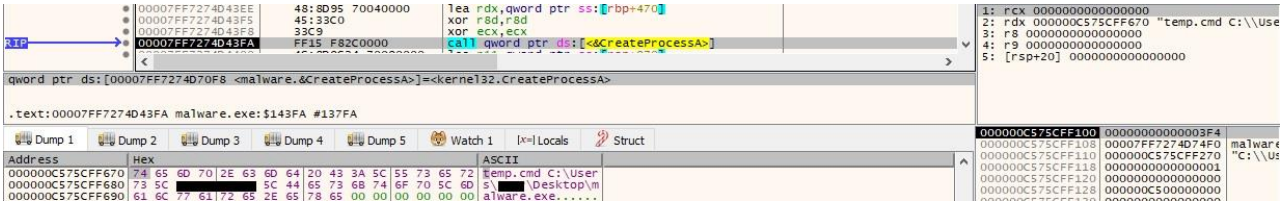


Figure 51

Indicators of Compromise

SHA256

d4a847fa9c4c7130a852a2e197b205493170a8b44426d9ec481fc4b285a92666

Underground Ransom Note

!!readme!!!.txt

Processes spawned

vssadmin.exe delete shadows /all /quiet

reg.exe add HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services /v MaxDisconnectionTime /t REG_DWORD /d <Value> /f

net stop MSSQLSERVER /f /m

temp.cmd <Executable path>