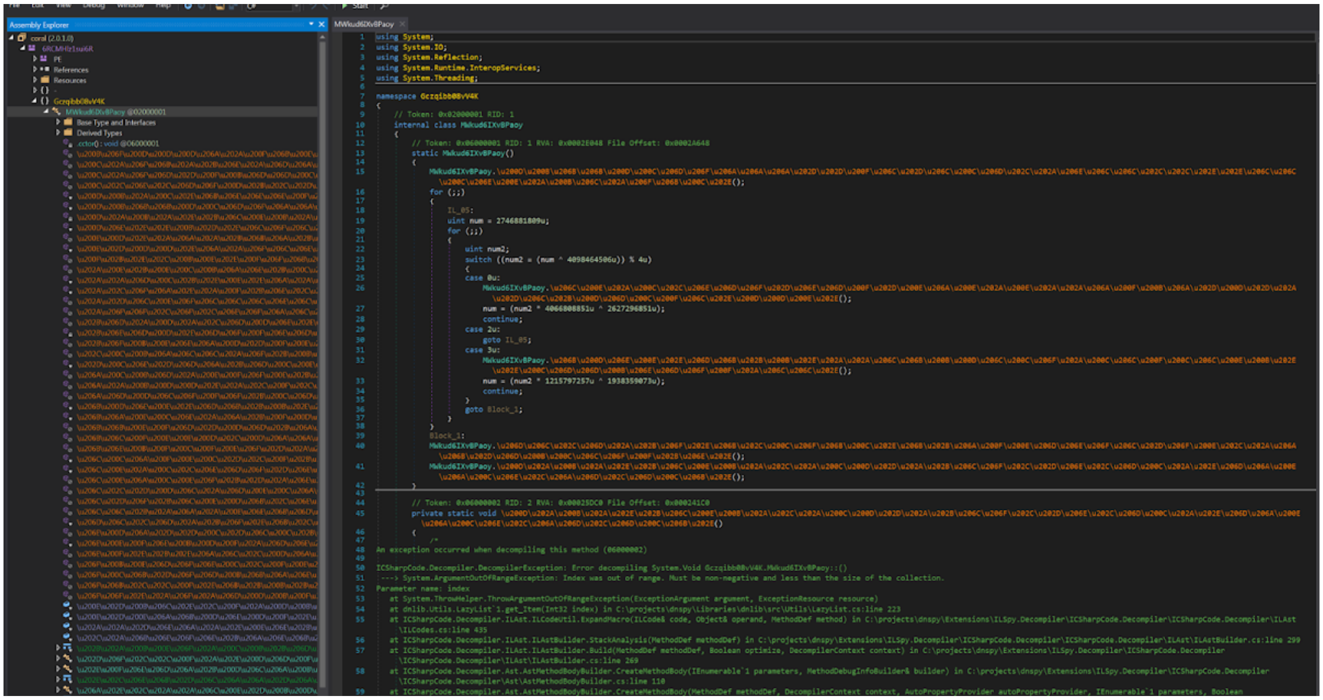
Recam Redux - DeConfusing ConfuserEx

blog.talosintelligence.com/2017/12/recam-redux-deconfusing-confuserex.html



This post is authored by Holger Unterbrink and Christopher Marczewski

Overview

This report shows how to deobfuscate a custom .NET ConfuserEx protected malware. We identified this recent malware campaign in our Advanced Malware Protection (AMP) telemetry. Initial infection is via a malicious Word document, the malware ultimately executes in memory an embedded payload from the Recam family. Recam is an information stealer. Although the malware has been around for the past few years, there's a reason you won't see a significant amount of documentation concerning its internals. The authors have gone the extra mile to delay analysis of the sample, including multiple layers of data encryption, string obfuscation, piecewise nulling, and data buffer constructors. It also relies on its own C2 binary protocol which is heavily encrypted along with any relevant data before transmission.

Technical Details

The Dropper

The word document (see above) uses common malware techniques, such as embedded VB code, to drop a .NET executable. We will not discuss these techniques further, but concentrate on the deobfuscation of the .NET malware dropper. The dropper is heavily obfuscated with a custom version of ConfuserEx, a free .NET Framework protector. On opening the binary in a .NET decompiler like dnSpy it is initially unreadable (see Fig. 1).

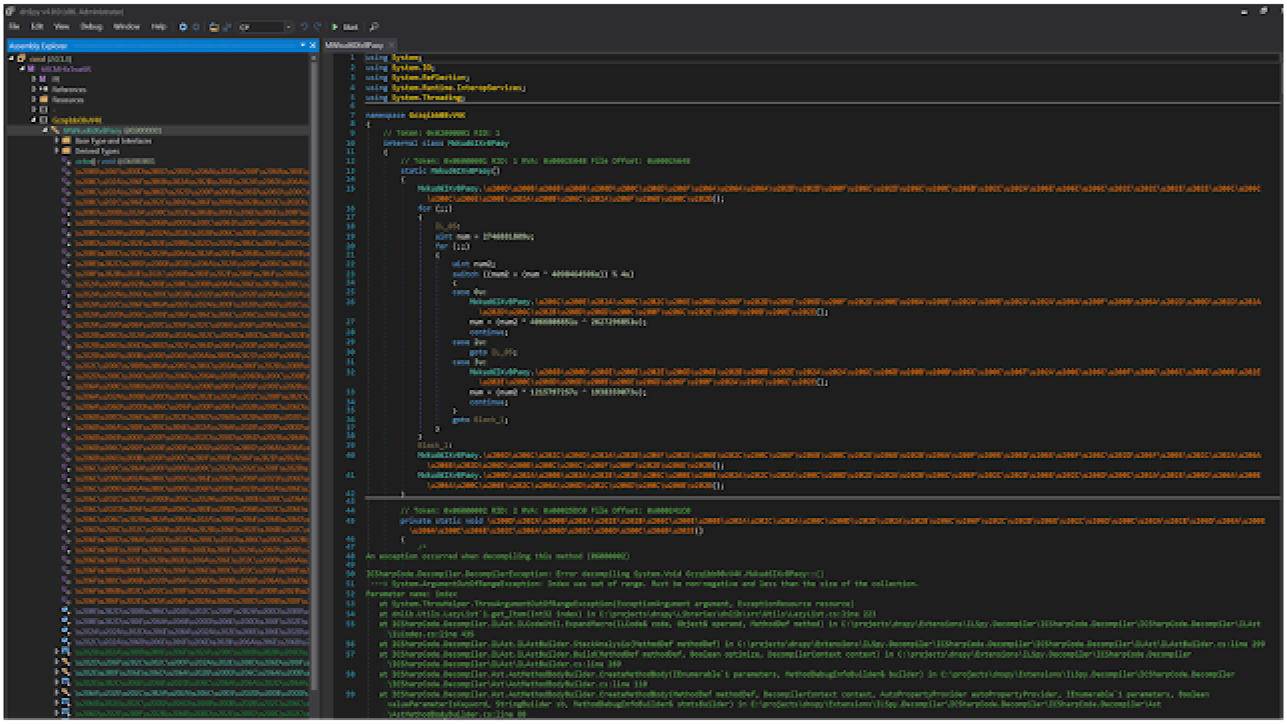


Fig. 1

There are a number of free deobfuscators available for ConfuserEx protected binaries; however, none of them are effective for this malware. Only some parts are able to be deobfuscated using these automated tools, leaving important sections of the binary unchanged, and breaking execution. This means we have no choice but to do it the hard way and deobfuscate it manually. There is documentation for manually unpacking ConfuserEx, but unfortunately, we hit bad luck again. The available documentation doesn't work with this version.

To get started, we first load the binary into dnSpy. We go to the <Module>. cctor and set a breakpoint on the last method (Fig 2). Now we can run the sample in our debugger and see that it has unpacked the first DLL ("ykMTM…" see Fig.2)

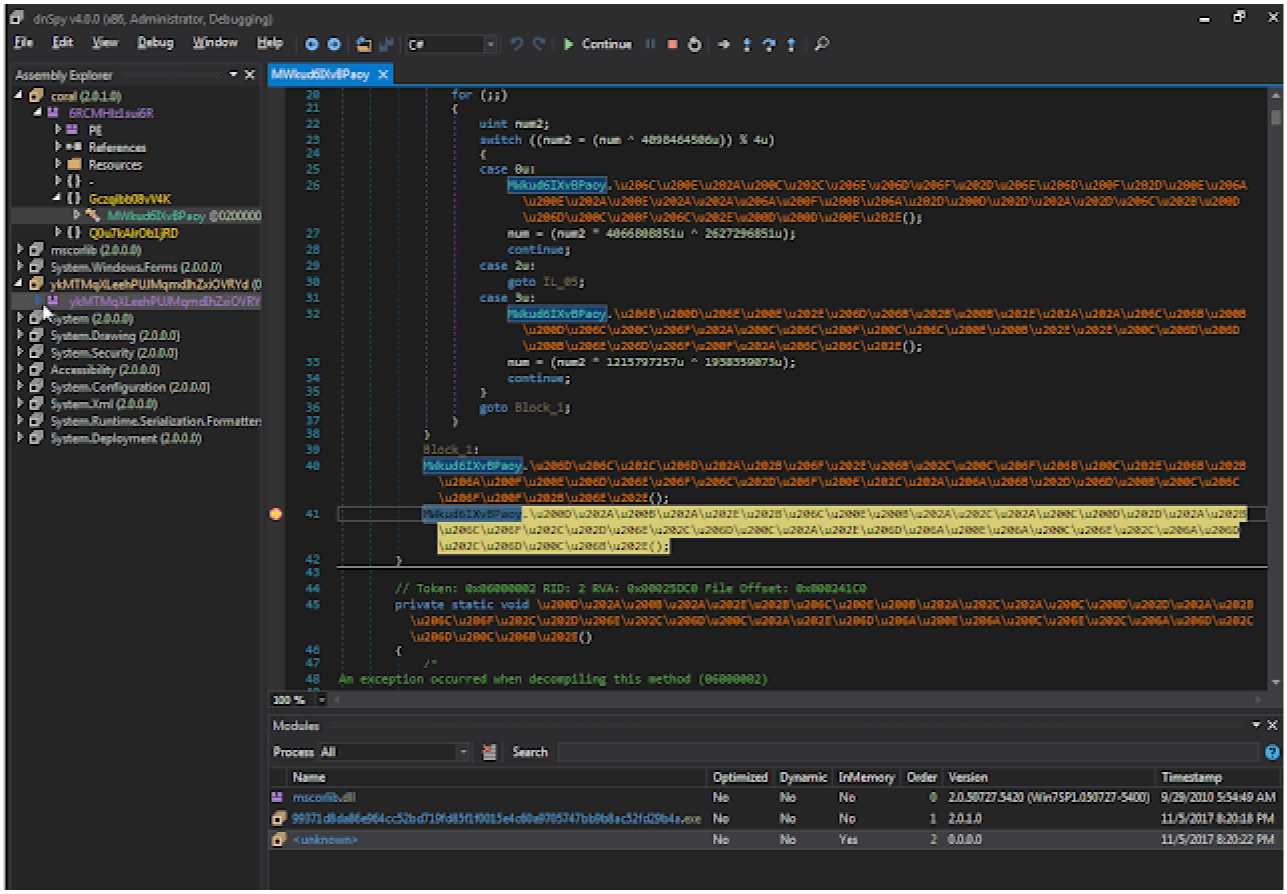


Fig. 2

We single step into the method where we hit the breakpoint and see in Fig. 3 that it has unpacked the next stage (coral).

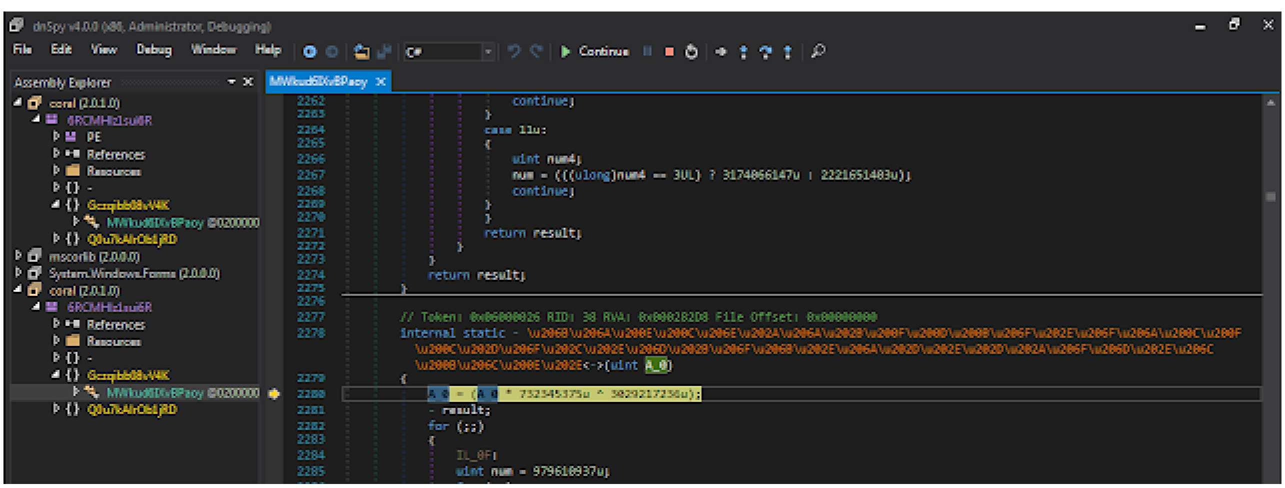


Fig. 3

We analysed this stage and found that we can set another breakpoint in the qMayiwZxj class on line 113 (see Fig. 4)

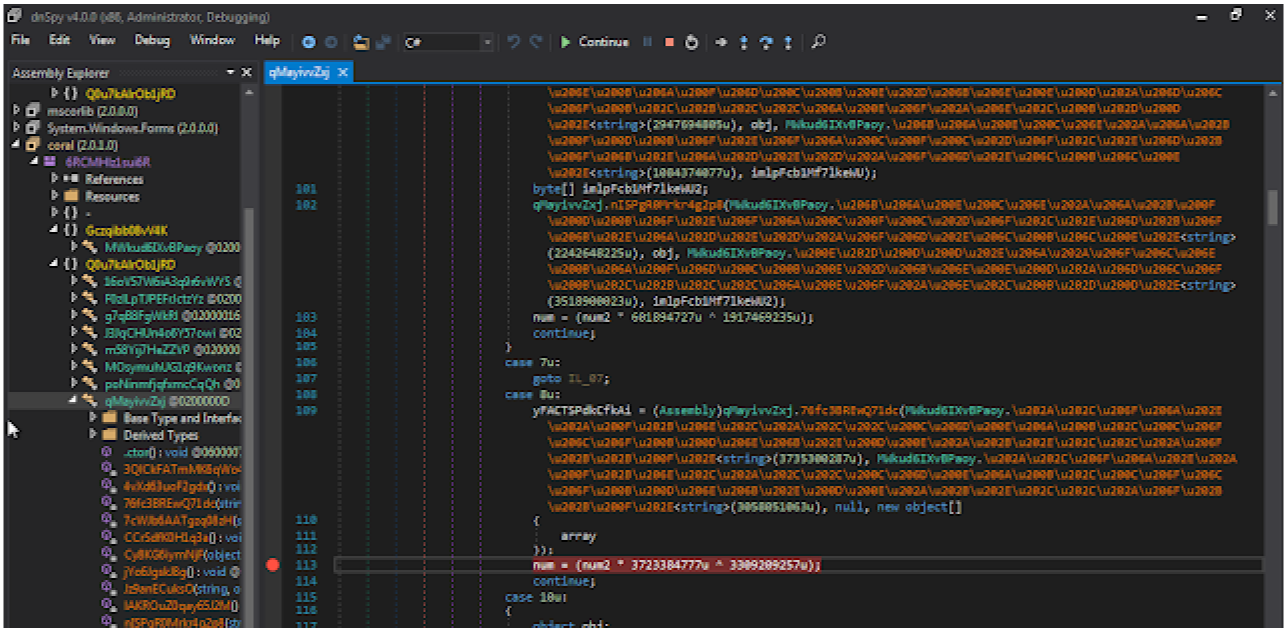


Fig. 4

This unpacks the next stage and we see the new unpacked stub.exe assembly (Fig. 5).

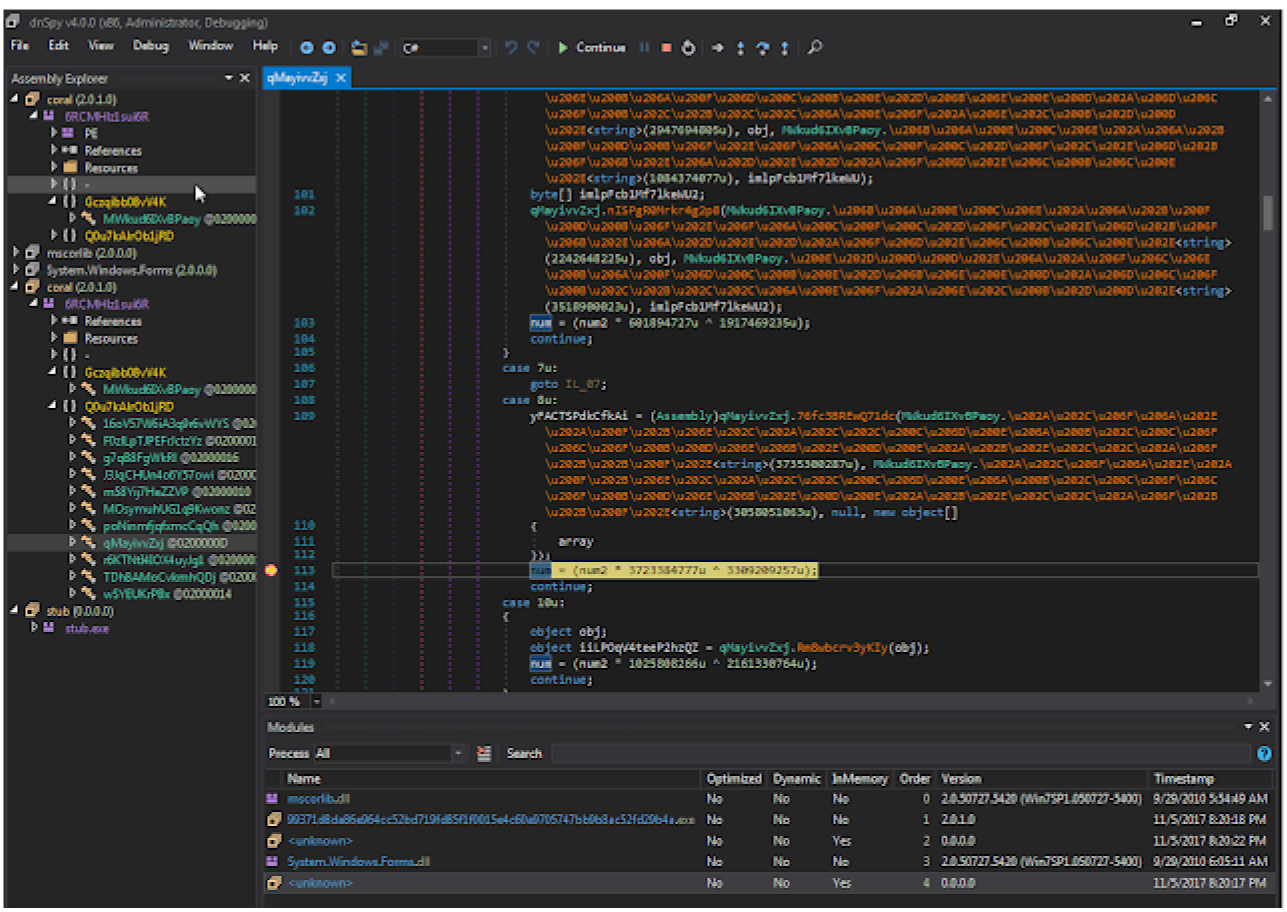


Fig. 5

If you have looked into other ConfuserEx obfuscated binaries, this looks familiar. Indeed, if you have a closer look, there is a well known friend, the gchandle.free() call on line 10082. This is our next breakpoint candidate. This call used to be the end of an unpacking stage in previous versions.

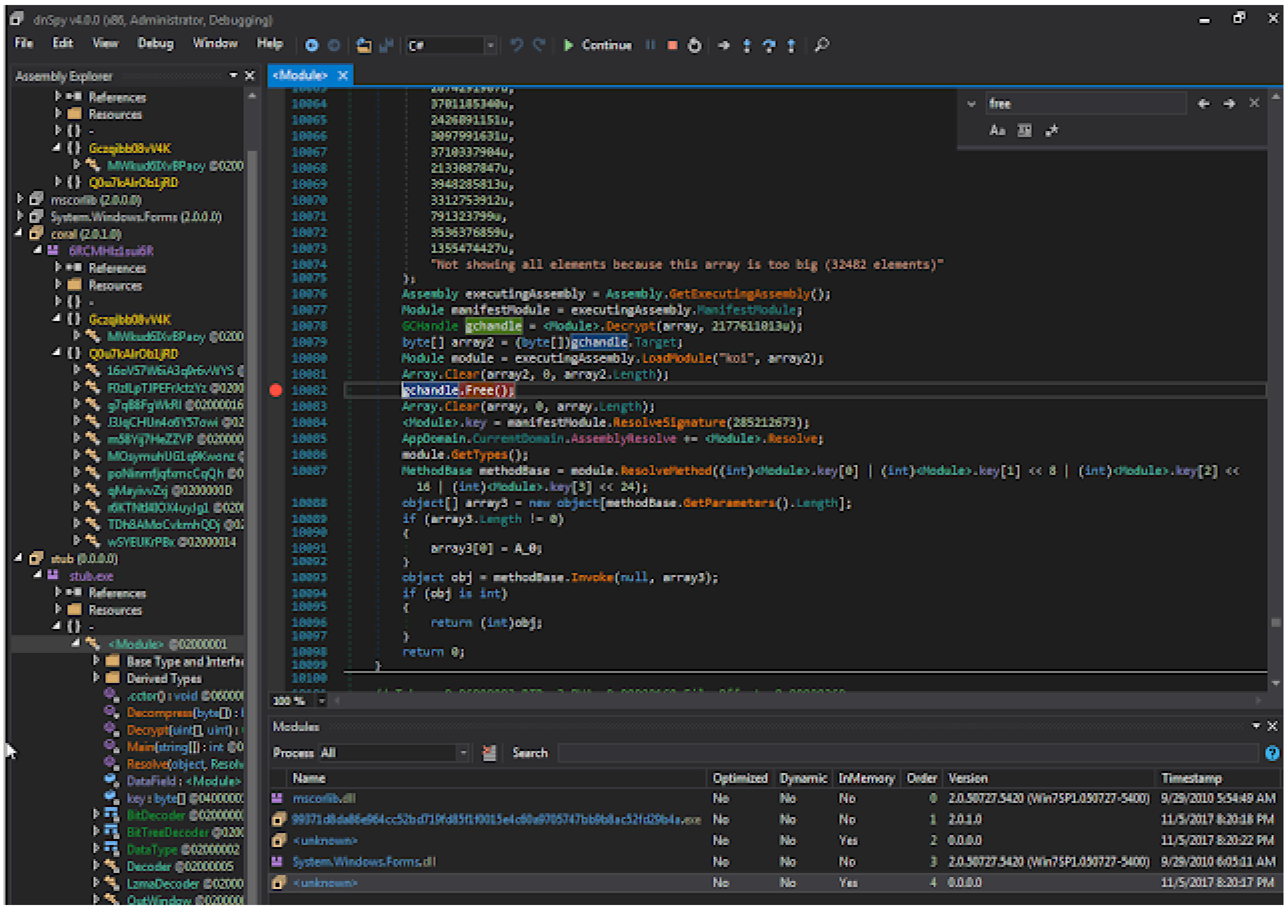


Fig. 6

As expected, this unpacks another module ConfuserEx is known for: koi.

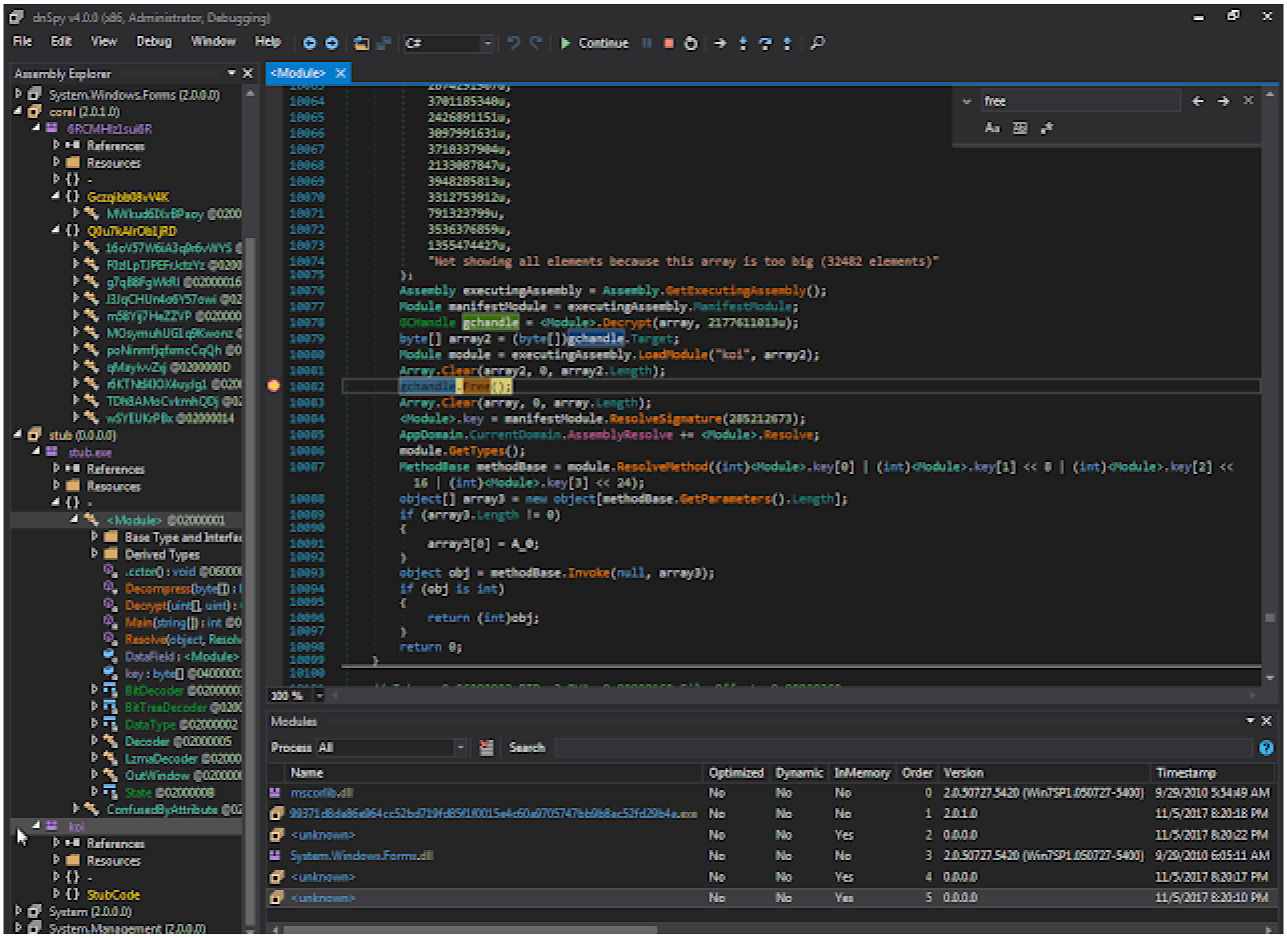


Fig. 7

We are getting closer, but the classes in koi are still empty and not yet filled with code:

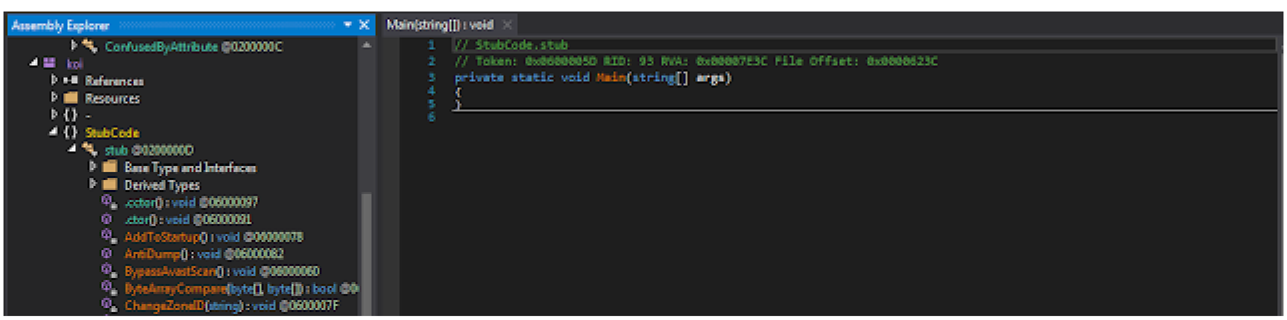


Fig. 8

Again, we set a breakpoint on the last method called in koi's cctor and proceed running the sample.

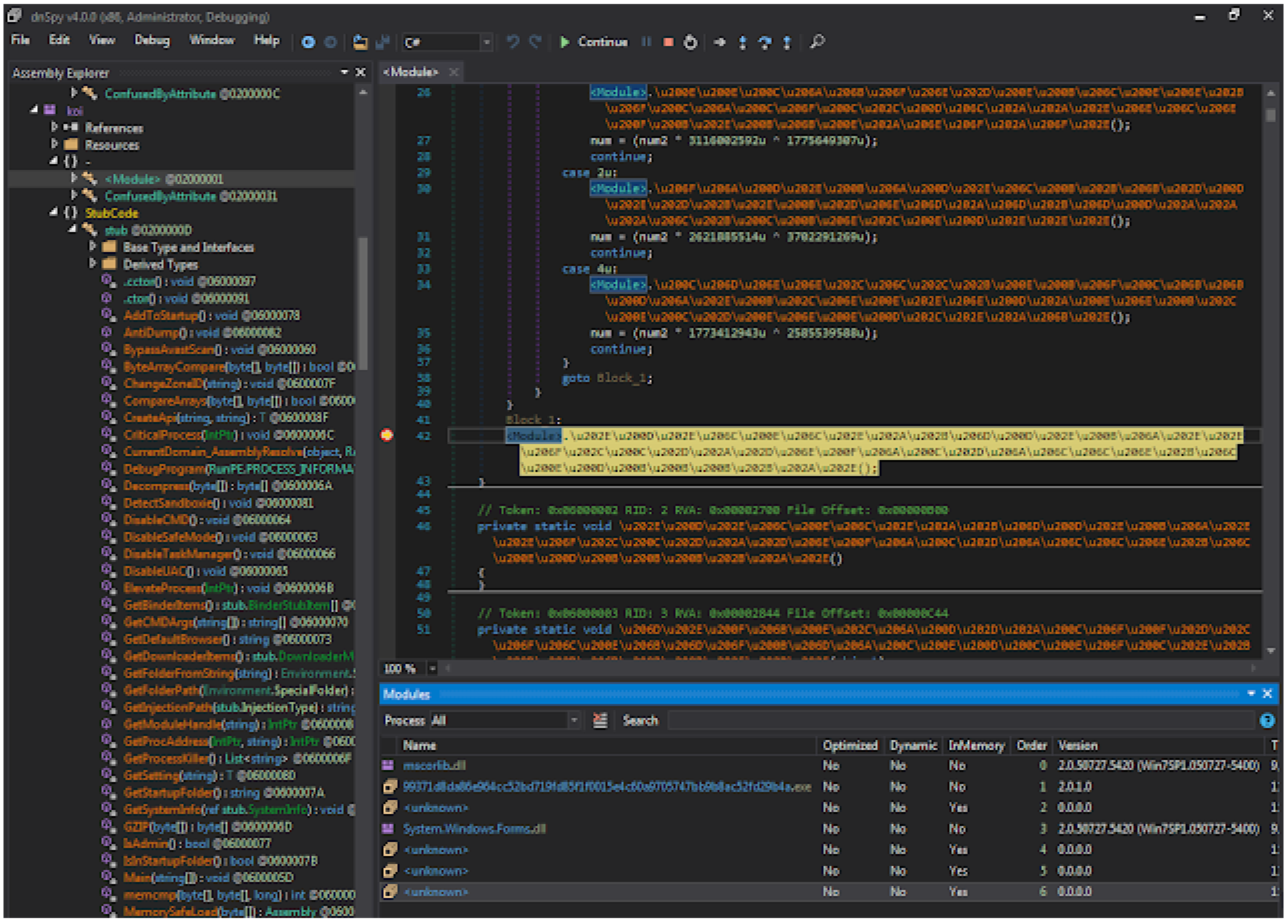


Fig. 9

Nice, another DLL is unpacked, unfortunately it is nothing important. Our Main class and most others in stub are still empty. Single stepping, brings us back into <module>. Once there, we analysed the methods and found out that we can set another breakpoint at line 92 for unpacking the next stage (see Fig. 10).

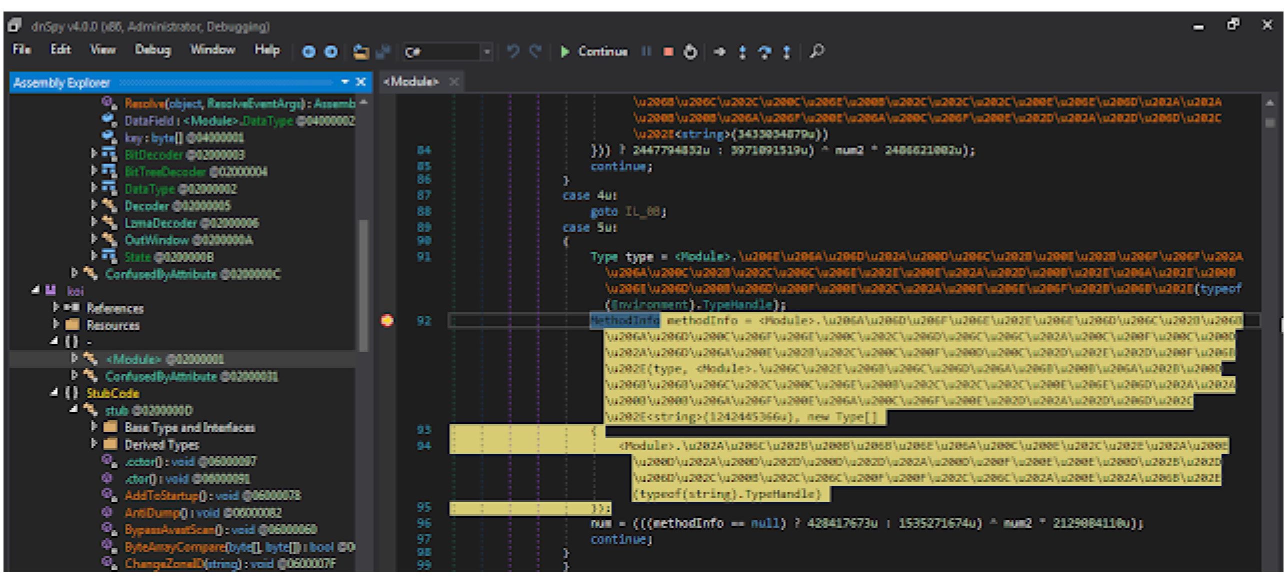


Fig. 10

Tada! If we now look in stub at the classes, they are filled with code. Now we can set a breakpoint on stub.Run() and start investigating what this malware loader is actually doing besides unpacking itself.

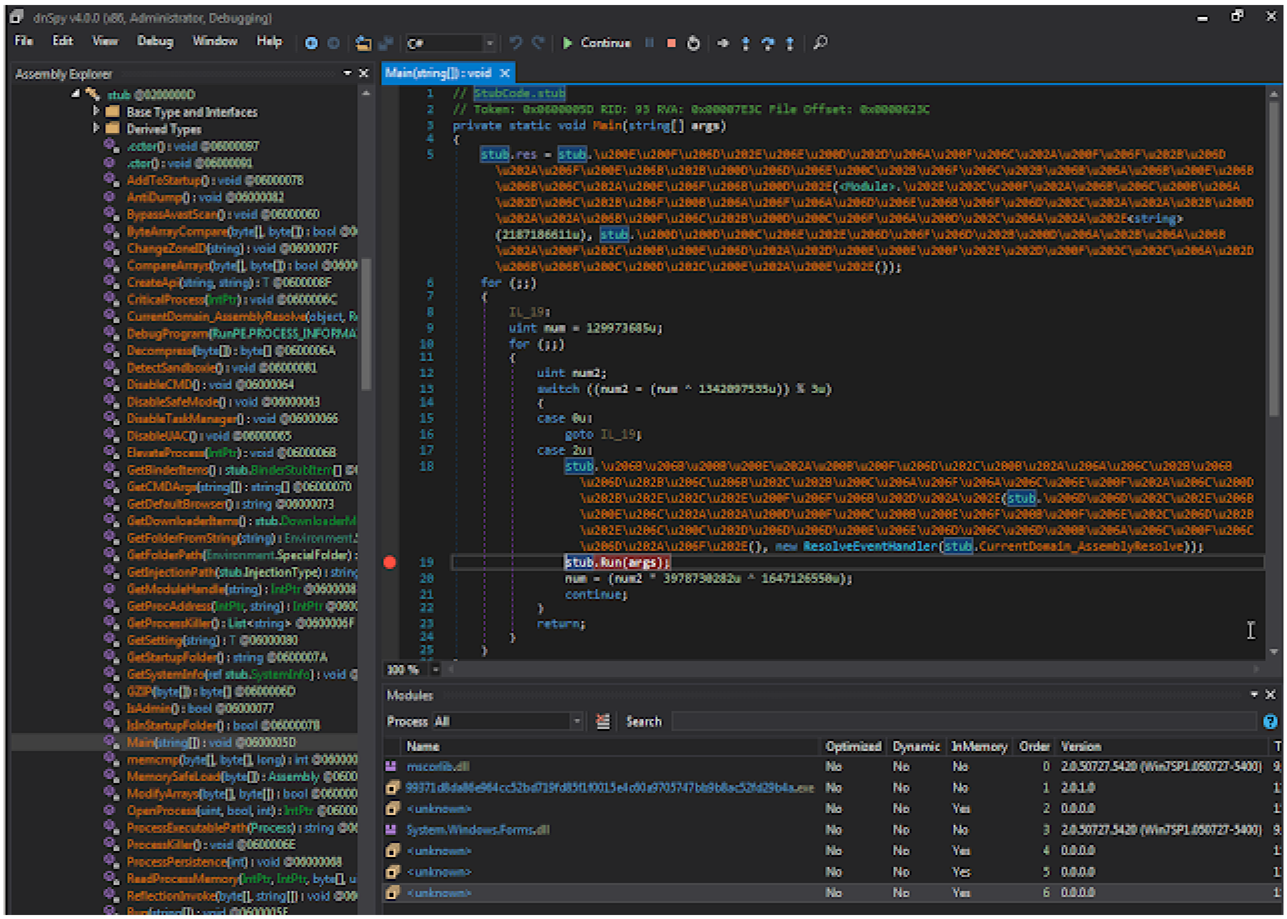


Fig. 11

We see that it is attempting to bypass some AV scans and reading several config parameters from the resource section. Below you can see the malware's configuration which was hidden encrypted in the resource section (Fig. 12) before unpacking.

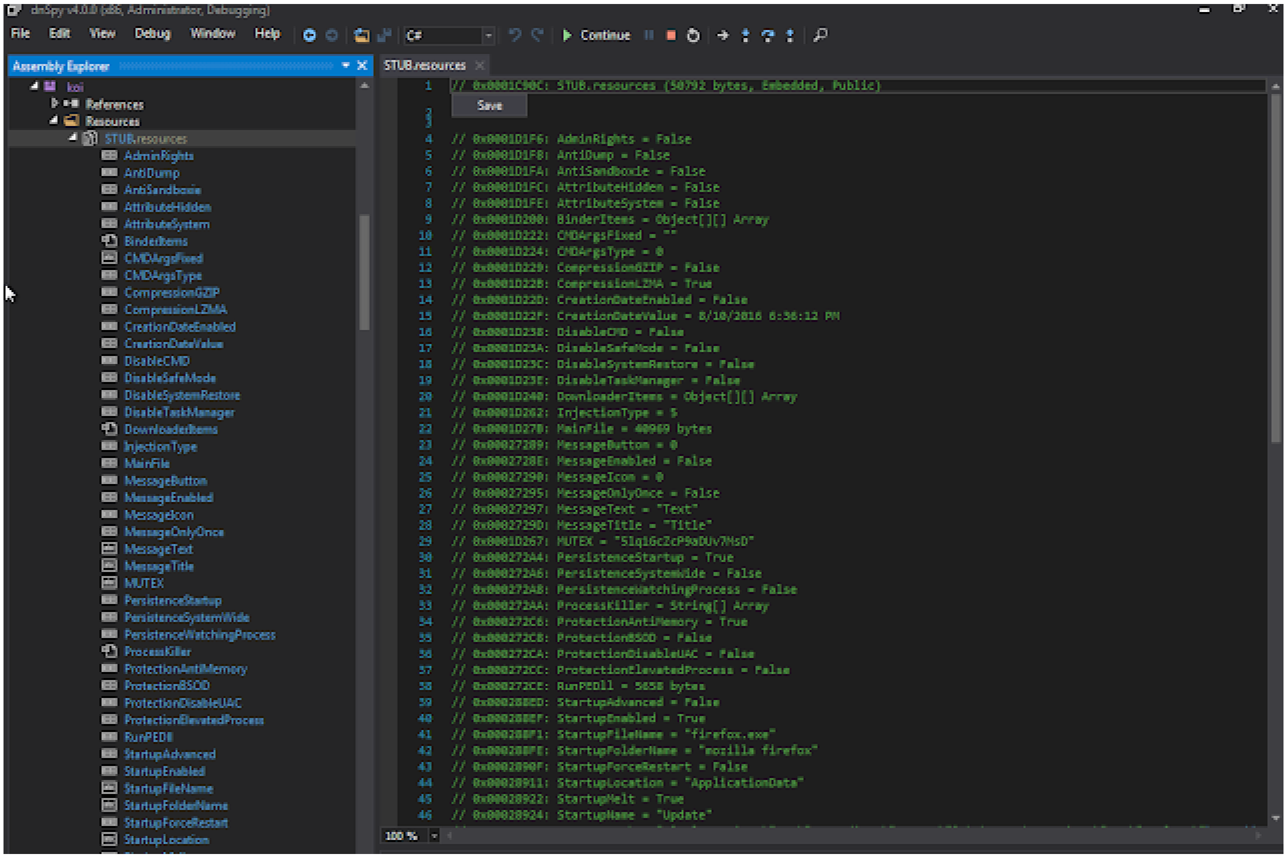


Fig. 12

It checks if it was executed from the Startup folder (e.g. %AppData%\mozilla firefox\firefox) as configured in the resource section. If not, it copies itself to the Startup folder and launches itself via cmd.exe. This means, we need to stop debugging and start again by loading the firefox.exe from %AppData%\mozilla firefox\firefox into dnSpy, following the unpacking again up to this point.

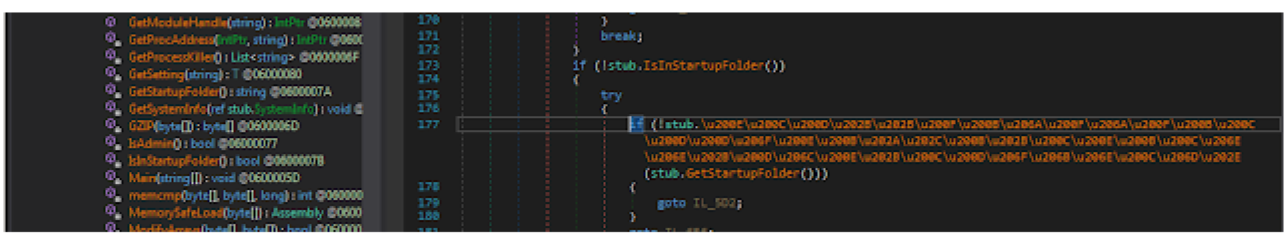


Fig. 13

Now we are in the "is executed from Startup location" branch. Here it gets interesting. First it makes itself persistent on the local machine. As you can see below, it writes a file called Update.txt with the following content to the %AppFolder%.

--- snip ---
C:\Users\dex\AppData\Roaming\mozilla firefox\firefox.exe
exit
--- snip ---

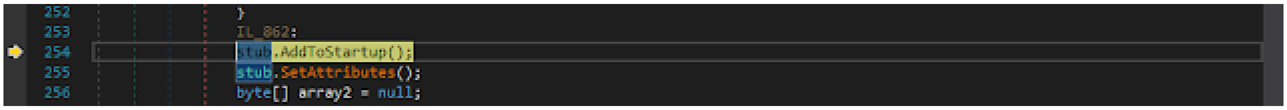


Fig. 14

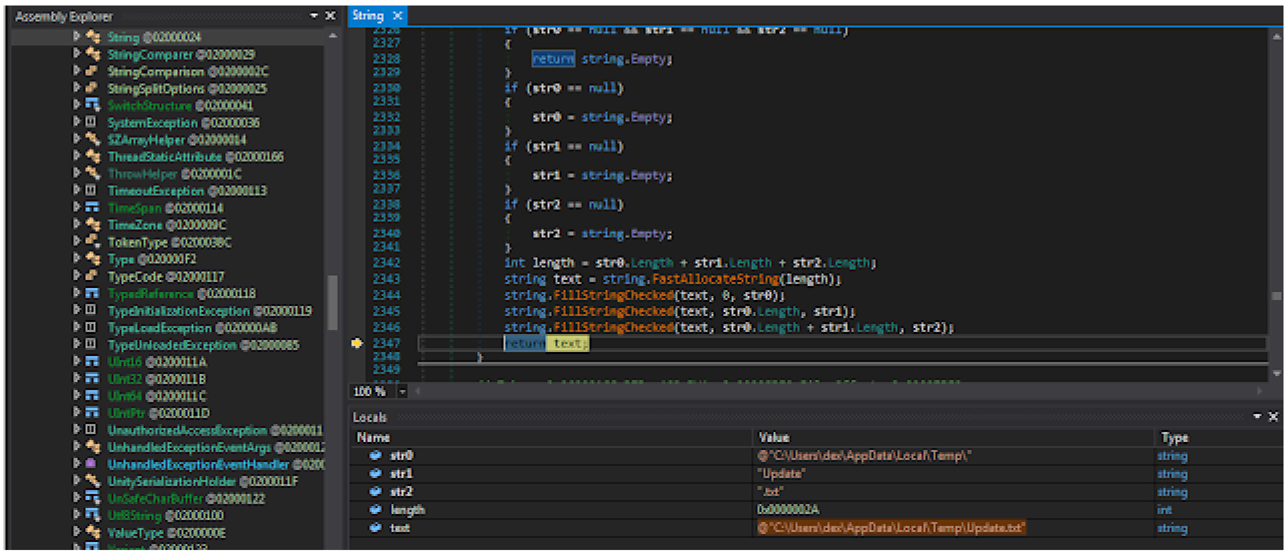


Fig. 15

Then it adds this file to the auto-run registry key by executing reg add in a cmd.exe to make sure the firefox.exe file gets executed at PC start up:

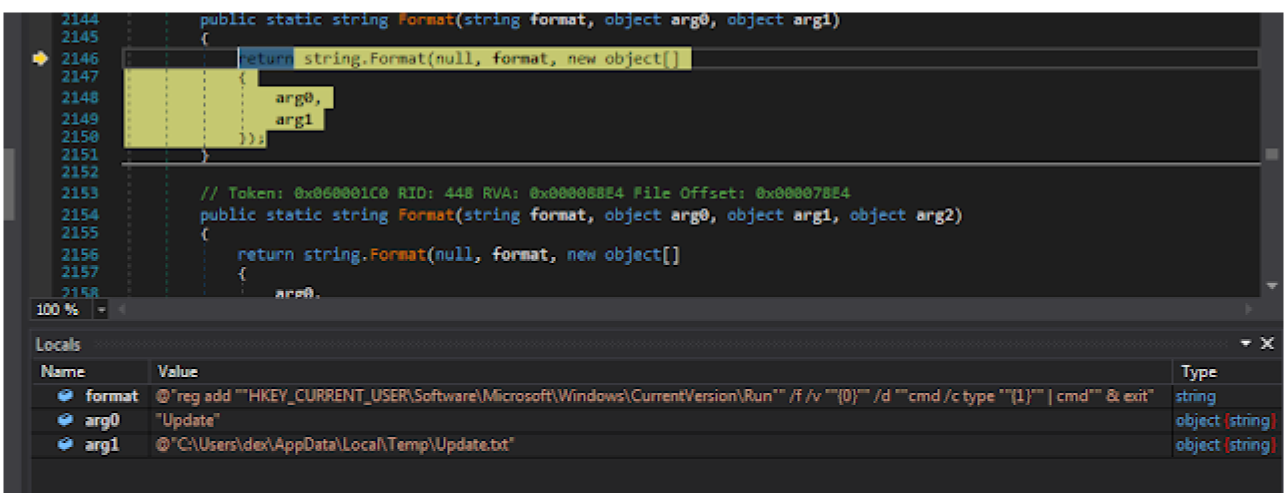


Fig. 16a

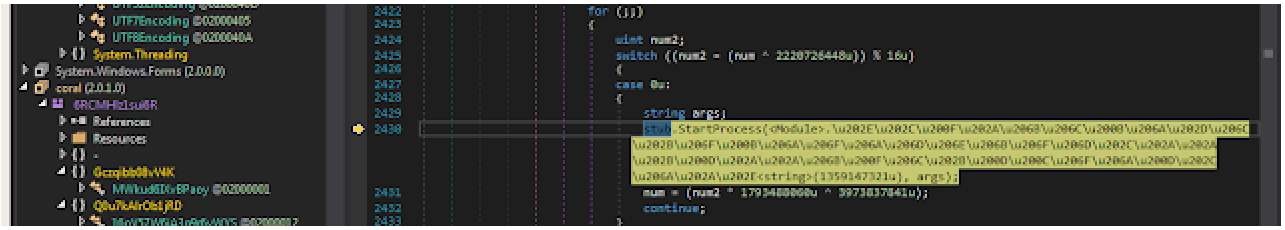


Fig. 16b

It executes a couple of other methods based on the configuration and then loads and decompresses the LZMA compressed malware payload file (Recam) from the resources MainFile section. After a couple of runtime fixes it loads RunPEDLL.dll and tries to inject the file into the user's browser. In case this fails (e.g. no browser is running), it injects the file into itself (firefox.exe). In both cases the RunPE.Run() method is used to do that.

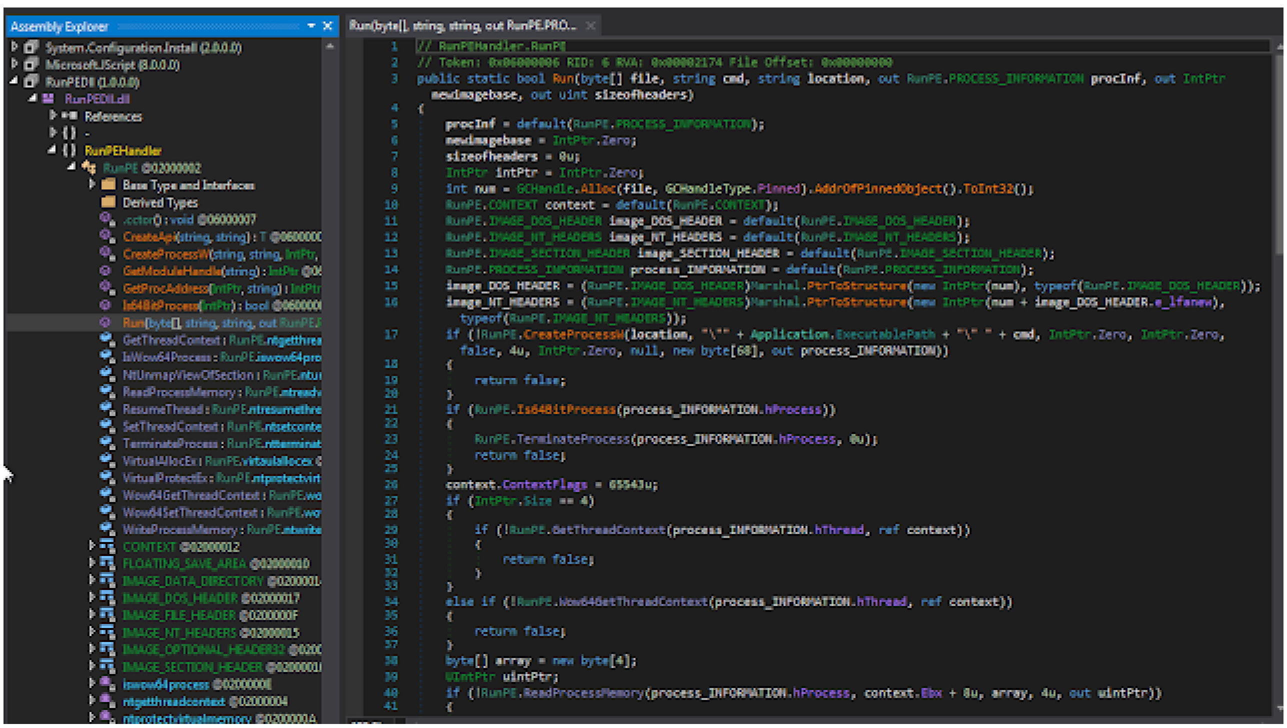


Fig. 17

From here on the work is done for the malware dropper and the loaded Recam binary takes over.

Payload

As mentioned in the introduction, the authors have gone the extra mile to frustrate analysis of the sample by using multiple obfuscation techniques, including multiple layers of data encryption, string obfuscation, piecewise nulling, and data buffer constructors. It also relies on its own C2 binary protocol. All relevant data is heavily encrypted before transmission.

The dropped binary is packed with vanilla UPX. This part is easy to unpack; the tricky part comes in the next stage. After the original Entry Point (OEP) is restored, it begins with some homebrew cryptographic initialization for several values that get used consistently throughout runtime. Most remain constant following the initialization routine, but some change over time. Some preliminary string deobfuscation occurs shortly thereafter and includes a single hard-coded Command and Control server (C2) IP.

```
call Recam_string_decode2
mov [esp+12Ch+var_12C], ebx
mov [esp+12Ch+len], 0FFh
mov [esp+12Ch+ciphertext], offset decode2_var_len255
call Recam_string_decode2
mov [esp+12Ch+var_12C], ebx
mov [esp+12Ch+len], 20h
mov [esp+12Ch+ciphertext], offset aPassword ; "Password"
call Recam_string_decode2
mov [esp+12Ch+var_12C], ebx
mov [esp+12Ch+len], 10h
mov [esp+12Ch+ciphertext], offset HostID_plus_rand ; "HostId-%Rand%Â°Ã¥"
call Recam_string_decode2
mov [esp+12Ch+var_12C], ebx
mov [esp+12Ch+len], 8
mov [esp+12Ch+ciphertext], offset mutex_name
```

This less frequently used deobfuscation routine is primarily based on a single-byte XOR loop. The other primary routine is JIT based and relies on a hard-coded decode key. Fortunately, IDA Pro's Appcall feature made short work of these obfuscations.

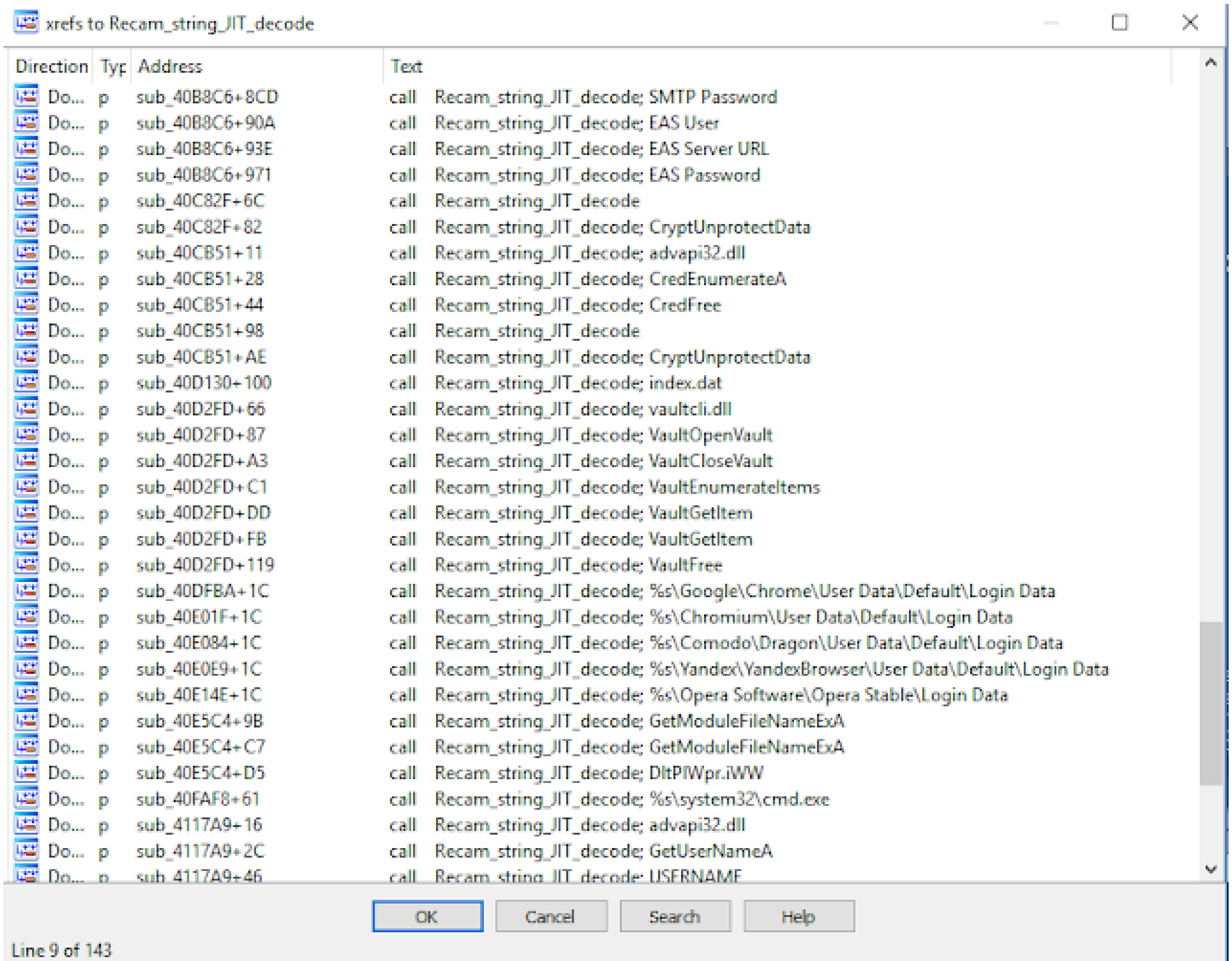


Fig. 18

Getting to the end of the preamble functions shortly following the PE Entry Point (EP), we get to an operation selection routine. The presence of unnecessary code and calculations disguises the fact that the jump to location 40849B will always be taken and the apparently interesting code that appears to involve file mangling and process creation is merely a decoy and always skipped in execution.

lea ebx, [esp+83Ch+var_618]
mov [esp+83Ch+lpValueName], 204h
mov [esp+83Ch+uExitCode], offset decode2_unk_len128
mov [esp+83Ch+Mode], ebx
call Recam_getenv
mov [esp+83Ch+uExitCode], 1
call Recam_arg0_AND_constant
test al, al
jz loc_40849B ; jmp taken (skip mangling & proc creation)

Recam_arg0_AND_constant proc near

```
var_1C= dword ptr -1Ch
arg_0= dword ptr 4

sub esp, 1Ch
mov [esp+1Ch+var_1C], offset flow_constant3
call Recam_base10_to_base16
and eax, [esp+1Ch+arg_0]
cmp eax, [esp+1Ch+arg_0]
setz al
add esp, 1Ch
retn
```

Moving forward, the malware sets a Run key for system persistence. Near the end of the operations function, an additional thread is created to start up a keylogger component, logging to %APPDATA%\Logs with <DAY>-<MONTH>-<YEAR> as the file name format. Logged input is stored in the commonly seen bracket delimiters. However, as one might expect by now, the final data is encrypted before written to the file on disk.

Next, the malware will create an ID file entitled .Identifier. If such a file already exists in the PWD of the sample (extracted via the GetModuleFilename API), it is simply read in instead of created from scratch.

Fig. 19

Data to be written to the file is generated piece by piece and results in the following format:

(4 bytes) Static ID

(13 bytes) HostId-<6 character alphanumeric rand, seeded from system time>

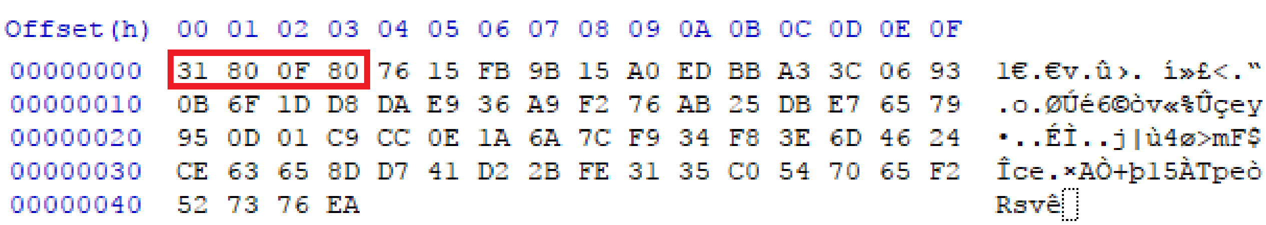
(19 bytes) 19 null bytes

(19 bytes) system time OR local time (19 byte format)

(13 bytes) 13 null bytes

Fig. 20

Note that since a static ID is used, the first 4 bytes of the file always remain the same (the cryptography used for the C2 data is much more complex).



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  31 80 0F 80 76 15 FB 9B 15 A0 ED BB A3 3C 06 93  1€.€v.û›. í»£<.“
00000010  0B 6F 1D D8 DA E9 36 A9 F2 76 AB 25 DB E7 65 79  .o.ØÚé6©òv«%Ûçey
00000020  95 0D 01 C9 CC 0E 1A 6A 7C F9 34 F8 3E 6D 46 24  •..ÉÌ..j|ù4ø>mF$
00000030  CE 63 65 8D D7 41 D2 2B FE 31 35 C0 54 70 65 F2  Îce.×AÒ+þ15ÀTpeò
00000040  52 73 76 EA                                      Rsvê
```

Fig. 21

```
mov     [esp+4CCh+Filename], ebp
call    Recam_string_decode
xor     edx, edx
cmp     dword ptr [ebx], 61985734h ; RECAM ID
jz      loc_40833F
```

Fig. 22a

```
lea     eax, [ebx+4]
mov     dword ptr [ebx], 61985734h ; RECAM ID
mov     [esp+4CCh+Count], 32
mov     [esp+4CCh+Mode], offset HostID_plus_rand ; "RÄ¦+8+¬Yr\nEè\b}%W"
mov     [esp+4CCh+Filename], eax
call    Recam_write_to_buffer ; Appends 'HostId-<6_alphanum_rand>'
```

Fig. 22b

As with the .Identifier file, the initial C2 beacon will also always be 68 bytes in length. Each C2 message (both client & server) will use the following format:

(4 bytes) Length of data following these bytes
(1 byte) C2 command
(n bytes) Data relevant to the command

It's often easiest to break on a few instructions prior to deciphering the C2 beacon for many malware families these days. Whether it was intentional or not, the authors decided to opt of a homebrew crypto scheme allowing for randomized beacon data for each run (only the length bytes & C2 command for the beacon remain the same), or their homebrew crypto implementation is severely complex & broken.

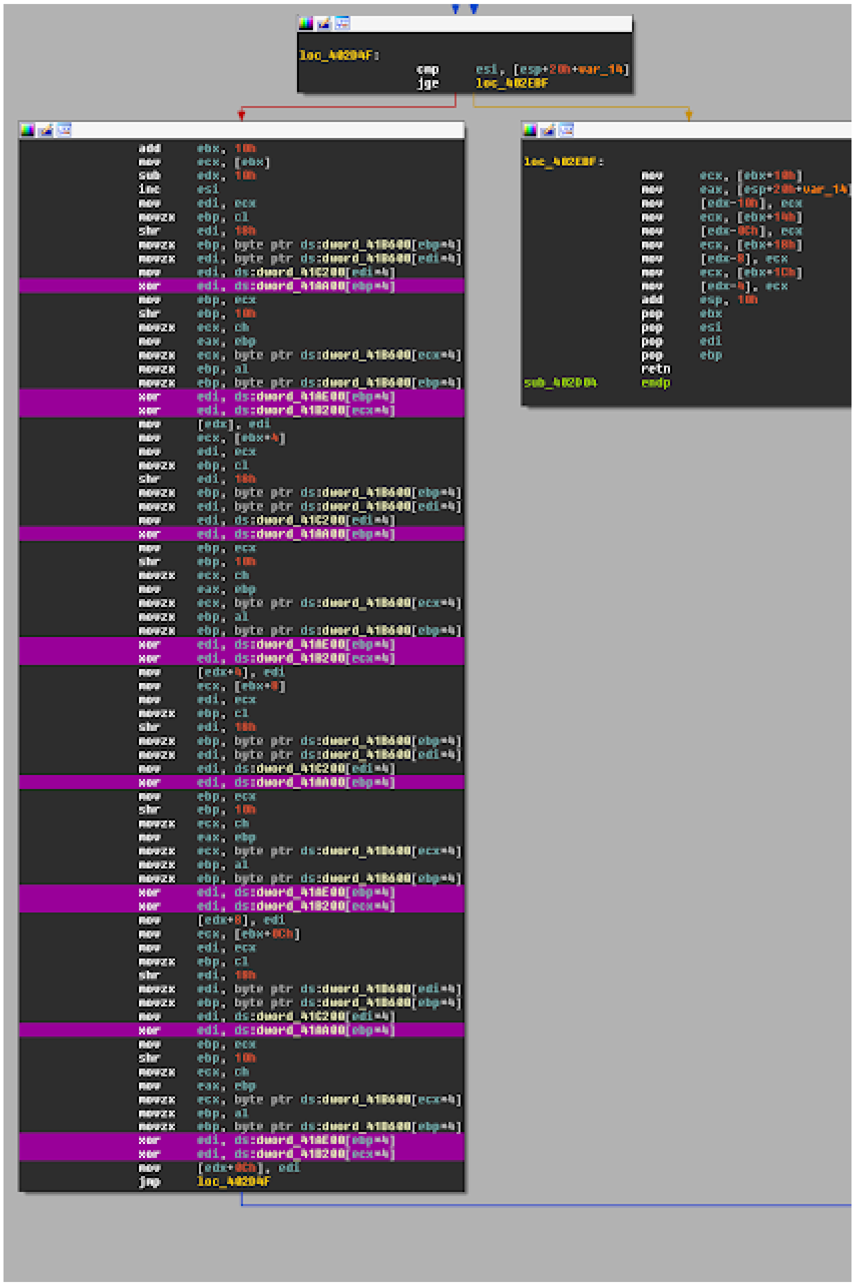


Fig. 23

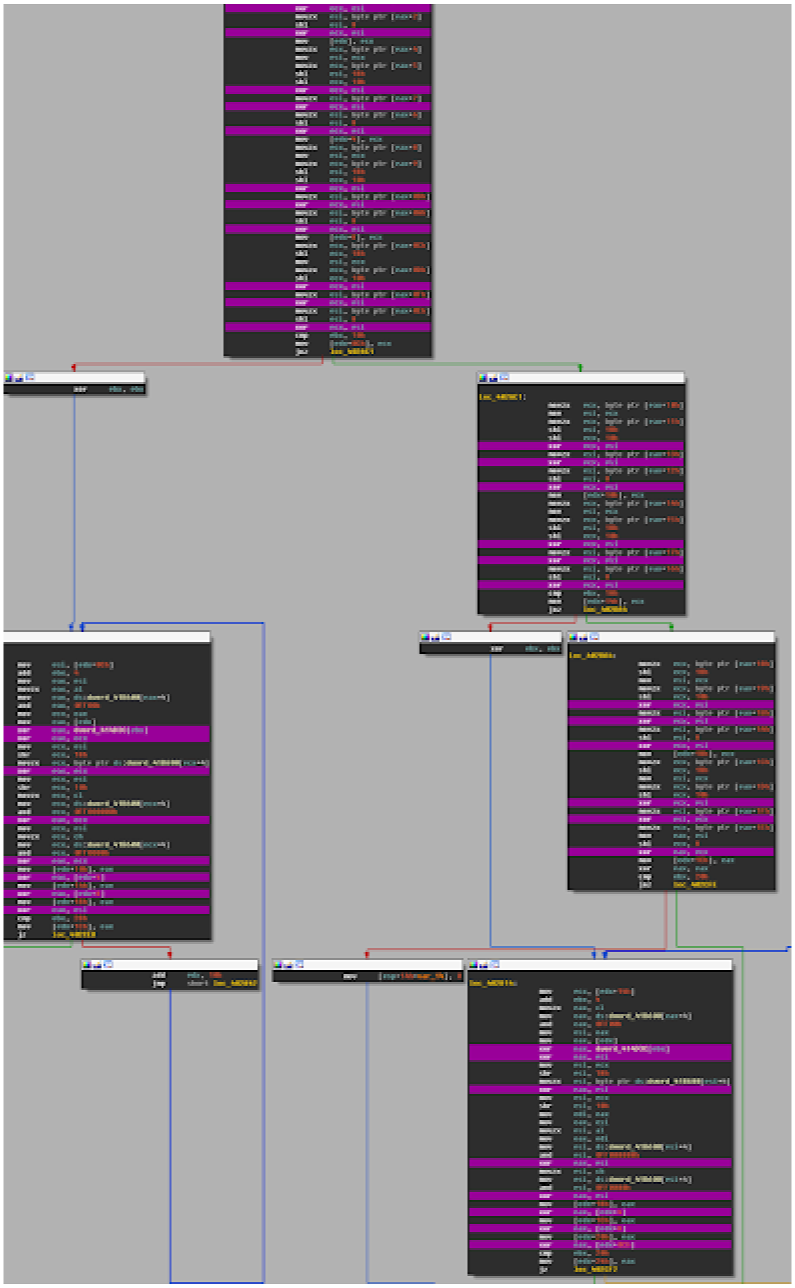


Fig. 24

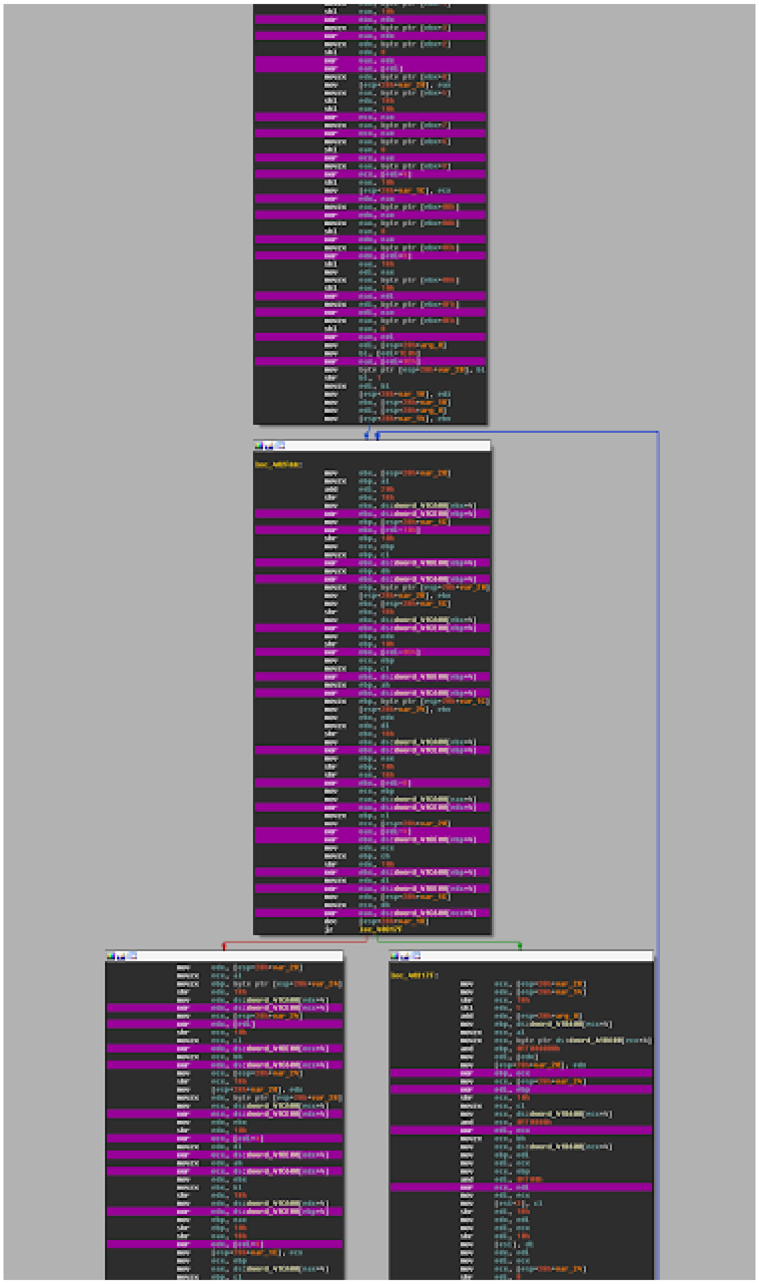


Fig. 25

Once the beacon is sent, the sample waits for a server response. The C2 we encountered is now down and resetting connections, but pcaps captured in sandbox environments at an earlier date can give us a better idea of what to expect for the rest of the communications. The following example shows the beacon, the initial response, one additional client transmission and a series of "keep alive" messages consisting of the sole command byte.

```
00000000  41 00 00 00 83 92 98 5f  b8 8e f8 48 e3 a8 4f 5f A......_ ...H..O_
00000010  b0 6c f8 ac 40 ad bb bd  d9 86 c1 a0 48 d0 48 d2 .l..@... ....H.H.
00000020  84 80 86 a0 58 22 a1 66  fc 5f dd d6 ee 38 c4 b6 ....X".f ._...8..
00000030  dc 4c af bb d3 a3 1b 73  09 c9 82 b5 85 85 f1 60 .L.....s .......`
00000040  91 29 79 dc 2a                                    .)y.*
    00000000  41 00 00 00 85 65 fe 8d  79 ac 99 49 c2 d2 9d 47 A....e.. y..I...G
    00000010  5c 59 b4 d4 3c 3d cb a9  6f 94 ad e7 7a 37 64 2d \Y..<=.. o...z7d-
    00000020  fa ed df a1 98 be cd d3  b4 4b 73 22 9b 12 4b c5 ........ .Ks"..K.
    00000030  2a ad 59 76 d9 6f 76 22  53 70 b2 ca bd 3b 6b 97 *.Yv.ov" Sp...;k.
    00000040  7f 53 64 28 b6                                    .Sd(.
00000045  3c 00 00 00 85 58 10 0a  85 88 a0 2e 8d 63 c3 40 <....X.. .....c.@
00000055  50 7f 48 73 0c 2b 3e 12  18 54 f5 4e 70 48 72 28 P.Hs.+>. .T.NpHr(
00000065  37 7e 91 f6 4d 16 53 b3  f1 f0 34 52 7d 49 6d c7 7~..M.S. ..4R}Im.
00000075  46 80 52 83 82 fb 15 c4  14 6c b7 45 9e 60 9b 38 F.R..... .l.E.`.8
    00000045  01 00 00 00 81                                    .....
00000085  01 00 00 00 81                                    .....
    0000004A  01 00 00 00 81                                    .....
0000008A  01 00 00 00 81                                    .....
    0000004F  01 00 00 00 81                                    .....
0000008F  01 00 00 00 81                                    .....
    00000054  01 00 00 00 81                                    .....
00000094  01 00 00 00 81                                    .....
    00000059  01 00 00 00 81                                    .....
00000099  01 00 00 00 81                                    .....
    0000005E  01 00 00 00 81                                    .....
0000009E  01 00 00 00 81                                    .....
    00000063  01 00 00 00 bd                                    .....
000000A3  01 00 00 00 bd                                    .....
    00000068  01 00 00 00 81                                    .....
000000A8  01 00 00 00 81                                    .....
    0000006D  01 00 00 00 c1                                    .....
000000AD  01 00 00 00 c1                                    .....
    00000072  01 00 00 00 c1                                    .....
000000B2  01 00 00 00 c1                                    .....
    00000077  01 00 00 00 81                                    .....
000000B7  01 00 00 00 81                                    .....
    0000007C  01 00 00 00 81                                    .....
```

Fig. 26

At this point, code execution depends on a flow state that is set only a few times throughout the binary (initially set to 0xFFFFFFFF). As far as the response length and C2 command are concerned, this state further dictates which each attribute must be. For example, the function responsible for checking the response length checks the flow state too. If state has changed, it checks if the message length exceeds 0x30000. If it's still in the default state, it checks if the length is 0x41 (length of the beacon message and its expected response). For the command byte itself, the default state checks if the command byte is set for the beacon phase of the communications (0x85). Once changed it will check to see if the command byte is less than or equal to 0xD2.

Recam_verify_C2_resp_length proc near
arg_0 = dword ptr 4
arg_4 = dword ptr 8
mov edx, [esp+arg_4]
xor eax, eax
test edx, edx
jz short locret_4022AC
mov eax, dword_4148DC
cmp [esp+arg_0], eax
jnz short loc_4022A6
cmp edx, 30000h
setbe al
retn
loc_4022A6:
cmp edx, 41h
setz al
locret_4022AC:
retn
Recam_verify_C2_resp_length endp

Fig. 27

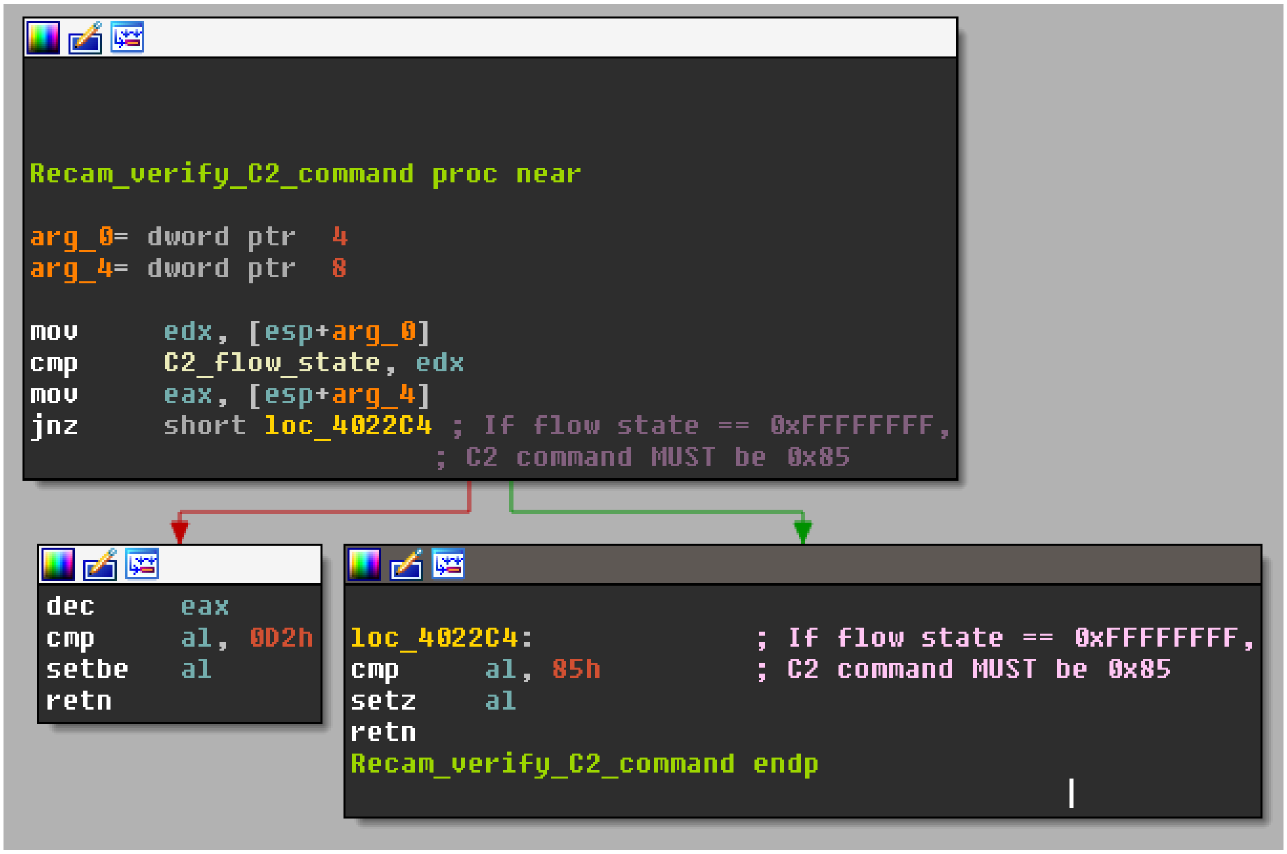


Fig. 28

The response and subsequent data (if any) are relayed to a large jump table that is responsible for checking the command byte and proceeding from there with a particular action as issued by the server.

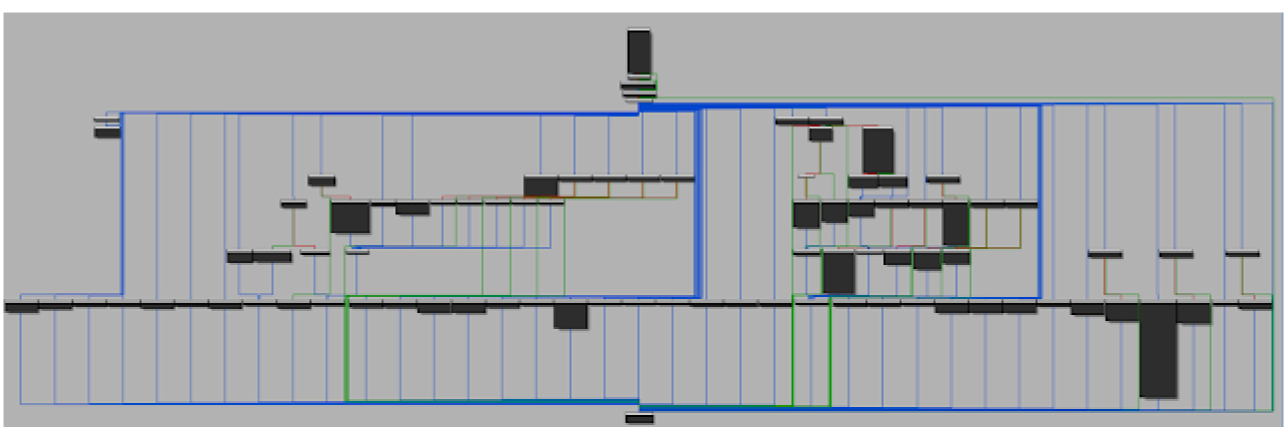


Fig. 29

The beginning of the previously mentioned function and jump table checks flow state again to see if the relevant parameter now equals a previously set state outside of the

0xFFFFFFFF. If this is the case, the data from the last server response is decrypted with the same routine used by the sample to encrypt data before transit. When in the default state, the command byte is passed to a LEA (Load Effective Address) where a calculated address is stored in EAX. In this case, there will be no calculated address due to the zero-extended command byte being referenced by the instruction. Instead, 0x7F gets added to the command byte. The single byte stored in AL taken from the DWORD stored in EAX is compared against 0x51. If equal, it proceeds to the function end and returns with no further action taken. Otherwise, the final byte stored in AL is zero-extended to EAX itself, multiplied by 4 and passed to the jump to determine the next action as requested by the server.

Fig. 30

As one might have gathered from the jump table, there are 82 possible commands that can be accepted from the server. However, not every command is unique. As we can see from the highlighted jump offsets below, many lead to the address shown earlier (RVA 0x227C) that is jumped to when completing no action.

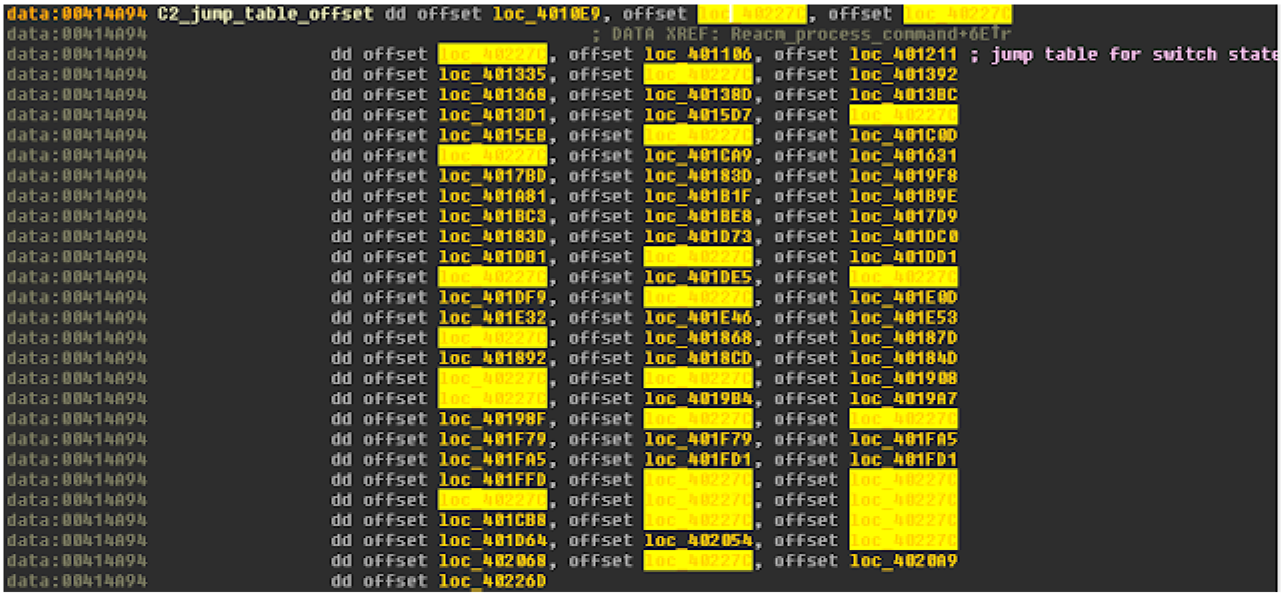


Fig. 31

While time did not allow us to deeply examine each and every path, gathered sandbox pcaps along with our understanding of the command protocol allowed us to examine the commands sent by the server and calculate the jumps ourselves. Here are some examples of the functionality available in this variant of Recam, given the command.

0x85 (case 4) - Process initial server acknowledgement and set flow state
0x81 (case 0) - Keep-alive message
0xBD (case 12) - Download file to %TEMP% OR download file to %TEMP% and create new process
0x87 (case 6) - Create new process from argument
0x89 (case 8) - Close network socket, release mutex, call WSACleanup, & terminate process.

IOC

Malicious Word Document:
C3b1a98c6bc9709f964ded39b288aff66abc5c39b9662fdd28ddfcc178152d67

Dropper binary:
99371d8da86e964cc52bd719fd85f1f0015e4c60a9705747bb9b8ac52fd29b4a

Payload (Recam):
1fd8520246c75702c000f4fac3f209d611c21bfdb81df054c9558d5e002a85ce

Command and Control Server:
185.140.53.212

Domain registered to the Command and Control Server
U811696.nvpn.so

Domain Owner
meridoncharles@yahoo.com

Directories and Files
%AppData%\mozilla firefox\firefox.exe
%AppData%\mozilla firefox\.Identifier
%Temp%\Update.txt

Registry:
Key: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\Update
Value: cmd /c type C:\Users\dex\AppData\Local\Temp\Update.txt | cmd

Similar files:

Dropper related:
006583023242bf4a8dcd0190aef32500dcacfacf1dda7b24409133dfccbfd186
0086bb92bd34b41e180bec90dd15d4b0d0eb9c7384a68b66354d603ad8e14706
01226da791e32f8cc907f88b2b672068b78b86b1a0d154bd22274234a7d9b5e6
031046538b60f9b243aa74bdec2a13ab2aee4b941a136daca12de78c3419dd6e
038aa33ab7363ea484efe9c79c718de500acf766266166b6a70e69b0a67c3984
03fe0caa0f3e21f1975bdceaa5e38e00d725879322c8815e7b3b60b1c0cfec20
13010c16d07c00d22ffebd78c7e00ee40678088f3f46292d3fc7e7b299296361
14d341bd2c2b7ea90b3b15fcb8154caa036b629156199d955047fb081de6669d
237ed26f6d9b19ddd2fc368ba871c1df651bf20e21f0f2cf2f1a0d9a885fdfab
2442515adb2cbc03f54b4edce7762e3b4ce0763ad073666cfec5743ff1bc62e2
264ba90eb6e5e38905bb8333830ea0ef86126d06d68a8609efbe18068d0861e8
26a6f02efcda37e1c0d1a66ce57ccfe0fb95bf1ddf9b20106e0d90407cb89d0c
289ab18d2c67c1c2f92829e11761ba3ec3a9088f36993031a002a0dbe7ab08d5
2a1a0356d96f97a4710511a5e023d43124e7e97261154b5e82d6047fd6dc85e1
2b3fe2be93456173aa1d998764284c687ce68db9c5bfc5175ccf475b090809c1
2fbd846efe0cbf9aa0e51da044403f7d96f159ab655bf44ed697062888fd4fbd
30c0f9250fbe8b8eccbf9b195a4803d33b320d20da4252763cd91b5a8c261872
31442c2ed9a803eba0e162616c16907985487f3bfddcfbc4e30e699fc58847e6
3e7aaa364e5d15d974fab62f828a7ab1f14f5c79ceca7e3dd8560f9a863517ad
3fa46682f660e7f0136544ec995657150bd42dd33ef4362bb0c002d25c9327c2

490183ce7288fe6b35f2ad05514cbdc8d212121c661edf570530980634983276
550db43b5965301a33d2bf71760c4e17c7ae249bcbb1f6b0c9173d76774623c4
61788dbc231ec56f115607e6d7b3cfe2962ecdc112bfbd0a8f2ef1e3f145b25b
6ac23f995572bd1b9caf860384abf62921cf6d572e679872314f67038bc076ab
6b965cc215758d529edaff3a2913226823848c47146121855a482059a6e8ee38
6e74c3a1f5bc17f38da7305068f9203943090067b2dfa67fb9361fa1821a674b
7960bc0d2d9422df90216d1b81fb4420c04607c374289ffe10eb9a80e18990cf
7c127f9f22a346d4369405f540c4e467abac02bcf4589dbd6d3e97f18fbe0f98
7dedf655e19430f538faab5638ec76383d651abf5c24ed799e101abb7c9f40b8
99371d8da86e964cc52bd719fd85f1f0015e4c60a9705747bb9b8ac52fd29b4a
9d0671ee5279abd7973a426c60f44723503fd09d4d771092e2453ac4a3e9334c
9dfee20b6e5c57387968da91ec6efd4e0ac8abce219c437a2cd4b5df36f2a79e
a9b4e5cf351bbb3f20fcdcd7cf0b2c03df788c4d0c0d59a31cbab09cccc335d1
c064c70d887ea1ccf37e3edd19dc717b53f2c0513424416fc783654ab9f61134
c13d2ce1b561ed02c02c9a2899a7d7e9c97400ef8f81eb48d6ba02a90d76c689
c2fef90a5ad97a030712b9245fccffb26f226370bd6f67c989bbf95e02ca03b8
c8d98b7f1cf8811c3e75b0ec8c1011d0ed4c3951420f04e3b743da564869e02b
cdd98208665360ca7745a1e5577e1e9607c49ac8790ffde2edd93e2c64ead8a1
d7d94b3aea52f0a9bda593806d56f962c09f4d389820cd9016ceed73767feb55
df46d34c0ff71d994f157cfe158ae7dba9dbf16ffb021a10e401bc9afb2f4e8d
e20c3b8e3460f719219cb4ddac61231db047c125818a391d34d87dab1726f070
e3704aa7bfb90e8e7f96aa529cbd96319696157cda076d92bab9eae07cfe977e
e8a4a890d34b328857389d48d837fcf4b9ab4774a29bcc582b4cdf05ac3a49db
f7ac275c2971820fb311f41049c5e460251486625696caa0fc60d0ffc683fb70
fc0bd596957f70e87954ef42acc0bfe0123c11be6879c7c910a4a7e434e36c46

RTF Document related:
27bf1851e64f5e6d6e33b2b3bc89b82dda2da2fd9a747c847c148909dda028d3
7a63150ebd09fd4f8c8f1b7485a0139108f22aea043c9759239d9976948b5c75
a27ebd73ac836d659512befe4fee57f9343eca992448242b1ab40730506a747c
ac2e9c83c7e23514c196b14e49f93345bfed2ed8fc10343c21d358ec3e809928
6f3a8557e2c95a717cb4808004229304501 1e74f2a4c79aaeffbfcc86456eb37
5a234b3e389a22b70da242b1c93d65f358d60f1347b03101f7613fa1db032645
660a6197a032cd2baf35df713575f16b71ebdcda67b89deb1afc45f2b986490a
7c7d00e12273fe924a9dc945add40c91692b8e468682240cf241a418d3ce198f
1d604531fd15a7476296612cbfcfa4761148a61d54f44351d97b5feb1bb293e4
46907300ed93309f083181489bf68ef8f357b1a583bda501b3ab5249f55778ad
6fed760b5eefd70dddef34daaf678f3d999a91c0f638c8afbe844dd8097c4bc8
f231bca66d0d8b241dfbd49630602405297c18cbe108b609f420ff6e9c987e7e
d3de3093b173746fd5cf892e32caafabb898616766953ccad2b6a303930731d4
662f7b3816dd59c5f1988c9d7b5b62b61b94028fa513ddfe0c00e21a79367883
b9d38a08e7b888940ad0bafb6032dfe45d4b6cab2f09a734f46d05d718eda57e
1fd682146e278a76b216215ba9bc8040f9fc4e817f0676ccbfd46ccc13ca95d4

135db94b1b75fbeccf2e28abf568e3384e14dc452e5f638cdb1cb533794d2bca
51e1da451e38159e73bf185f627fedf63aca03734e95f88dfbf85536eb6bef9a
18ba9d4175c48148d4dfa7d022a687e4c63da18337fd86a3f48efd5afa5cfcde
4f6dd728633bba78841da6fce1ce9a3e8673830588bd51e2bfd283e2afcb0ecb
6943484dad1a3246a82b1a9444811e0f4c561770291dd936a1f1a5314e13fda9
53c325566fc88c8355478e001445da54debb009c60581e2e6c3345f10c4f65f0
a546965c6a46e9f19c374b1195ba0924c0a6d462a9d37210527d1f2801719932
2752ed543f880807557ba9556838e4ef6a1582ea50351fc2723e4d63f4f57ad9
771f02d96d7eb5b4e8b719e3d9848482220a27ca0704c0a156c613348e3a64bd
8ecb075bba3f583d350e136b9df81d2a773f26174a014c59f26915eca785dc4a
57e31a5ea3b7fa86cf8b1b257cf91e5e499e633b17d2a9bff388beba2fd101d7
41015210e0faac0825b1c51dde385d390bc7cba7b4c45a534688d58cea5d0d58
2c69630f2fc87ff69cd7644e312b484c118b7d447985c655ba772954085c8f33
8481f95d09058d24f4a764223c128bd38e430aa9211f135e684744cc3bbc1b6c
42b1093ebfd76a869f8a50e7529348f6a5b9d8b85c0d6b2b90d243a32e35baea
3ee302f650b87c151bb6e91d2b3ff38ab49b1e4ad30f6f18224d8236d9784ad7
2a8410b31c50f966704f0abb5a976f0faf59604116c4a4908992b33bb9dc4e19
05ce4c14144b9935de8d30dee60dfaa513bebdccb35099c8892ece10c86f0a6d
5885ea86add67e7a149634e8c9ae4da63e35b431c5210256a570b56387ee1f8a
5daba9df4ce8d4e7bc3e5bb6dc43dac7034185a218e53a261f814d6398eabadd
e5666e76d755a8c1e0430860677dbb1991c9025b99318e42dabefc09f276b91e
5df14265d5ce37ae0dc0dcb332c42b7331cdc305fc1b265b767bb96d6e6d93cf
180734591883412d5ad4db3f768b59e7e918fcc987915fb130e72077a27f15a0
aa3f5e91fb4f447bfff93123c404a62b8ebe2a9790ebbd02c57f5ab61d2d2882
70ebcc2aa157230051490f5480d49dcef22ad8c26be1307ad8eab63bd4233c40
a2139efcf459ad80e6ffe1e3e346d18c466dd4a0c1e9973771e3c674abd56456
98b4862bba7bda1169bc8a3f4726c0033b0a02f9d4d934265c56307c8a9cae11
944f1bd37c0b223d45b1f33f1b5cb77bee2eb720ac2d17b1098ad770d85083a5
aed64804a5191a6c119cba1f81ad5f09260bac0f8642d1f0101da73e8a7b3329
e338747a4f62d1826ee92cf74ae3161bb817a19eefec7902f4f334c43bf94399
888f9c76d26cdfa7340779a015d95785253cf14a1a850fe79ec64a6a24b1deac
26092c74ff3ad634577f5b890fbdc911b8813b337b77aa5db5132b5d3a51990d
810afd3bee3016d443536163cffdb7819cdabbf69b16fffc9108f9747d6314a8
5062a698879870e73250973e45cadcaade05a314f4f17fb17f14f82007bd6b1b
f4ecc8fc573401cbd54022d91fe1fe4575d16fe899aa7e30fcc7c17cd8b66ade
5f485d6c9b210bdb7a7dc4987a3bef3c180cff3a389229853987818673032823
9cb7191e87fc0efe04f38e8881dd09367c267a5abf0fac51f54d89b6fe1c74e3
2da4433c077bc5b7a9bb74e89b77e2f441e6f3faca86f1b33ec62745f9d1c119
31e1bf4c259b09f9a0af1d25e1978fb1cb0aff4dd3ed19c40654b2cf0fc7baa7
b05d350a806200a7d98386b0955e778f5cde98d1c0b29ae94245731b811983c3
678fa6d2731f25e7529ead56c1274b40a15275fa8df95004e3c8fa8fb4f87985
3d4324f0e9eb827b180b8c406c7893a57524cb9c943d0cf51851277c55fa137b

ff9403de6d602fdfbce67c39929425a6ddf1702f8ccb5f73a45590f41ec9b298
75ded8ff160d4a4cb1a15617ba82c2fd979f4c6de068ad3c1628c3fa96ccb06e
c167a67bf9f07204a6d4f15848d40c152138d56d1e105cf851dda74049ef4e02
d873d716210c4c3eb86d2e27c72ec5b083b29a3ceaa0e71e0a784aac74667670
9299804183d38a89f7790527b21acbc958c40dd09b09c40fe19c4629cdeaacf0
b9b5adb3b3918e9652b73d06e2ea98370a48c3a58ef86405f56c47771cc21af4
f982fb8ef63466c16cdc5c8d107770f986e4b28c476edf5d848a600e2f4320fa
202e364f3b50fd785831515c6991f5db2c03923890785788c15c04e8128b4e12
4789ee4e6c8490b31fa68b40dc134d29baae0872dd1fda04a28d62f8455fcf2e
167594296c9be12c0aa8f9ab40e83409361ba0e4eb8044fe3276d0c927fcd3c7
90d293195bf573c14d39eac6aa6e0786b3282bbd3f4bff850b7dedb7eebb6049
ecee734d28bf51f199efea4d27e42865fdafe0d880ba6b9ed846a502050d8e20
7cdcb22b02294c4d981c161e6f527b9a4b9e0ec8a63778f0f9ba4b0d86daac2a
84684f264b035865a26dca9631f0bb0d88597daaed3cfe1bb69c6c81892cb20f
b89ebe23b4d2d47885c7431ddb1ffd084e1537c59585fb29a825a06581af0dea
bbe8d4f57f896b96cb2f1c147afdf4bd50277d436bb68c03216547bdc9d88b7a
603c5d3dffc0ecfb898bbab51130b7b1c95c95c0dba8691873eed5a21ef8a052
2f8141c3517394dfd818bd9ae762332a9454c8ae12b815de7eb73851a47d9b83
17db38e774338602058f883997841aa4ca2e647b605768b9a5dbdd89ac252d5e
e83f5d93e97625407ed4529e05f6e6116cbd183177e3ba65f2acd8e81f07bf36
bf38e39eb1dc78c4b854546f8e212527aeb8990e7fff3d63f49b8c15846d19b0
3823a6914e7fa21764df33c51efb307a53385bc4692d167af5220e7ec7425235
c8e01564f14c5677db7c0e264cd2b125952edee26731b42d280d476b3c191a56
88a58c8848ab67c3552acac7250fb8408c06c398e6dd79df8437e064b209023b
9068cb2c0257b905d8079bcb40033fc58e832e7c0d2d687a1d942542443634a0
ca2ce6f6cf845e5b17934c3a5a0f43ef860fb5ed67ec771f293baf4d9938cc50
3963c2ebbcbb893f2695e90249dcb6ebc546a0fbd0ba126d398d9774071262b1
ca270cae900efb83979e0aa2f997876aaab68714c242ab131b5a0929ca661ea8
1f262c5c54bbb0f6622fb8b5ca806d8544cac3fe6184d300c8f28074e180b9f7
9923f18495de93f7c957fe991f0dd3b185c8c7660b9ef93f9a83ebd4bc31cc1a
d451a59c8900f7e9fbd535d1a3bba645d2f7bce4bf95df73fbca72755602325b
147fd5fb0f771d22f38f33d217be393358c3e3b6452b3064922acdb1f160e164
ce0e5fb3022a01bb6ff24d9430e599eb50cf670e6f3d6c3e96adb448774e510f
eb8973d6ad89e13c5f87e14c5cfc897d5d44436891dc9ed651388334041a5f31
47883497485f0f841395071590f9bef0e13a6d2859a511002cb2a3060c128c02
e6bf577e20cf0834f2b7ab050f81e475ad41cd1580f53dd040e919b58bf57ccb
33bd585673e1c40499e7702bdf04d6326ef440da297611301c190ac508c107f1
2433294309a1c514ce9733b0ee8b0ad4ce807d4f93c0df17a2370168577e10c4
b710e3f9bf878c2c7940bba099ae0fc8363a5446f6c9a2428f5fb6fef39cd814
449f09003bc890df9bf5cc8885eb1e278a724eb3872bbf95fbabb504aacc9ca4
77e77ebd338d1829ac00047f4c64fe84443699290f67ac0ebe01697b38b6d439
ae8404e4404174a6fab8477b27e60d49a19b72ee623ef60e37b391812682944c

ff68a536f40f20c0bca80d30077949a9b69f37b5bbc4bf32eeb0e36803a1cce7
723b22279a368ab7e1db69352ecd4e4524f3d062a30a3dbeb4101d3ed681002f
a40f324c032f9af3a0a26be7d21655f75381058bdbd7ec49f6cf9fa6ec01074d

Payload related:
08433debdfd2e520ac5d2f0ec922f84020e8f4beb711fff868e607f06f40e133
0a6be4d49ec1c9fa2e4d0f097d3808802117836effe1f4087cb9743a88695bc0
104024e27738377f05a24aa493561d9ebc9d38d462fedd0d888830cbaae670e0
155033e85069c716c3819c5e81fa876ad0c4269d911f3832c4cc61dd45bc2aa7
15a6a46e678ecf4d5d9a9d9d75eb12aa33d55611ed8cb555079315e4b9e22553
16e6b6bd449f4ffb634df45808adaf78456c91b53bee79b8b8b7bff1ed3d4b50
191a3d8104ad83ac76cb36014021f2c7ab81b74ae15eae59d799bdc0f4798951
1f3a5878104747d51c08d75ef0c7eb644859ea01e46246d97c278292d4251960
20208b337e4f0a418a0e3329f7f863700538bed1a50ca720db05e7f0f9ad8c4e
2258fd874969f13c8b6b62db03c5db12df9848bceb2e0fc6654690b691956889
22e249f2d2bef1632aa61d6f07909eeeda706670e5d89f11aa8c3bb89bc92f16
234add1226f0ce30f83f72afa1067329ac22b20759c5333c404926775f298119
281729d8c1131999a6ac0f784a5f0ba45ca83149406bcbbf06f866f446d88ce2
2a917a89e324402b23e084820fcfec8c3e98e4a1124be1f29323e1694b5755d6
2eebd42599746cf252f08338f81ed3df9e975342c12e13ca8c70fafa4c4dc83b
2f4a6936dafd89cef6b8ba828e90439daa81a2a898c2da0767bec6b51cdaf11a
2fb2b9863d534f9436e438f86275ea1e0c4741fb8fa5d9aa7311cc025f78f30a
300f8301f23487e2433002999f1acbc804f4f684f6403e3a4c65f4e5979540e5
304b15d20b64622250cf0d0954f3c887a059c4c98642b3fd6b3e3eb94b07546a
344d063a2ce0b614cd073725b9b9a018c8706b6b9f380800506eadd80e92bd7a
3738bbfd8ab9a37a5ad5bff41f4fabfc3a7ad0f4085a5290b83c1fc7ee3723b4
37907c564942af4f1b235ec8592b60f7286de2e67206506733955c8a70ae5d7b
39675f4f6648317e322593cb654094e548bf2735df063537ee896037e946d451
396f61528c9b25144b89cb20e5668daa965489311bb8a00b7a6244feb0b79190
3cc77f145194653036c846505fd26563f3a1fac0cc2d3fd5123701685e84b28e
3cf115056782e777dbfa2b6393ab1463547401c8bb2d59ab89455bf77305ea98
3d825d79bda177aadb305d8c6822bb97f7fe2ee451e5257e85bad42e99535606
40e90e30cee9c1726154b6678221fead406b94bc3259ba8386ab438cd5ebb250
41475ce4186414fbbb8589f29d5f7001bb286167e77324ce4da719acb21f4dea
42a4cd926419aa35a4b17264d190256b9cb52cdfc33c943e68e578566ceae334
43024e16cbd0ef8ef160355e051477a3b31018c9dbe0afb8cac06576f57deca9
43b361398583c3c0ce75429a84b30ff10e047e757223eb433414a99d35160a0d
45c375bfea191f576e2a5deaaacc8e695d2d7868ae5b87859411dc5438240212
4976a6ad02ff39f568cb9cd7a1ebdea8c381b8f9600909c273f7f53436dd0f05
4a1d7a0688318d216c3b3c9883b7e35c4ff2bcc9cd7ecf63486e622339bf5059
4c959c764c3bc7d5141cd002c261382b95db1d22c371071409c08d771bb5e413
4cfac44cf32a5c067383d815997f7d474af97bb82ebbfa25d909f86a1081eabd
51b4cdf23ea83bf87d741e92522c6f6670b98cc0b2b13bb41b7cb24afba959b7

53546e823073cbda154c74b6e511556cb17db067f2a2e746693c2d1c7f1f7eda
556f0b284c63cc884661c583726ddc5bdec2e679f53e2b3bb52233b1750d78aa
5934d3620e3d2905cba1cc431e8040f00ca0b763c8a38a6fbb1121e9c4c25cf6
593516e03212b03296720bd0a1f09221e156c19067d713b723fe57a81d3d5faf
5c9154974d5daaf850614fc660e1e9864f9df8f4d7e0ce68ca3e00b3e15d4e58
5ce46c27795fb25bea268f233f89be52336e344737e059bb21f3daa9815eda96
5cf4effade63cf1aa9a3cc6909bcf2e16867ce8db60e7f20a9d08dd4a5c59bfc
5deeb0fd202427f5763562bc80672096884655b834e219d527ab200f2ba5f01b
62387f4d55f15ea72c8d7f5d2ab1bd4029d895fe3f22d95980291b2595a658e6
63c0153b2deff5a9286db1bbf41c327d55e4afeb938472b04f5f2e5367a1418f
64f2e9c68908a7ce978ddeb17b661e787f1464ae706eb1ff1ce006cafe81e07e
6c8eadfcecafdebccc737420d83c1f8493d12fcbecf13198aff88c10017316fc
713a536264bff75649da55d367804c47333c55ed326be5ff73384fc3f2e87977
72adc4612c53415a4b924acff7f8f400d1f20f5eb7d02f5e2d12488db168fd9d
73afba6db12fda23caf12b2e71310631abbe6b9467636d8343ddb3674a49f527
771000d6307e4372f2f51b6958e8b7e7a35ee34e9f9fb636b77559534a9fc1ce
789481741e0df019d839bf61181b5cdd085b2334123be58a4b64c12fa7bd6193
7e53b73cca7189e97e46de3e6e056edcb98b1aec6c2c338e043b8732ae46580e
806f0d040303841e913e0a2a3ac96ce4427c269dafe6f135b1b64f5d9ba3d740
8263655df987bd6c2bf8087e6b8759a360482757cd28a1d7e1742f652fb6bfc5
8321a2104d185e9ddeb7f19c15bfc92ec4fe7fba67ba5a167270706fdb90e6bb
85007ecdc6baec70a7e259395066c96fb7ddafd1be8a84244c92a7dc52a003e4
87ae17f03b47f54fd9f2a3cf9f3e1e4d5cdfa6592bec4a05214c30756599a3ef
8af2013a53c58346f9d558155691d9d736b253a6b179a40fc9e167cfee025dba
8b42ef0774a9eb367ad36d51e990ca5bcc2708c33380dbaeeeffb29e113656b6
8c9c9a4689a4b0319a32f32ded1e97dce3185d1d91fb3338b38081d28ccfb80c
8f256697f2c7c5c19671c5ffec1b7458a609efbfa3c54b0aa83d71c524afb3b4
903b5ca6aace73bc14ea9a22c900772078f063cccd9c144de6236513184d07b8
9172fef013aca4eeef4bc4abfe0a283f89acaa58c6fb61a654eb66fe60bfb42b
942d5bebedebee78c19d4dfce09c2a8e69c94ed77d55e7f16db3e0dd8f179f9b
94948ea1285abe593df349b24340cf176726e92b0e20b35a3d9668de43b856dc
94ef90e1cf5286566afb6ff7204b2045f5cc44d895f4e1ed3691b61ffd961bfa
9510899b01066f15d622a79611c3b1f3444559ccc483aad04f52ef3ed0c0e955
9751f750493d38ee62c7139cbc4ee4191ca6c7f9a6ba629468f33d8c653cf660
989a36bca1548dc37bbe8c9716e69d789cc1eece69dc0b3c1ffccbf01764a0e5
9b7442f32111a4acff0f85a4c38b79dd756f2dae3b73545ee53a4c000aba8b1f
9e56b2771f3bfc3003183d0cda39821c2d1d8f427ea521ba0cbce0bf69734005
a07c4ced1cc47b8df61eb5d4be42ea64a79d47ef6dd4967e9c956dc49796ba80
a0dacb034d3b1e563714ad91fe51a71fe83c488a5931c1fb82f8cad04ae2fc1d
a54ab03086af7b96b88efa0fa1abad56870947ebca89c05eb6503eee5f9a4cf8
a6693a690c7b1f1363d61301db838c0854f7f95f5322ad38353f9888dd386346
b25b113c1159e0fd35eb6b07a4f2318439e779ce9916af6b8e909e1eca744809

b8c92c94113509e2d60ee97c0233d107ccd4dd11ce1bbdb2eb5b92113b4d27cd
bcf5b70de961cf53dcb845ad766ed8d36653408e72478f3c4f36883018cf4264
bee0a43ffaeda2d39e0bb754ab6ac5f3bf159f8bf79d5c3619086eb26974aac9
bf9ff9889bcbd338586a17ea8a7f270efd45efeea367f118cf2c036b9a63c45f
c0d71564ae30d7336504ae8bac556eca7338f519586060edb378250f2af46ca2
c2e56d8c1090398b2d60370b55cb5bd3359630451acdb554ee8c1aa7f5c1b51c
c5d2fb553d6406100c0b4a1be2f9e11ac2101bfa081527e211b7ac113fff6ff5
c63281c74f8ca4309b1651a18241a69979fb135048beac2267620efc46e798e3
cbfc83e8341728851650b3bb78855e5e16e030e33fc64dcbb8e27004dce7b290
ce0c460160d5a2000e807f225940fbcd8e0e629f3bbec52e952c3355ffa688ff
d27406627260481d000551236d65bfb284ada771f105adf41ee231041fe5b2fe
d8a8f29bcea4e4b3a2515b31926abd1ce7b2adb0679a26a68df992c2c14ec730
d9a3fc467e1198db9eec0a2231396f94d5fc44206d829ddf74dbeb529cb6e6f6
d9ea0bc4909f894c30368ab6fcce37046523945bc64ab46b671491bea1d48904
dab2045ebb2a37e78f0c3a0fff063b32ca66e16acecb197e33e671948f9ba4fc
db5ea1d29e6681c30c49f640020fac8aab39e696e2b871ce4cdae45bacc1da0e
dbba4d5a7f81feff29265babf103cf6c5e240254a6dc157ddc4bc3cfa2941b75
dbdddbe487c70319c928d8b4a9a499dbe18c347a622ef78e665b27d9563340c5
ddaf881c973e8302146ca59f0d65327ea0fc4b9bd30961e245fea693f377e893
de1edc5dacc285b7f4910f4252a01c3d418b5f9fe14b76c33b00e8eaaef9976c
dee635d66b0ecd41655f180a9e51967a84c6c65be8574197da161f3cf51662b7
e242f942726ebd6c123a6d868d6a5c8c420e15056f0662773c14f218b20c550d
eacf6fbc19a4a8d39ba16be72a51d0f376d59ab09b6726e0632eabd00382dba2
f013c00f93260c72da674522b0169a65553b67c71ccf2bdc1879c7c91381332e
f070a3e1450d56266a24d0f8f0878d249ec777305f85b4b4b55e1f1aa474c5c0
f0e38d842f1ef4d67c7510440f7c8985cd79b6c52bbaf07c43aa370e178a10dd
f14fb553d340d6f62c640711e058f147d186a9f51b0f422bd55187dc3c34ca8c
f2b47d7b9becbcdaea350fc0fb01f258e5e31ebc3a39b8e9e170730b194869d2
f47254880625736325ba64853ddaf35a5fdf940843fb8af7ea66008953539170
f62cfb5f412ce47c611430e4445c14376791eb68ed665ab8641731b923643b14
f63c4ece8caee488bb4873d7a494f8d52b0bcfa8746794fb5e0b913808b3f1e6
fd80d1c741ca0f648f7ea14bb8f345358681f5335d909f4f0e56f14640bde1df

Conclusion

Malware is a moving target, it is constantly evolving in an arms race between the malware authors and the security researchers. This analysis shows the level of sophistication employed by threat actors in order to attempt to escape detection.

Obfuscation is an art form. Techniques can range from frequently changed packers to the multiple techniques employed in malware such as this. Often malware packers are modified by their authors very soon after deobfuscation tools or reports are publically released. In many cases it is enough for them to change minor parts of the obfuscator to confuse the deobfuscation tools. Hence, malware researchers can't rely on these tools and must resort to be able to manually deobfuscate code when necessary.

Understanding the steps that threat actors will go to to hide from detection and analysis is vital when it comes to protecting systems from malware. It is by applying lessons learnt from analyses such as this, that we are able to detect advanced malware with tools such as Advanced Malware Protection (AMP) and Threatgrid.

Coverage

Additional ways our customers can detect and block this threat are listed below.

PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

CWS or WSA web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such asNGFW,NGIPS, andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.