# CruLoader: Zero2Auto

k malwarebookreports.com/cruloader-zero2auto/

muzi

July 8, 2022

Taking a break from my normal blog posts to complete the practical analysis from the Zero2Automated course from Vitali Kremez and Daniel Bunce.

### **Assignment Background**

Hi there,

During an ongoing investigation, one of our IR team members managed to locate an unknown sample on an infected machine belonging to one of our clients. We cannot pass that sample onto you currently as we are still analyzing it to determine what data was exfiltrated. However, one of our backend analysts developed a YARA rule based on the malware packer, and we were able to locate a similar binary that seemed to be an earlier version of the sample we're dealing with. Would you be able to take a look at it? We're all hands on deck here, dealing with this situation, and so we are unable to take a look at it ourselves.

We're not too sure how much the binary has changed, though developing some automation tools might be a good idea, in case the threat actors behind it start utilizing something like Cutwail to push their samples.

### Stage 1 Exe

```
Filename: main_bin.exe
MD5: a84e1256111e4e235250a8e3bb11f903
SHA1: 1b76e5a645a0df61bb4569d54bd1183ab451c95e
SHA256: a0ac02a1e6c908b90173e86c3e321f2bab082ed45236503a21eb7d984de10611
```

Stepping into the main function, several obfuscated strings are moved into registers and a function call is made immediately afterwards. Next, LoadLibrary and GetProcAddress are called, indicating that these obfuscated strings are almost certainly obfuscated libraries to import.

E8 53150000	call main_bin.10825B0	
83C4 0C	add esp,C	
в9 <u>94480901</u>	<pre>mov ecx,main_bin.1094894</pre>	1094894:".5ea5/QPY4//"
E8 96020000	<pre>call <main_bin.decrypt_string></main_bin.decrypt_string></pre>	
B9 <u>A4480901</u>	mov ecx,main_bin.10948A4	10948A4: "pe51g5Ceb35ffn"
E8 8C020000	call <main_bin.decrypt_string></main_bin.decrypt_string>	
8B35 04E00801	<pre>mov esi,dword ptr ds:[&lt;&amp;LoadLibraryA&gt;]</pre>	
68 <u>94480901</u>	push main_bin.1094894	1094894:".5ea5/QPY4//"
FFD6	call esi	
68 <u>A4480901</u>	push main_bin.10948A4	10948A4:"pe51g5Ceb35ffn"
50	push eax	
FF15 08E00801	<pre>call dword ptr ds:[&lt;&amp;GetProcAddress&gt;]</pre>	
8D8D A8FBFFFF	<pre>lea ecx,dword ptr ss:[ebp-458]</pre>	

Figure 1: Obfuscated Library Imports

### String Decryption/Decoding

Stepping into the function called right after moving the obfuscated strings, a few things stand out that indicate the purpose of it:

- The long string of characters that appear to be an extended/custom alphabet
- The add edx, D instruction (ROT-13 anyone?)



Figure 2: ROT-13 Decryption/Deobfuscation Routine

After returning from the decryption function, kernel32.dll is decoded. Now that the decryption/decode function has been identified, a string decoder can be written. String Decrypter/Decoder (and unpacker) can be found on my GitHub.

2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: I9egh1/n//b3 --> Decrypted: VirtualAlloc 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: t5gG8e514pbag5kg --> Decrypted: GetThreadContext 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: yb3.E5fbhe35 --> Decrypted: LockResource 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: pe51g5Ceb35ffn --> Decrypted: CreateProcessA 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: F5gG8e514pbag5kg --> Decrypted: SetThreadContext 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: yb14E5fbhe35 --> Decrypted: LoadResource 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: E514Ceb35ffz5=bel --> Decrypted: ReadProcessMemory 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: s9a4E5fbhe35n --> Decrypted: FindResourceA 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: Je9g5Ceb35ffz5=bel --> Decrypted: WriteProcessMemory 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: I9egh1/n//b3rk --> Decrypted: VirtualAllocEx 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: F9m5b6E5fbhe35 --> Decrypted: SizeofResource 2022-05-30 22:06:05,423 - CruLoader Unpacker - CRITICAL [\*] Encrypted: .5ea5/QPY4// --> Decrypted: kernel32.dll 2022-05-30 22:06:05,424 - CruLoader Unpacker - CRITICAL [\*] Encrypted: E5fh=568e514 --> Decrypted: ResumeThread

Now that the strings are decrypted, some additional functionality becomes apparent: packed code located in a resource and process injection (process hollowing). The executable being analyzed looks to be a crypter that will unpack the code contained in the RCDATA resource.

#### **RC4 Decrypt Resource**

Loading the executable into PE Studio, the RCDATA resource stands out – it is large in size at 87068 bytes and the 7.98 entropy indicates this is most likely encrypted data. In addition, the decrypted/deobfuscated strings above lend credibility to this theory, as the following libraries are used to access the resource:

- FindResourceA
- LoadResource
- SizeOfResource
- LockResource

elo													
C 8 T													
ween/unue/vdesktop/unein_bin_	type (1)	name	file-offset.	signat.	non-stan	size (8706	file-ratio (5-	md5	entr	langua	first-bytes-hes	first-bytes-test	
indicators (4/25)	errista	101	0.00034	unimo		STDEA	51.53.96	222220142314058243	7.998	Inclub	00 DD 0C 92 00 22 00 0	in the second	
visuatotal (53/71)													
dos-header (64 bates)													
dos-stub (192 bytes)													
file-header (kan 2020)													
optional-header (file-checksue													
frectories (6)													
actions (99,39%)													
Proving Joseph 1													
monte (13/67)													
riporte (najvo)													
ports (pp)													
Calcolog (1/4)													
COLORS AND COLORS													
rings (15/2095)													
(beg (time-stang)													
anifest (n/ii)													
rsion (n/n)													
rtificate (n/k)													
wrlay (ryla)													
-													

Figure 3: Resource (RCDATA)

Following in the debugger, the RCDATA resource is loaded, then, starting at position 0x1C (28 decimal), the ciphertext is copied into an allocated buffer.

		-
6A 0A	push A	
6A 65	push 65	
6A 00	push 0	
	move dword ntn cc: John 1101 opy	
	all asi	FindBosouncoA
88E0	mov esi eav	FINARESOUTCEA
56	nuch aci	
50 00	push 0	
5A 00	coll odi	L and D and a second
		LoadResource
50	push est	
OA UU	push 0	
8BF8	mov edi,eax	
FFD3	call ebx	SizeOfResource
33C9	xor ecx,ecx	
83C0 1C	add eax, 1C	
0F92C1	setb cl	
F7D9	neg ecx	
0BC8	or ecx,eax	
51	push ecx	
E8 24240000	call main_bin.FC38F4	
83c4 04	add esp,4	
57	push edi	
FF95 F0FEFFFF	call dword ptr ss:[ebp-110]	LockResource
8BD8	mov ebx.eax	
899D ECFEFFFF	mov dword ptr ss: [ebp-114].ebx	
8B4B 08	mov ecx.dword ptr_ds:[ebx+8]	
8D3C89	lea edi.dword ptr ds:[ecx+ecx*4]	
B9 5049ED00	mov ecx, main bin ED4950	ED4950. "VirtualAlloc"
0355	add edi edi	rb4550. Vir cuarArroc
80BD EREFEEE	mov dword ntr ss: [ebn_118] edi	
	call emain hin Decrypt Strings	
68 2040ED00	push main hin ED/020	504020, "komo122, d11"
CE15 0450500	all dward atta day [20 and ihannuka]	FD4920: Kernet52.dtt
68 50405000	call dword ptr us:[<@LoadLibraryA>]	554050 - William - 1411 W
08 <u>5049FD00</u>	push main_bin.FD4950	FD4950: "VirtualAlloc"
50	push eax	
FF15 USEUFCUU	<b>call</b> dword ptr ds:[<&GetProcAddress>]	
6A 04	push 4	
68 00100000	push 1000	
57	push edi	
6A 00	push 0	
FFD0	call eax	VirtualAlloc
8985 FOFEFFFF	<pre>mov dword ptr ss:[ebp-110],eax</pre>	
57	push edi	
8D4B 1C	<pre>lea ecx,dword ptr ds:[ebx+1C]</pre>	CT (Encrypted Resource)
51	push ecx	
50	push eax	
E8 82180000	call main_bin.FC2DB0	Copy CT to Buffer
C0_03010000	1 103	

Figure 4: Resource Loaded and Copied into Allocated Memory

Next, the ciphertext decrypted using RC4. The key starts at offset 0xC (12) of the resource and is 16 bytes long.

000000000000000000000000000000000000	
<pre>     OverLists</pre>	
00Fc1558         3D 00010000         cmp eax,100         Cmp Len S-Box (255 Decimal)           00Fc1559         7 C F1         Man.bit.Fc1559         Cmp Len S-Box (255 Decimal)           00Fc1567         8BB0 ECFEFFF         xor esi, esi         Cmp Len S-Box (255 Decimal)           00Fc1570         8B9888888         mov edi, dword ptr ds:[eax+eax], ax         Cmp Len S-Box (255 Decimal)           00Fc1577         8B9888888         mov edi, dword ptr ds:[eax+eax], ax         Cmp Len S-Box (255 Decimal)           00Fc1576         8Secf         mov edi, dword ptr ds:[eax+eax], ax         Cmp Len S-Box (255 Decimal)           00Fc1577         8B9888888         mov bl, sytte ptr ds:[ebp-los]         Cmp Len S-Box (255 Decimal)           00Fc1576         8Bc6         mov eax, esi         Cmp Len S-Box (255 Decimal)           00Fc1588         2BcA         sub eax, ecx         Strength           00Fc1588         2BcA         Sub eax, ecx         Strength           00Fc1580         00Fc1588         2BcA         Sub eax, ecx         Strength           00Fc1580         00Fc1580         00Fc1580         00Fc1580         Strength         Nov bl, strength         Strength           00Fc1580         00Fc1580         8BfE F0010000         mov bl, strengt         Strength         Strength </th <th></th>	
• 00F:1550       • 7c F1       11 main bits FC1550       cbp 114         • 00F:1567       38B0 ECFFFFF       mov edi, dword ptr ds:[eax-eax], ax       mov edi, dword ptr ds:[eax-eax], ax         • 00F:1570       8A9235 F8FEFFFF       mov edi, dword ptr ds:[eax-eax], ax       mov edi, dword ptr ds:[eax-eax], ax         • 00F:1570       8A9235 F8FEFFFF       mov eax, esi       mov eax, esi       mov eax, esi         • 00F:1576       F7E6       mov eax, esi       mov eax, esi       mov eax, esi         • 00F:1580       CLE 04       shl ecx, edx       she eax, ecx         • 00F:1588       2BC1       sub eax, ecx       mov eax, byte ptr ds:[eax-edi+C]         • 00F:1582       CLE 04       shl ecx, edx       sub eax, byte ptr ds:[eax-edi+C]         • 00F:1582       CLE 04       shl ecx, edx       sub eax, byte ptr ds:[eax-edi+C]         • 00F:1582       CLE 04       shl ecx, edx       sub eax, byte ptr ds:[eax-edi+C]         • 00F:1582       CLE 04       sub eax, byte ptr ds:[eax-edi+C]       Key         • 00F:1582       02F8       add dh, al       movz eax, byte ptr ds:[eax-edi+C]       add al, al         • 00F:1584       02F8       add dh, al       mov di, dword ptr ss:[ebp-108]       scramble S-Box (PSGA/S-Box Permutation         • 00F:1580       8819       <	
000000000000000000000000000000000000	
ORC:158:         BBBD ECFEFFF         mov edi, dword ptr ss: [ebp+14]         mov edi, esi           ODFC:156:         66:00:1F:8400 0000000         nop word ptr ds:[eax+eax], ax         nov bi, pow eax, 8588589           ODFC:157:         B8 8988888         mov eax, 8588589         nov eax, 8588589           ODFC:157:         B8 8988888         mov eax, 8588589           ODFC:157:         B8 8988888         mov eax, 858           ODFC:158:         BECA         mov ecx, edx           ODFC:158:         CIE1 04         sh ecx, 4           ODFC:158:         CIE1 04         sh ecx, 4           ODFC:158:         CIE1 04         sh ecx, 4           ODFC:158:         CIE1 04         sh ecx, edx           ODFC:158:         CIE1 04         sh ecx, edx           ODFC:158:         CIE1 04         sh ecx, edx           ODFC:158:         BECA         sub ecx, edx           ODFC:158:         BEFEFFFF         lea ecx, dowrd ptr ss: [ebp-108]           ODFC:158:         BEFEFFF         mov by te ptr ds:[ecx]           ODFC:158:         BEFEFFF         mov dx, eax, bht           ODFC:158:         BEFEFFFF         mov dx, eax, bht           ODFC:158:         SEFEFFFF         mov bi, pte ptr ss:[ebp-108]	
<ul> <li>00FC1565</li> <li>00FC1565</li> <li>00FC1570</li> <li>8A9C35 F8FEFFFF</li> <li>00FC1577</li> <li>8A9C35 F8FEFFFF</li> <li>00FC1577</li> <li>8A9C35 F8FEFFFF</li> <li>00FC1577</li> <li>8A9C35 F8FEFFFF</li> <li>00FC1577</li> <li>8A9C35 F8FEFFFF</li> <li>00FC1578</li> <li>00FC1578</li> <li>00FC1578</li> <li>00FC1578</li> <li>00FC1580</li> <li>C1EA 03</li> <li>shr edx,3</li> <li>shr edx,3</li> <li>shr edx,3</li> <li>shr edx,3</li> <li>shr edx,3</li> <li>shr edx,4</li> <li>shr edx,3</li> <li>shr edx,4</li> <lishr edx,4<="" li=""> <lishr edx,4<="" li=""></lishr></lishr></ul>	
<pre>     00rc1587     00rc1587     8A0257     8A025     8A025</pre>	
<pre>     U0Ec136/     Obj:Unit:0n0000000000000000000000000000000000</pre>	
00rc1570       8A9C35 F8FEFFFF       mov bl,byte ptr ss:[ebp+esi-108]         00rc1577       F7E6       mov ax,esi         00rc1578       8E60       mov ex,esi         00rc1578       CIEA 03       shr edx,3         00rc1588       CIEA 03       shr edx,4         00rc1588       CIEA 03       sub eax,ecx         00rc1588       CIEA 03       sub eax,ecx         00rc1588       CIEA 03       sub eax,ecx         00rc1589       OrB64438 0C       movzx eax,byte ptr ds:[eax+edi+C]         00rc1599       OrB67       movzx eax,byte ptr ds:[eax]         00rc1599       OrB67       movz eax,byte ptr ds:[eax]         00rc1591       03C8       add ecx,eax         00rc1581       03FEFFFFF       mov byte ptr ds:[ecx],bl         00rc1581       7C B8       mov bi,byte ptr ss:[ebp-18]         00rc1581       33F6       mov csi,esi         00rc1582       7C F8       mov csi,esi         00rc1583       33F6       mov csi,esi         00rc1584       88B9 E8FFFFF       mov cl,byte ptr ds:[e	
<pre> • 00rc1577 B8 8988888 mov exx,888889 • 00rc1577 B8 606 mov exx,888889 • 00rc1580 B8C6 mov exx,esi • 00rc1581 CLEA 03 mov exx,esi • 00rc1583 CLE1 04 shl ecx,edx • 00rc1584 28CA sub ecx,edx • 00rc1584 28CA sub ecx,edx • 00rc1584 28CA sub ecx,edx • 00rc1587 02C3 • 00rc1599 07886438 0c movzx exx,byte ptr ds:[eax+ed1+C] • 00rc1599 0788677 movzx eax,byte ptr ds:[eax+ed1+C] • 00rc1599 0788677 movzx eax,byte ptr ds:[eax+ed1+C] • 00rc1599 0788678 add bh,al • 00rc1599 0788678 mov exx,bth • 00rc1599 0788678 movzx eax,byte ptr ds:[ecx] • 00rc1599 0788678 movzx eax,byte ptr ds:[ecx] • 00rc1598 0788 B88435 F8FEFFFF mov byte ptr ds:[ecx],bl • 00rc1598 0788 B819 mov byte ptr ds:[ecx],bl • 00rc1588 8819 mov byte ptr ds:[ecx],bl • 00rc1588 8819 mov byte ptr ds:[ecx],bl • 00rc1588 8819 mov byte ptr ss:[ebp-118] • 00rc1588 8819 mov byte ptr ss:[ebp-118] • 00rc1588 8819 mov byte ptr ss:[ebp-118] • 00rc1588 8819 mov byte ptr ds:[eax+eax],ax • 00rc1508 8440 F9 mov cd1,byte ptr ss:[ebp-108] • 00rc1508 8440 F9 mov db,byte ptr ds:[eax+eax],ax • 00rc1508 8440 F9 mov db,byte ptr ds:[eax+eax],ax • 00rc1508 8440 F9 mov db,byte ptr ds:[eax+e</pre>	
<pre>     Our CL157;</pre>	
<pre>00Fc15xC F/E6 mules1 00Fc15x8 C1EA 03 shredx,3 00Fc15x8 C1EA 03 shredx,3 00Fc15x8 C1E1 04 shl ecx,edx 00Fc15x8 C1E1 04 shl ecx,edx 00Fc15x8 C2BCA sub ecx,edx 00Fc15x8 2BCA sub ecx,edx 00Fc15x8 0F8FEFFF lea ecx,dword ptr ss:[ebp-108] 00Fc15y9 02F8 add al,bl 00Fc15y9 02F8 add b,al 00Fc15y8 0F8BCF7 mov2x eax,byte ptr ds:[ecx] 00Fc15y8 0F8BCF7 mov2x eax,eax 00Fc15y8 0F8BCF7 mov2x eax,ebx 00Fc15y8 0F8BCF7 mov2x eax,byte ptr ds:[ecx] 00Fc15y8 0F8BCF7 mov2x eax,ebx 00Fc15x0 0F8B50 mov2x eax,ebx 00Fc15x0 88H19 mov byte ptr ds:[ecx],bl 00Fc15x8 88H9 mov byte ptr ds:[ecx],bl 00Fc15x8 88H9 mov edi,dword ptr ss:[ebp-118] 00Fc15x8 88FF mov edi,dword ptr ss:[ebp-118] 00Fc15x8 84A7 F8 mov di,dword ptr ss:[ebp-118] 00Fc15x8 84A7 F8 mov cl,byte ptr ss:[ebp-108] 00Fc15x8 84A7 F8 mov cl,byte ptr ss:[ebp-108] 00Fc15x8 0F8FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</pre>	
<ul> <li>00FC157E</li> <li>00FC157E</li> <li>00FC1585</li> <li>00FC1585</li> <li>00FC1585</li> <li>00FC1585</li> <li>00FC1585</li> <li>00FC1585</li> <li>00FC1585</li> <li>20FC1</li> <li>00FC1585</li> <li>20FC1</li> <li>00FC1585</li> <li>20FC1</li> <li>00FC1585</li> <li>20FC1</li> <li>00FC1585</li> <li>20FC1</li> <li>00FC1585</li> <li>00FC1585</li> <li>00FC1585</li> <li>00FC1586</li> <li>00FC1587</li> <li>00FC1587</li> <li>00FC1599</li> <li>00</li></ul>	
<pre>     Ourclis80</pre>	
<pre>Clear D3 Clear D4 clear D</pre>	
<pre></pre>	
<pre>     OPECIS85 CLE1 04 shl ecx,4     OPECIS8 28CA sub ecx,edx     Sub ecx,edx     OPECIS8 28C1 8FEFFFF lea.ecx,dword ptr ss:[ebp-108]     OPECIS9 07E458 add bh,al     OPECIS9 07E48 add bh,al     OPECIS40 0FB601 movx eax,byte ptr ds:[ecx]     OPECIS40 0FB601 movx eax,byte ptr ds:[ecx],bl     OPECIS40 884435F8FEFFFF mov byte ptr ds:[ecx],bl     OPECIS40 8819 mov eff.dbin.FCL570     OPECIS40 8819 mov eff.dbin.FCL570     OPECIS40 8870 F8     OPECIS40 88</pre>	
<pre>     Our Liss Sub ex, ex     Our Liss Sub ex, ex     Our Liss Sub ex, ex     Sub ex     Sub ex, ex     Sub ex     Sub ex, ex     Sub ex</pre>	
<pre>sub control = sub control</pre>	
<ul> <li>ODECLSRA</li> <li>ODECLSRA</li> <li>ODECLSRC</li> <li>ODECLSPC</li> <li>ODE</li></ul>	
<ul> <li>ODFC158C</li> <li>ODFC1592</li> <li>ODFC1592</li> <li>ODFC1597</li> <li>O2C3</li> <li>add al,bl</li> <li>add al,bl</li> <li>add al,bl</li> <li>oDFC1597</li> <li>O2C3</li> <li>add bh,al</li> <li>movzx eax,bh</li> <li>oDFC1588</li> <li>OFB6C7</li> <li>movzx eax,bt</li> <li>oDFC1589</li> <li>OFB6C1</li> <li>movzx eax,bt</li> <li>oDFC1580</li> <li>OFB601</li> <li>movzx eax,bteptr ds:[ecx]</li> <li>mov byte ptr ds:[ecx],bl</li> <li>ODFC15A8</li> <li>B8B0</li> <li>E8FEFFFF</li> <li>mov byte ptr ds:[ecx],bl</li> <li>ODFC15A8</li> <li>B8B0</li> <li>E8FEFFFFF</li> <li>mov byte ptr ds:[ecx],bl</li> <li>ODFC15A8</li> <li>Cmp Len S-Box (255 Decimal)</li> </ul> Cmp Len S-Box (255 Decimal) Cmp Len S-Bo	
OperationOperationOperationOperationMove car, byte ptr ds:[eax+edi+C]KeyODFC1592OPE64438 OCmovex eax, byte ptr ds:[eax+edi+C]add al, blmovex eax, byte ptr ds:[eax+edi+C]KeyODFC1599O2F8add dcx, eaxmovex eax, byte ptr ds:[ecx]movex eax, byte ptr ds:[ecx]KeyODFC15A0OF8601movex eax, byte ptr ds:[ecx], blScramble S-Box (PRGA/S-Box PermutationODFC15A3888435 F8FEFFFFmov byte ptr ds:[ecx], blScramble S-Box (255 Decimal)ODFC15A446inc esimov edi, dword ptr ss:[ebp-118]Scramble S-Box (255 Decimal)ODFC15A381FE D0010000cmp esi, 100Cmp Len S-Box (255 Decimal)ODFC15A381FE D0010000cmp esi, esiCmp Len S-Box (255 Decimal)ODFC15A381FE D0010000mov edi, dword ptr ss:[ebp-118]Cmp Len S-Box (255 Decimal)ODFC15A38A7D F8mov bh, byte ptr ss:[ebp-7]test edi, ediODFC15C38A7D F8mov cl, byte ptr ss:[ebp-7]test edi, ediODFC15C466:0F1F8400 00000000nop word ptr ds:[eax+eax], axin chODFC15D0FEC7in chlea edx, dword ptr ss:[ebp-108]mov bl, CTODFC15D80A7L0A14mov bl, byte ptr ds:[edx]mov bl, CTODFC15D80A1Amov bl, byte ptr ds:[edx]mov bl, CTODFC15DF0F86C1movzx eax, clmovzx eax, cl	
<pre>     OUPCLS92     OUPCLS92     OUPCLS92     OUPCLS92     OUPCLS92     OUPCLS9     OUP</pre>	
<pre>     OurclSP9 02c3 add al,bl     ourclSP9 02c8 add al,bl     add bh,al     movzx eax,bh     ourclSP8 0F86C7 movzx eax,bt     ourclSA0 0F6601 movzx eax,byte ptr ds:[ecx]     ourclSA1 888435 F8FEFFFF mov byte ptr ds:[ecx],bl     ourclSA8 8819 mov byte ptr ds:[ecx],bl     ourclSA8 8819 mov byte ptr ds:[ecx],bl     ourclSA8 8819 mov byte ptr ds:[ebp+esi-108],al     ourclSA8 8819 mov byte ptr ds:[ecp],bl     ourclSA8 8819 mov byte ptr ds:[ebp-118]     ourclSA8 88B0 E8FEFFFF mov edi,dword ptr ss:[ebp-118]     ourclSA8 33F6 xor esi,esi     ourclSC5 7F5 test edi,edi     ourclSC7 66:0F1F8400 0000000 inp word ptr ds:[eax+eax],ax     inc bh     ourclSC9 8095 F8FEFFFF lea edx,dword ptr ss:[ebp-108]     ourclSC9 8095 F8FEFFFF lea edx,dword ptr ds:[eax,bh     ourclSC9 8095 F8FEFFFF lea edx,dword ptr ds:[eax],ax     inc bh     ourclSC9 803D0 add edx,eax     ourclSC9 803D0 add edx,eax     ourclSC9 803D0 add edx,eax     ourclSC9 803D0 add edx,eax,cl     ourclSC9 8042 movz eax,cl     ourclSC9 8043 movz eax,cl     ourclSC9 8</pre>	
<ul> <li>00FC1599</li> <li>00FC1599</li> <li>00FC1599</li> <li>00FC1590</li> <li>00FC1500</li> <li>00F</li></ul>	
<ul> <li>OUTCL1998</li> <li>OUTC</li></ul>	
<ul> <li>OUFC1598</li> <li>OUFC1598</li> <li>OUFC1500</li> <li>SB80 E8FEFFF</li> <li>OUFC1500</li> <li>OUFC1500</li> <li>SB7E</li> <li>OUFC1500</li> <li>SB7E</li> <li>OUFC1500</li> <li>SA7D F8</li> <li>OUFC1500</li> <li>SAFF test edi.edi</li> <li>Ile main_bin.FC1610</li> <li>OUFC1500</li> <li>OUFC1500</li> <li>SAFF test edi.edi</li> <li>Ile main_bin.FC1610</li> <li>OUFC1500</li> <li>OUFC1500</li> <li>SAFF test edi.edi</li> <li>Inc with byte ptr ds:[eax+eax],ax</li> <li>Inc with byte ptr ds:[eax+eax],ax</li> <li>OUFC1500</li> <li>OUFC150</li></ul>	
<ul> <li>00FC15A0 00FC15A0 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15A3 00FC15B3 00FC15B3 00FC15B3 00FC15B3 00FC15B3 00FC15B3 00FC15B3 00FC15B3 00FC15B3 00FC15C3 00FC15D2 00F</li></ul>	
<ul> <li>ODFC15A0 ODFC15A0 ODFC15A0 ODFC15A0 ODFC15A0 ODFC15AA ODFC15AA ODFC15AA ODFC15AB ODFC15AB ODFC15BB ODFC15BB ODFC15BB ODFC15BB ODFC15BB ODFC15BB ODFC15C5 ODF</li></ul>	
<pre>vorc15A3 vorc15A3 vorc15A3 vorc15A3 vorc15A3 vorc15A3 vorc15A3 vorc15A3 vorc15A4 vorc15A</pre>	
<ul> <li>OUFC15A3</li> <li>OUFC15A8</li> <li>OUFC15A8</li> <li>OUFC15A8</li> <li>OUFC15A8</li> <li>OUFC15A8</li> <li>OUFC15A8</li> <li>OUFC15A8</li> <li>B1FE 00010000</li> <li>cmp esi,100</li> <li>c</li></ul>	
<ul> <li>00FC15AB</li> <li>00FC15AB</li> <li>00FC15AB</li> <li>00FC15AB</li> <li>00FC15AB</li> <li>00FC15BB</li> <li>A 7C BB</li> <li>00FC15BB</li> <li>00FC15BB</li> <li>00FC15BB</li> <li>00FC15BB</li> <li>00FC15BB</li> <li>00FC15BB</li> <li>00FC15BB</li> <li>00FC15C0</li> <li>00FC15C0</li> <li>00FC15C3</li> <li>00FC15C0</li> <li>00FC15D0</li> <li>FEC7</li> <li>10 eadx, dword ptr ds: [eax+eax], ax</li> <li>inc bh</li> <li>10 eadx, dword ptr ds: [eax+eax], ax</li> <li>inc bh</li> <li>00FC15D0</li> <li>00FC15</li></ul>	
<ul> <li>ODFCISAB</li> <li>ODFCISAB</li> <li>8819</li> <li>NOV byte ptr ds:[ecx],bl</li> <li>Cmp Len S-Box (255 Decimal)</li> <li>Cmortise</li> <li>Cmortise</li> <li>Cmortise</li> <li>Cmp Len S-Bo</li></ul>	
00FC15AD     80FF     00F0 byte ptr 0s1[eck][off       00FC15BB     80FE     00F0 byte ptr 0s1[eck][off       00FC15BB     7C BB     11 main_bin_FC1570       00FC15BB     80BD E8FEFFFF     mov edi, dword ptr ss:[ebp-118]       00FC15BB     33F6     xor esi, esi       00FC15BB     8A7D F8     mov bh, byte ptr ss:[ebp-7]       00FC15C0     8AAD F9     mov cl, byte ptr ds:[ebp-7]       00FC15C3     85FF     test edi, edi       00FC15C4     7E 56     jle main_bin_FC1610       00FC15C5     7E 56     jle main_bin_FC1610       00FC15D0     FEC7     inc bh       00FC15D0     FEC7     inc bh       00FC15D0     S8FEFFFF     lea edx, dword ptr ss:[ebp-108]       00FC15D8     03D0     add edx, eax       00FC15D8     03D0     add edx, eax       00FC15D8     03D0     add edx, eax       00FC15DF     02CB     add cd, bl       00FC15DF     02CB     add cd, bl       00FC15DF     02CB     mov zc eax, cl	
•       00rcl3bal       81FE 00010000       cmp esi,100	
<ul> <li>ODFC1583</li> <li>A 7C BB</li> <li>IDFC1583</li> <li>B 8BD E&amp;FEFFFF</li> <li>ODFC158B</li> <li>ODFC158B</li> <li>B 8A7D F&amp;</li> <li>ODFC15C0</li> <li>B 8A7D F&amp;</li> <li>Mov bh,byte ptr ss:[ebp-8]</li> <li>ODFC15C3</li> <li>B 5FF</li> <li>UDFC15C3</li> <li>UDFC15C3</li> <li>B 5FF</li> <li>UDFC15C4</li> <li>UDFC15C5</li> <li>C 650F1F8400 0000000</li> <li>Mow dptr ds:[eax+eax],ax</li> <li>Ibh</li> <li>ODFC15D8</li> <li>ODFC15D8</li> <li>ODFC15D8</li> <li>ODFC15D8</li> <li>B 7BFEFFFF</li> <li>Iea edx,dword ptr ds:[eax+eax],ax</li> <li>ODFC15D8</li> <li>ODFC15</li></ul>	
<ul> <li>DORCISES</li> <li>SBBD E&amp;FFFFFF</li> <li>DOFC15BB</li> <li>OOFC15BB</li> <li>OOFC15C0</li> <li>BA7D F&amp;</li> <li>MOV bh, byte ptr ss: [ebp-8]</li> <li>OOFC15C0</li> <li>BA4D F9</li> <li>MOV cl, byte ptr ss: [ebp-7]</li> <li>OOFC15C3</li> <li>TE 56</li> <li>OOFC15C4</li> <li>OOFC15C5</li> <li>TE 56</li> <li>OOFC15C5</li> <li>OOFC15C5</li> <li>TE 56</li> <li>OOFC15C6</li> <li>BO95 F&amp;FFFF</li> <li>Ie main_bin.FC161D</li> <li>OOFC15C7</li> <li>G6: OF 1F8400 0000000</li> <li>Mow dh yth ds: [eax+eax], ax</li> <li>OOFC15D2</li> <li>BO95 F&amp;FEFFFFF</li> <li>Iea edx, dword ptr ss: [ebp-108]</li> <li>OOFC15D2</li> <li>BO95 F&amp;FEFFFFF</li> <li>Iea edx, dword ptr ds: [edx]</li> <li>OOFC15D8</li> <li>OOFC1</li></ul>	
00FC158B     33F6     xor esi,esi       00FC158B     8A7D F8     mov bh,byte ptr ss:[ebp-8]       00FC15C0     8A4D F9     mov cl,byte ptr ss:[ebp-7]       00FC15C3     85FF     test edi,edi       00FC15C6     7E 56     jle main_bin.FC1610       00FC15C7     66:0F1F8400 0000000     nop word ptr ds:[eax+eax],ax       00FC15D0     FEC7     inc bh       00FC15D0     FEC7     movz: eax,bh       00FC15D8     03D0     add edx,eax       00FC15D8     03D0     add edx,eax       00FC15DF     02C8     add cl,bl       00FC15E1     0F86C1     movz: eax,cl	
• 00FC158B       33+0       xor esi,esi         • 00FC158D       8A7D F8       mov bh,byte ptr ss:[ebp-8]         • 00FC15C3       85FF       test edi,edi         • 00FC15C5       7E 56       jle main_bin.FC161D         • 00FC15C5       66:0P1F8400 00000000       nop word ptr ds:[eax+eax],ax         • 00FC15D2       FEC7       inc bh         • 00FC15D2       8D95 F8FEFFFF       lea edx,dword ptr ss:[ebp-108]         • 00FC15D8       0F86C7       movzx eax,bh         • 00FC15D8       0B6C7       movzx eax,bh         • 00FC15D8       03D0       add edx,eax         • 00FC15D8       03D0       add edx,eax         • 00FC15DF       02C8       add cl,bl         • 00FC15DF       02C8       add cl,bl         • 00FC15E1       0F86C1       movz eax,cl	
<ul> <li>00FC15ED</li> <li>8A7D F8</li> <li>mov bh,byte ptr ss:[ebp-8]</li> <li>00FC15C3</li> <li>85FF</li> <li>test edi,edi</li> <li>00FC15C7</li> <li>66:0F1F8400 0000000</li> <li>nop word ptr ds:[eax+eax],ax</li> <li>00FC15D8</li> <li>00FC1</li></ul>	
<ul> <li>00FC15C0</li> <li>8A4D F9</li> <li>mov cl,byte ptr ss:[ebp-7]</li> <li>00FC15C3</li> <li>7E 56</li> <li>00FC15C5</li> <li>7E 56</li> <li>00FC15C5</li> <li>7E 56</li> <li>00FC15C7</li> <li>00FC15D0</li> <li>6E:0F1F8400 0000000</li> <li>nop word ptr ds:[eax+eax],ax</li> <li>00FC15D2</li> <li>8D95 F8FEFFFF</li> <li>lea edx,dword ptr ss:[ebp-108]</li> <li>00FC15D8</li> <li>00FC15D8</li> <li>00FC15D8</li> <li>00FC15D9</li> <li>00FC15D1</li> <li>00FC15D1</li> <li>00FC15D2</li> <li>00FC15D2</li> <li>00FC15D2</li> <li>00FC15D2</li> <li>00FC15D1</li> <li>00FC15D2</li> <li>00FC15D1</li> <li>00FC15D1</li> <li>00FC15D1</li> <li>00FC15D1</li> <li>00FC15D2</li> <li>00FC15D1</li> <li>00FC15D2</li> <li>00FC15D1</li> <li>00FC15D2</li> <li>00FC15D2</li> <li>00FC15D2</li> <li>00FC15D1</li> <li>00FC15D2</li> <li>00FC15D2&lt;</li></ul>	
OUPCISC3         OUPCISC3         OUPCISC3         SFF         test edit,edi           00FC15C3         V         7E         56         1         1         main_bin.FC1610           00FC15C7         66:0F1F8400         00000000         nop word ptr ds:[eax+eax],ax         inc bh           00FC15D8         SB95         FBFEFFFF         1	
•     00Fc15c3     85FF     test edi,edi       •     00Fc15c5     •     7E 56       •     00Fc15c7     66:0F1F8400 0000000     nop word ptr ds:[eax+eax],ax       •     00Fc15D0     FEC7     inc bh       •     00Fc15D2     805 F8FEFFFF     lee adx,dword ptr ss:[ebp-108]       •     00Fc15D8     0FB6C7     movzx eax,bh       •     00Fc15D8     03D0     add edx,eax       •     00Fc15D8     03D0     add edx,eax       •     00Fc15DF     02CB     add c1,bl       •     00Fc15E1     0FB6C1     movzx eax,cl	
ODFC15C5         F 56         jle main_bin_FC1610           •         ODFC15C7         66:0F1F8400 0000000         nop word ptr ds:[eax+eax],ax           •         ODFC15D0         FEC7         inc bh           •         ODFC15D8         SP85EFFFF         lea edx,dword ptr ss:[ebp-108]           •         ODFC15D8         OB6C7         movzx eax,bh           •         ODFC15D8         O3D0         add edx,eax           •         ODFC15DF         02CB         add cl,eax           •         ODFC15DF         02CB         add cl,bl           •         ODFC15E1         OFB6C1         movzx eax,cl	
<ul> <li>00Fc15c7</li> <li>66:0F1F8400 0000000 nop word ptr ds:[eax+eax],ax</li> <li>00Fc15b0</li> <li>FEC7</li> <li>inc bh</li> <li>00Fc15b2</li> <li>80F86C7</li> <li>mov zeax,bh</li> <li>00Fc15b8</li> <li>03D0</li> <li>add edx,eax</li> <li>00Fc15b8</li> <li>03D0</li> <li>add edx,eax</li> <li>00Fc15b8</li> <li>03D0</li> <li>add edx,eax</li> <li>00Fc15b7</li> <li>00Fc15b7</li> <li>02C8</li> <li>add cl,b1</li> <li>movzx eax,c1</li> </ul>	
00FC15C1     00FC15D0     FEC7     inc bh       00FC15D2     8D95     F8FEFFF     lea edx,dword ptr ss:[ebp-108]       00FC15D8     03D0     add edx,eax       00FC15DF     02CB     add cl,bl       00FC15E1     0F86C1     movzx eax,cl	
•         OUFC15D0         FEC/         Th obh           •         OUFC15D2         8D95         FFFFF         lea edx,dword ptr ss:[ebp-108]           •         OUFC15D8         OFB6C7         movzx eax,bh           •         OUFC15D8         03D0         add edx,eax           •         OUFC15DB         03D0         add edx,eax           •         OUFC15DF         02CB         add cl,bl           •         OUFC15DF         02CB         add cl,bl           •         OUFC15E1         OFB6C1         movzx eax,cl	
•         00Fc15D2         80P5         F8FEFFFF         lea edx,dword ptr ss:[ebp-108]           •         00Fc15D8         0F86c7         movzx eax,bh           •         00Fc15D8         03D0         add edx,eax           •         00Fc15D8         8A1A         mov bl,byte ptr ds:[edx]         mov bl, CT           •         00Fc15DF         02C8         add cl,bl         movzx eax,cl           •         00Fc15E1         0F86c1         movzx eax,cl         movzx eax,cl	
*         00FC15D8         0FB6C7         movzx eax,bh           *         00FC15D8         03D0         add edx,eax           *         00FC15DD         8AlA         mov bl,byte ptr ds:[edx]         mov bl,CT           *         00FC15DF         02C8         add cl,bl         movzx eax,cl           *         00FC15E1         0FB6C1         movzx eax,cl         movzx eax,cl	
OUTCISDE         O3DO         add edx,eax           00FCISDE         03DO         add edx,eax           00FCISDE         8AIA         mov bl,byte ptr ds:[edx]         mov bl,CT           00FCISDE         02CB         add cl,bl         excl,bl         mov bl,CT           00FCISE1         0F86C1         movzx eax,cl         excl,cl         excl,cl	
OUFC15DB         OSDU         add eax,eax           •         00FC15DD         8AIA         mov bl,byte ptr ds:[edx]         mov bl, CT           •         00FC15DF         02CB         add cl,bl         movzx eax,cl           •         00FC15E1         0FB6C1         movzx eax,cl	
<ul> <li>00Fc15pp</li> <li>00Fc15pr</li> <li>00Fc15pr</li> <li>00Fc15p1</li> <li>00Fc15c1</li> <li>0F86c1</li> <li>movz eax,c1</li> </ul>	
<ul> <li>00Fc15pF</li> <li>02CB</li> <li>add cl,bl</li> <li>00Fc15pI</li> <li>0FB6c1</li> <li>movzx eax,cl</li> </ul>	
* OUFCISE1 OFB6C1 movzx eax,cl	
* OURCISEL OFBOLI MOVEX eax, cl	
a too at a stand the stand s	
UUFCIDE4 888D F/FEFFFF MOV Dyte ptr ss: edp-109,cl	
ODECISEA 8080 E8EEEEEE lea ecx.dword ntr ss: [ebp-108]	
add ecx,eax	
<ul> <li>00FC15F2</li> <li>0FB601</li> <li>movzx eax,byte ptr ds:[ecx]</li> </ul>	
00EC15E5 8802 mov byte ptr ds:[edx].a]	
a Operite? 9810 may be at de Corrigin	
* OUFCLSF7 8619 mov byte ptr os:[ecx],01	
* 00FC15F9 0F8602 movzx eax,byte ptr ds:[edx]	
ODCC15CC 8880 EDEEEEEE mov ecx dword atc ssileba-1101	
* 00FC1602 02C5 add a1,01	
* 00FC1604 0F86C0 movzx eax,al	
<ul> <li>OCC1507 OFB68405 E8FEFEFE movzx eax byte ptr ss: [ebpaeax-108]</li> </ul>	
a Operifor 30040r and both the for any all all you to platent to puffer	
* UDFCLODF SUDFOR XOP BYTE PTP 05: [ES1+ECX], AI XOR TO Plaintext in Buffer	
* 00FC1612 46 inc esi	
* 00EC1613 8A8D F7EEEEE mov cl.byte ptc ss:[ebp-109]	
a 000cc1610 30c7 cmp art ad	
VOCATOR JP JP/ Cill Cill Cill Cill Cill Cill Cill Cil	
• 00FC1618 ^ 7C B3 j1 main_bin.FC15D0	
BODEC1610     8B8D FOFEFFFF     mov ecx.dword ptr ss:[ebp-110]	
<ul> <li>UUFCID281 884D FC   mov ecx,dword ptr SS: edp-4</li> </ul>	

main\_bin.00FC1000

.text:00FC1623 main\_bin.exe:\$1623 #A23

🚛 Dump 1	🚛 Dump 2	🔛 Dump 3 🚦	🔛 Dump 4 🛛 🔛 Dump	5   Watch 1	x=  Locals 🤣 Struct
Address   He	ex			ASCII	
000E0000 41	D 5A 90 00 03	00 00 00 04 00	0 00 00 FF FF 00 00	MZÿÿ	
000E0010 B8	8 00 00 00 00	00 00 00 40 00	0 00 00 00 00 00 00 00		
000E0020 0	0 00 00 00 00	00 00 00 00 00	0 00 00 00 00 00 00 00		
000E0030 0	0 00 00 00 00	00 00 00 00 00	0 00 00 00 01 00 00		
000E0040 0E	E 1F BA 0E 00	B4 09 CD 21 B8	8 01 4c CD 21 54 68	º'.1!L1!Th	
000E0050 65	9 73 20 70 72	6F 67 72 61 6D	D 20 63 61 6E 6E 6F	is program canno	
000E0060 74	4 20 62 65 20	72 75 6E 20 69	9 6E 20 44 4F 53 20	t be run in DOS	
000E0070 60	D 6F 64 65 2E	0D 0D 0A 24 00	0 00 00 00 00 00 00 00	mode\$	
000E0080 2/	A E9 99 31 6E	88 F7 62 6E 88	8 F7 62 6E 88 F7 62	*é.ln.÷bn.÷bn.÷b	
000E0090 7/	A E3 F4 63 64	88 F7 62 7A E3	3 F2 63 E1 88 F7 62	zāôcd.+bzāòcá.+b	
000E00A0 7/	A E3 F3 63 7C	88 F7 62 65 E7	7 F2 63 48 88 F7 62	zāóc .+beçòcK.+b	
000E00B0 6	5 E7 F3 63 7F	88 F7 62 65 E7	7 F4 63 7F 88 F7 62	eçőc+beçőc+b	
000E00C0 7/	A E3 F6 63 6D	88 F7 62 6E 88	8 F6 62 38 88 F7 62	zāocm.+bn.ob;.+b	
000E00D0 A/	A E7 FF 63 69	88 F7 62 AA E7	7 F5 63 6F 88 F7 62	°çyci.+b°çõco.+b	
000E00E0 5	2 69 63 68 6E	88 F7 62 00 00	0 00 00 00 00 00 00 00	Richn.+b	
000E00F0 0	0 00 00 00 00	00 00 00 00 00	0 00 00 00 00 00 00 00		
000E0100 50	0 45 00 00 40	01 04 00 0C C2	2 E8 SE 00 00 00 00	PELAe^	
000E0110 0	0 00 00 00 E0	00 02 01 08 01	1 OE 19 00 DA 00 00	aU	
000E0120 0	0 88 00 00 00	00 00 00 F3 22	2 00 00 00 10 00 00		
000E0130 0	0 FO 00 00 00	00 40 00 00 10	0 00 00 00 02 00 00	.0	
000E0140 00	6 00 00 00 00	00 00 00 06 00	0 00 00 00 00 00 00 00		
000E0150 0	0 80 01 00 00	10 00 00 00 00			
000E0160 00	0 00 10 00 00	10 00 00 00 00			
000E01/0 00	4 44 01 00 28	00 00 00 00 00 00		\$n (	

Figure 5: RC4 Decryption Routine Unveils Decrypted Executable

After decrypting the executable, it is loaded and executed via process injection (process hollowing) with the following calls:

- CreateProcessA
- VirtualAlloc

- GetThreadContext
- ReadProcessMemory
- VirtualAllocEx
- WriteProcessMemory
- SetThreadContext
- ResumeThread

Once ResumeThread is called, execution is transferred to the new, decrypted executable.

## Stage 2 Exe

```
Filename: cruloader
MD5: f56a2fd3fd94be87f5c79e822734168d
SHA1: 191050c7ae62c5665938f7666519bcd94f784fd5
SHA256: a4997fbff9bf2ebfee03b9373655a45d4ec3b1bcee6a05784fe4e022e471e8e7
```

Jumping straight into main, the malware first computes the CRC32 of its filename, then checks it against the CRC32 hash 0xB925C42D, which corresponds to svchost.exe. The CRC32 algorithm can be easily identified by 0xedb88320, which is the so-called "reversed" representation of the CRC32 generator polynomial. Luckily for us, 0xB925C42D aka "svchost.exe" is included in the <u>following list of CRC32 API hashes</u>. Lists of API hashes such as this are commonly found on GitHub and can be found simply by Googling for the specific API Hash/constant.

			uintfastcal	l CRC32_Hash(undefined4 param_1	, byte *
uint			EAX:4	<return></return>	
unde	fined	14	ECX:4	param_1	
byte	*		EDX:4	param_2	
int			Stack[0x4]:4	param_3	XF
			CRC32_Hash		XREF[2]:
00401660	55		PUSH	EBP	
00401661	8b e	C	MOV	EBP,ESP	
00401663	51		PUSH	param_1	
00401664	83 30	d 8c	CMP	dword ptr [DAT_0041628c],0x0	
	62 43	1 00			
	00				
0040166b	56		PUSH	ESI	
0040166c	57		PUSH	EDI	
0040166d	8b f:	2	MOV	ESI,param_2	
0040166f	Of 83	5 b0	JNZ	LAB_00401725	
	00 00	00 0			
00401675	33 f:	£	XOR	EDI,EDI	
			LAB_00401677		XREF[1]:
00401677	8b c:	£	MOV	param_1,EDI	
00401679	8b c'	7	MOV	EAX,EDI	
0040167b	d1 e	9	SHR	param_1,1	
0040167d	8b d:	1	MOV	param_2,param_1	
0040167f	81 f:	2 20	XOR	param_2,0xedb88320	
	83 b	8 ed			

#### Figure 6: Compute CRC32 Hash of Filename

Depending on if the filename is svchost.exe, the code will take one of two branches. Branch one, where the filename is not svchost.exe will be examined first, followed by branch two.

#### Branch One: Filename != svchost.exe

If the filename is not svchost.exe, Cruloader begins an anti analysis routine to identify if it is being analyzed. First, it resolves the API IsDebuggerPresent to detect an attached debugger. Next, it resolves CreateToolhelp32Snapshot, Process32FirstW, and Process32NextW in order to compute the CRC32 checksum of every running process to compare against the following analysis tools:

- 7C6FFE70 (processhacker.exe)
- 47742A22 (wireshark.exe)
- D2F05B7D (x32dbg.exe)
- 659B537E (x64dbg.exe)

[stage2_dumped.013D3CE0]=70 FE 6F 7C 22 2A 74 47 7D 5B F0 D2 7E 53 9B 65										
.text:013c101	3 stage2_dum	ped.bin: <b>\$</b> 1013	#413							
🚛 Dump 1	🚚 Dump 2	🚛 Dump 3	🚛 Dump 4	📖 Dump 5	💮 Watch 1					
Address Hex				CTT						
01303050 70	EE 6E 7C 22	2A 74 47 7D	58 E0 D2 7E	53 9B 65 pb						
Figure 7: Pro cal	culated Hash	as to Chock Pu	aning Process	Against						
			Ining Processe		aav#4 14 <b>1</b> aav					
	013C11C0	39448D EC ✓ 74 31	ie sta	ge2_dumped.13C11	F7					
	013c11c6	41	inc ec	к						
•	013c11c7	83F9 04	cmp ec	<b>k</b> ,4	-0					
	013C11CA	A /C F4 8BBD B4EDEEE	= mov ed	j] stage2_dumped.13C11C0						
	013c11b2	8D85 C0FDFFF	- lea ea	k, dword ptr ss:	ebp-240					
•	013c11D8	50	push e	ax						
•	013c11D9	53	push el	x						
	013C11DA 013C11DC	FFD/ 85C0	call e							
·	013C11DE	OF85 BCFEFFF	F ine sta	age2_dumped.13C1	0A0					
•	013C11E4	5F	pop ed	i						
•	013C11E5	5E	pop es	pop esi						
•	013C11E6	33C0	xor eas	k,eax						
	013C11E0	884D FC	mov ec	k k.dword_ptr_ss:	ebp-4					
•	013C11EC	33CD	xor ec	xor ecx,ebp						
•	013C11EE	E8 A80E0000	call s	<pre>call stage2_dumped.13C209B</pre>						
•	013C11F3	8BE5	mov es	mov esp,ebp						
	013C11F5	50 C3	pop eb	)						
L	013c11F7	8B4D FC	mov ec	k,dword ptr ss:	ebp-4					
•	013C11FA	B8 01000000	mov ear	c,1						
•	013C11FF	5F	pop ed							
	013C1200	33CD	pop es	l v ebn						
	013c1203	5B	pop eb	κ, ευρ						
•	013C1204	E8 920E0000	call s	call stage2_dumped.13C209B						
•	013c1209	8BE5	mov es	o,ebp						
•	013C120B	5D C3	pop eb	0						
	013c120D	cc	int3							
•	013C120E	CC	int3							
•	013C120F	CC	int3							
•	013C1210	33 88EC	push el	op Diesp						
	013c1213	83EC 0C	sub est	, csp						
•	013c1216	53	push e	x						
•	013c1217	56	push e	si						
•	013C1218	5/ EE348D 443C3	push e	an word ntr de foc	*4+13030441					
	01261220	0055 -4	push di	and and an and an	-1-1-					
	•		111							
dword ptr [ebp+e eax=D2F05B7D	cx*4-14]=[003A	F354]=D2F05B7D								

#### .text:013C11C0 stage2\_dumped.bin:\$11C0 #5C0

#### Figure 8: Example Match for x32dbg.exe

If any of the tools listed above are discovered running, the malware exits immediately. After determining that it is not being analyzed/debugged, the malware next resolves the same APIs used previously for process injection.

013c1p50	BA 16D951A8	mov edx, A851D916	CreateProcessA
013C1D55	33c9	xor ecx,ecx	
013c1p57	E8 B4F4FFFF	call stage2_dumped.13C1210	
013C1D5C	BA 2E97584F	mov edx, 4F58972E	WriteProcessMemory
013C1D61	A3 A46A3D01	mov dword ptr ds:[<&CreateProcessA>],ea	
013C1D66	33c9	xor ecx,ecx	
013C1D68	E8 A3F4FFFF	call stage2_dumped.13C1210	
013C1D6D	BA B9BE7238	mov edx, 3872BEB9	ResumeThread
013C1D72	A3 CC6A3D01	mov dword ptr ds:[<&WriteProcessMemory>	
013c1p77	33C9	xor ecx,ecx	
013c1p79	E8 92F4FFFF	call stage2_dumped.13c1210	
013C1D7E	BA 4D822EE6	mov edx,E62E824D	VirtualAllocEx
013C1D83	33C9	xor ecx,ecx	
013C1D85	E8 86F4FFFF	<pre>call stage2_dumped.13C1210</pre>	
013C1D8A	BA 4A0DCE09	mov edx,9CE0D4A	VirtualAlloc
013C1D8F	A3 <u>C86A3D01</u>	<pre>mov dword ptr ds:[&lt;&amp;VirtualAllocEx&gt;],ea</pre>	
013c1p94	33C9	xor ecx,ecx	
013C1D96	E8 75F4FFFF	<pre>call stage2_dumped.13c1210</pre>	
013C1D9B	BA 108C80FF	mov edx,FF808C10	CreateRemoteThread
013C1DA0	A3 <u>C46A3D01</u>	<pre>mov dword ptr ds:[&lt;&amp;VirtualAlloc&gt;],eax</pre>	
013C1DA5	33C9	xor ecx,ecx	
013C1DA7	E8 64F4FFFF	<pre>call stage2_dumped.13C1210</pre>	
013C1DAC	A3 <u>D06A3D01</u>	mov dword ptr ds:[<&CreateRemoteThread>	
013C1DB1	33c0	xor eax,eax	
013C1DB3	C3	ret	

Figure 9: Resolving APIs for Process Injection

After resolving the APIs, the malware decrypts the string

C:\Windows\System32\svchost.exe, then creates a suspended svchost.exe process.

012B1CD3	0F1005 5C3C2C01	movups xmm0, xmmword ptr ds:[12C3C5C]	Ciphertext
012B1CDA	83C4 0C	add esp,C	
012B1CDD	C745 88 44000000	mov dword ptr ss:[ebp-78],44	44:'D'
012B1CE4	8D45 D0	lea eax,dword ptr ss:[ebp-30]	
012B1CE7	0F1145 D0	movups xmmword ptr ss:[ebp-30],xmm0	
012B1CEB	0F1005 6C3C2C01	movups xmm0,xmmword ptr ds:[12C3C6C]	
012B1CF2	50	push eax	
012B1CF3	0F1145 E0	movups xmmword ptr ss:[ebp-20],xmm0	
012B1CF7	FF15 <u>1CF02B01</u>	<pre>call dword ptr ds:[&lt;&amp;lstrlenA&gt;]</pre>	
012B1CFD	33C9	xor ecx,ecx	
012B1CFF	90	nop	
012B1D00	8A540D D0	mov dl,byte ptr ss:[ebp+ecx-30]	
012B1D04	C0C2 04	rol dl,4	
012B1D07	80F2 A2	xor dl,A2	
012B1D0A	88540D D0	mov byte ptr ss:[ebp+ecx-30],dl	
012B1D0E	41	inc ecx	
012B1D0F	3BC8	cmp ecx,eax	
012B1D11	∧ 7C ED	jl stage2_dumped.12B1D00	
012B1D13	FF73 08	push dword ptr ds:[ebx+8]	
012B1D16	8D45 88	lea eax,dword ptr ss:[ebp-78]	
012B1D19	50	push eax	
012B1D1A	6A 00	push 0	
012B1D1C	6A 00	push 0	
012B1D1E	6A 04	push 4	Suspended
012B1D20	6A 00	push 0	
012B1D22	6A 00	push 0	
012B1D24	6A 00	push 0	
012B1D26	6A 00	push 0	
012B1D28	8D45 D0	lea eax,dword ptr ss:[ebp-30]	C:\Windows\System32\svchost.exe
012B1D2B	50	push eax	
012B1D2C	FF15 A46A2C01	call dword ptr ds:[<&CreateProcessA>]	

Figure 10: Decrypt String and Create Suspended Svchost Process

Next, the malware calls VirtualAlloc to allocate memory inside the running process to copy itself into. It then calls VirtualAllocEx to allocate a RWX region inside the new suspended process and uses WriteProcessMemory to write the copy of itself into the new process. Finally, it calls CreateRemoteThread to execute the code injected into svchost.exe.

#### Branch One: Filename == Svchost.exe

Now that CruLoader is running under svchost.exe (whether by changing the name or letting it inject into svchost.exe), the malware will take the other branch. This branch begins by resolving a few APIs for internet activity.



Figure 11: Resolve wininet APIs

Next, the URL configuration is decrypted with a simple rol and xor.

01371E3A		66:A1 9C3C3801	mov ax,word ptr ds:[1383C9C]
01371E40		0F1145 C0	movups xmmword ptr ss:[ebp-40],xmm0
01371E44		66:8945 E0	mov word ptr ss:[ebp-20],ax
01371E48		8D45 C0	lea eax,dword ptr ss:[ebp-40]
01371E4B		0F1005 8C3C3801	movups xmm0, xmmword ptr ds: [1383C8C]
01371E52		50	push eax
01371E53		0F1145 D0	movups xmmword ptr ss:[ebp-30],xmm0
01371E57		FF15 <u>1CF03701</u>	<pre>call dword ptr ds:[&lt;&amp;lstrlenA&gt;]</pre>
01371E5D		33C9	xor ecx,ecx
01371E5F		90	nop
01371E60		8A540D C0	mov dl,byte ptr ss:[ebp+ecx-40]
01371E64	Ī	C0C2 04	rol dl,4
01371E67		80F2 C5	xor dl,C5
01371E6A		88540D C0	mov byte ptr ss:[ebp+ecx-40],dl
01371E6E		41	inc ecx
01371E6F		3BC8	cmp ecx,eax
01371E71	^	7C ED	jl svchost.1371E60

Figure 12: Decrypt URL Config

After decrypting the Pastebin URL, the malware makes a connection to the URL and receives a second URL back. It then makes another request to download a PNG. (Note: User-Agent of CruLoader could be used for detection. This kind of thing used to be more popular in the early 2010s, but <u>Bumblebee Malware did this just last year</u>.) The PNG is then written to the following path C:\Users\USER\AppData\Local\Temp\cruloader\output.jpg.

Next, CruLoader decrypts another string redaolurc. It then searches for this payload marker inside the PNG file.

001D1570		8D7D F0	lea edi,dword ptr ss:[ebp-10]	
001D1573		8B07	mov eax,dword ptr ds:[edi]	edi:"redaolurc"
001D1575		3B06	<pre>cmp eax,dword ptr ds:[esi]</pre>	
001D1577	<ul> <li>V</li> </ul>	75 15	jne svchost.1D158E	
001D1579		8B47 04	mov eax, dword ptr ds:[edi+4]	edi+4:"olurc"
001D157C		3B46 04	<pre>cmp eax,dword ptr ds:[esi+4]</pre>	
001D157F	*	75 OD	jne svchost.1D158E	
001D1581		0FB647 08	movzx eax,byte ptr ds:[edi+8]	
001D1585		3A46 08	<pre>cmp al,byte ptr ds:[esi+8]</pre>	esi+8:"\rIHDR"
001D1588		0F84 A4000000	je svchost.1D1632	
001D158E		41	inc ecx	
001D158F		46	inc esi	
001D1590		3BCA	cmp ecx,edx	
001D1592	^	7C DC	j] svchost.1D1570	
001D1594		8B15 <u>A86A1E00</u>	mov edx,dword ptr ds:[1E6AA8]	
001D159A		33C9	xor ecx,ecx	
001D159C		85D2	test edx,edx	
001D159E	×	74 51	je svchost.1D15F1	
001d15a0		83FA 40	cmp_edx,40	40: '@'
001D15A3	× 1	72 4C	jb svchost.1D15F1	
001D15A5		0F2815 D03C1E00	movaps xmm2, xmmword ptr ds:[1E3CD0]	
001D15AC		8D43 20	lea eax,dword ptr ds:[ebx+20]	
001D15AF		8BF2	mov esi,edx	
001D15B1		83E6 C0	and esi, FFFFFC0	
001D15B4		0F1040 E0	movups xmm0, xmmword ptr ds:[eax-20]	
001D15B8		8D40 40	lea eax,dword ptr ds:[eax+40]	
001D15BB		83C1 40	add ecx,40	
001D15BE		UF28CA	movaps xmm1, xmm2	
001D15C1		00:UFEFC8	pxor xmm1, xmm0	
001D15C5		0F1148 A0	movups xmmword ptr ds:[eax-60],xmm1	
001D15C9		0F1040 B0	movups xmm0, xmmword ptr ds:[eax-30]	
001D15CD		00:0FEFC2	pxor xmmo, xmm2	
00101501		0F1040 C0	movups xmmword ptr ds:[eax-50], xmm0	
00101505		0F1040 C0	movups xmm0, xmmword ptr ds:[eax-40]	
00101509		051140 C0	movine viewood oto de: [asy_40] view0	
00101500		051040 00	movups xmm0 xmmuond ptn ds:[eax-40], Xmm0	
00101565		66:0FFFC2	novaps xmm0, xmm2	
001015E3		051140 00	movius ymmword ntr ds:[aay_301 ymm0	
00101529		3BCE	cmp ecy esi	
00101525	•	72 C3	ib sychost 101584	

Figure 13: Search for Payload Marker 'redaolurc' in PNG

Once the payload marker is found, it XOR decrypts with the key 0x61 'a'.

00	00	00	00	00	00	00	72	65	64	61	6F	6C	75	72	63	redaolurc
4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZÿÿ
в8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	·····@
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	
0E	1F	ΒA	0E	00	в4	09	CD	21	в8	01	4C	CD	21	54	68	º´.Í! <sub>.</sub> .LÍ!Th
69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode\$
04	EA	в9	50	40	8B	<b>D</b> 7	03	40	8B	<b>D</b> 7	03	40	8B	<b>D</b> 7	03	.ê'P@.x.@.x.@.x.
49	F3	44	03	4A	8B	D7	03	4B	E4	D6	02	42	8B	D7	03	IÓD.J.×.KäÖ.B.×.
4B	Е4	D2	02	53	8B	<b>D</b> 7	03	4B	Е4	D3	02	4C	8B	D7	03	KäÒ.S.×.KäÓ.L.×.
4B	E4	D4	02	41	8B	<b>D</b> 7	03	54	E0	D6	02	45	8в	D7	03	KäÔ.A.×.TàÖ.E.×.
40	8B	D6	03	6E	8B	<b>D</b> 7	03	84	Е4	DF	02	42	8в	D7	03	@.Ö.n.xäß.B.x.
84	Е4	28	03	41	8в	<b>D</b> 7	03	84	Е4	D5	02	41	8в	D7	03	.ä(.A.×äÕ.A.×.
52	69	63	68	40	8в	<b>D</b> 7	03	00	00	00	00	00	00	00	00	Rich@.x
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
50	45	00	00	4C	01	05	00	19	AA	E8	5E	00	00	00	00	PELªè∧
00	00	00	00	Е0	00	02	01	0в	01	0e	19	00	0e	00	00	à
00	14	00	00	00	00	00	00	6F	12	00	00	00	10	00	00	
00	20	00	00	00	00	40	00	00	10	00	00	00	02	00	00	@
06	00	00	00	00	00	00	00	06	00	00	00	00	00	00	00	• • • • • • • • • • • • • • • • • • • •
00	60	00	00	00	04	00	00	00	00	00	00	03	00	40	81	
00	00	10	00	00	10	00	00	00	00	10	00	00	10	00	00	
00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	
8C	25	00	00	в4	00	00	00	00	40	00	00	E0	01	00	00	.%´@à
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00	50	00	00	54	01	00	00	48	21	00	00	70	00	00	00	.PTH!p
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00	00	00	00	00	00	00	00	в8	21	00	00	40	00	00	00	····
00	00	00	001	00	00	00	001	00	20	00	001	C4	00	00	00	ä

Figure 14: Decrypted Payload

After decrypting the payload, the malware decrypts the string

C:\Windows\System32\svchost.exe and resolves APIs to once again inject the final payload into svchost.exe.

00p71p50	BA 16D951A8	mov edx. A851D916	CreateProcessA
00071055	3309	YOP ECX ECX	createrrocessa
00071057	E8 B4E4EEEE	call sychost D71210	
00071050	BA 2E97584E	mov edv 4E58972E	WhiteProcessMemony
00071050	A3 A46AD800	mov dword ntr ds: [D86444] eav	writeerrocessmenory
00071061	3300	Non ack ack	
00071060	53C3	call suchast p71210	
00071060		mou adv 38730500	BesumeThread
000/1000	DA D9DE/230	mov eux, 30726EB9	Resumernread
00D/1D/2	A3 CC0AD800	mov dword ptr ds:[DooACC],eax	
00D/1D//	5309	xor ecx, ecx	
00D/1D/9	E8 92F4FFFF	call svchost.D/1210	
00D/1D/E	BA 4D822EE6	mov edx,E62E824D	VirtualAllocEx
00D/1D83	33C9	xor ecx,ecx	
00D71D85	E8 86F4FFFF	call svchost.D/1210	
00D71D8A	BA 4A0DCE09	mov edx,9CE0D4A	VirtualAlloc
00D71D8F	A3 <u>C86AD800</u>	mov dword ptr ds:[D86AC8],eax	
00D71D94	33C9	xor ecx,ecx	
00D71D96	E8 75F4FFFF	call svchost.D71210	
00d71d9b	BA 108C80FF	mov edx,FF808C10	CreateRemoteThread
00D71DA0	A3 C46AD800	mov dword ptr ds:[D86AC4],eax	
00D71DA5	3309	xor ecx,ecx	
00D71DA7	E8 64F4FFFF	call svchost.D71210	
00D71DAC	A3 D06AD800	mov dword ptr ds:[D86AD0],eax	
00D71DB1	33c0	xor eax, eax	
00p71pb3	C3	ret	

Figure 15: Time to Inject into svchost.exe Again

Opening up the newly decrypted payload in Ghidra shows us that we have completed the challenge.

```
undefined4 FUN_00401000(void)
{
    MessageBoxA((HWND)0x0,"Uh Oh, Hacked!!","FUD 1337 Cruloader Payload Test. Don\'t upload to
    VT.",0)
    ;
    return 0;
}
```

Figure 16: Challenge Complete

### Automation

Earlier I showcased the string decrypter and unpacker for stage one. Let's finish automating the config extraction and string decryption for stage two, along with the decryption of the final payload embedded in the PNG. <u>Code available on GitHub</u>. I got a bit lazy with my coding here at the end, but it does the job.

> python3 unpack.py -d -f main\_bin.exe -o output.bin 2022-07-08 14:10:41,895 - CruLoader Unpacker - CRITICAL [\*] Key is: b'6b6b64355964504d32345642586d69' 2022-07-08 14:10:41,896 - CruLoader Unpacker - CRITICAL [\*] Unpacking payload 2022-07-08 14:10:41,896 - CruLoader Unpacker - CRITICAL [\*] Payload written to output.bin 2022-07-08 14:10:41,899 - CruLoader Unpacker - CRITICAL [\*] Encrypted: pe51g5Ceb35ffn --> Decrypted: CreateProcessA 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: F5gG8e514pbag5kg --> Decrypted: SetThreadContext 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: yb14E5fbhe35 --> Decrypted: LoadResource 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: Je9g5Ceb35ffz5=bel --> Decrypted: WriteProcessMemory 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: .5ea5/QPY4// --> Decrypted: kernel32.dll 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: I9egh1/n//b3rk --> Decrypted: VirtualAllocEx 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: F9m5b6E5fbhe35 --> Decrypted: SizeofResource 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: yb3.E5fbhe35 --> Decrypted: LockResource 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: E514Ceb35ffz5=bel --> Decrypted: ReadProcessMemory 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: I9egh1/n//b3 --> Decrypted: VirtualAlloc 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: t5gG8e514pbag5kg --> Decrypted: GetThreadContext 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: s9a4E5fbhe35n --> Decrypted: FindResourceA 2022-07-08 14:10:41,900 - CruLoader Unpacker - CRITICAL [\*] Encrypted: E5fh=5G8e514 --> Decrypted: ResumeThread > python3 extract\_config\_decrypt\_strings.py -f stage2\_bin.exe [\*] Config URL(s): ['hxxps://pastebin[.]com/raw/mLem9DGk']

[\*] Decrypted strings: ['\\output.jpg', 'redaolurc', 'C:\\Windows\\System32\\svchost.exe']

> python3 extract\_payload\_from\_png.py -f cruloaderpng.png -o final\_extracted.bin 2022-07-08 14:04:57,794 - CruLoader Unpacker - CRITICAL [\*] Payload written to final\_extracted.bin