CCleaner Backdoor: Analysis & Recommendations

x crowdstrike.com/blog/protecting-software-supply-chain-deep-insights-ccleaner-backdoor/

Karan Sood

October 4, 2017

4	Windows Applications				10		
Geaner	Internet Explorer Temporary Internet Files History	Analysis Complete - (0.675 secs) 220 MB to be removed. (Approximate size)					
	Cookies Recently Typed URLs	Details of files to be deleted (Note: No files h	ave been dek	eted yet)			
Registry	Last Download Location	Internet Explorer - Temporary Internet Files	3,949 XB	124 files			
	Autocomplete Form History	E Internet Explorer - History	57 KB	5 files			
~	Saved Passwords	E Internet Explorer - Cookies	13 KB	30 files			
Tools	🔡 Windows Explorer	Windows Explorer - Recent Documents	3 KB	6 files			
	 Recent Documents 	System - Temporary Files	92,831 KB	100 files			
	 Run (in Start Menu) 	System - Windows Log Files	20,237 KB	28 files			
- 0	Other Explorer MRUs Thumboal Cache	Google Chrome - Internet Cache	106,526 KB	807 files			
Ontions	 Tasibar Jump Lists 	Google Chrome - Internet History	188 KB	10 files			
	Network Passwords	Google Chrome - Cookies	119 KB	237 files			
	📫 System	Google Chrome - Download History	6 KB	1 fies			
	 Empty Recycle Bin 	Google Chrome - Session	221 KB	16 files			
	 Temporary Files 	Applications - Adobe Reader DC	116 KB	3 files			
	✓ Clipboard	💥 Utilities - Windows Defender	10 KB	1 fies			
	 Memory Dumps 						
	 Chkdsk File Fragments 	Analyze		Run Cleane	r		

The term "supply chain attacks" means different things to different people. To the general business community, it refers to attacks targeting vulnerable third-parties in a larger organization's supply chain. A well-known retail chain's massive breach in 2013 is a classic example: Adversaries used a poorly protected HVAC vendor as their gateway to hack into the giant retailer's enterprise network. However, threat researchers have another definition: To them, supply chain attacks can also denote the growing phenomenon in which malicious code is injected into new releases and updates of legitimate software packages, effectively turning an organization's own software supply infrastructure into a potent and hard-to-prevent attack vector. The recent backdoor that was discovered embedded in the legitimate, signed version of CCleaner 5.33, is just such an attack.

To help inform the user community and empower them to better defend against software supply chain attacks, the CrowdStrike® Security Response Team (SRT) conducted a thorough analysis of the CCleaner backdoor. A popular PC optimization tool, the 5.33 version of CCleaner has had widespread distribution across multiple industries, but the embedded code appeared to actually be targeted at specific groups in the technology sector. (More information on targeted industries is available for CrowdStrike customers in our Falcon Intelligence™ portal.) <u>CrowdStrike's threat intelligence team</u> had also previously reported on the malware's C2 (command and control) infrastructure in a recent alert for CrowdStrike customers identifying possible links to Aurora Panda. The report also outlines the potential for additional adversary tactics, techniques and procedures (TTPs).

CCleaner

CCleaner is a PC cleaning utility developed by Piriform, which was recently acquired by antivirus (AV) provider Avast in June 2017. The affected version of the utility contains a modified __scrt_common_main_seh function that routes the execution flow to a custom function meant to decode and load the malware. This takes place even before the entry point (EP) of the utility is reached. The new execution flow leads to a function that decodes a blob of data, as reproduced in Python below:

```
def decode(indata):
    key = 0x2547383
i = 0
dec = []
for i in range(0, len(indata)):
    key = ((key * 0x47a6547) & 0xFFFFFFF) & 0xFF
    dec.append(blob[i] ^ key)
    key = key >> 0x8
```

return dec

The result of the decoding subroutine is shellcode and the payload (which is missing the IMAGE_DOS_HEADER field). The missing IMAGE_DOS_HEADER is likely to subvert AV solutions that search for MZ (0x4d5a) headers in memory.

Next, the program creates a memory heap with the flag HEAP_CREATE_ENABLE_EXECUTE to allow for execution, and copies the shellcode on the heap, and executes it.

ShellCode

The shellcode is responsible for loading the payload in memory. It attains the PEB (Process Environment Block) of the malware process to load kernel32.dll and find the location of the function **GetProcAddress**. This function is used to retrieve the addresses of functions such as **VirtualAlloc**, **memcpy**, and **LoadLibrary**. It allocates PAGE_EXECUTE_READWRITE memory to which it copies the previously decoded payload (minus the IMAGE_DOS_HEADER) as shown below.

 0000000:
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000
 0000

0000030:	0000	0000	0000	0000	0000	0000	d000	0000	
0000040:	0000	0000	0000	0000	0000	0000	0000	0000	
0000050:	0000	0000	0000	0000	0000	0000	0000	0000	
0000060:	0000	0000	0000	0000	0000	0000	0000	0000	
0000070:	0000	0000	0000	0000	0000	0000	0000	0000	
0000080:	0000	0000	0000	0000	0000	0000	0000	0000	
0000090:	0000	0000	0000	0000	0000	0000	0000	0000	
00000a0:	0000	0000	0000	0000	0000	0000	0000	0000	
00000b0:	0000	0000	0000	0000	0000	0000	0000	0000	
00000c0:	0000	0000	0000	0000	0000	0000	0000	0000	
:00000d0	5045	0000	4c01	0200	c23a	8059	0000	0000	PELY
00000e0:	0000	0000	e000	0e21	0b01	0600	0020	0000	·····
00000f0:	0002	0000	0000	0000	0011	0000	0010	0000	
0000100:	0030	0000	0000	0010	0010	0000	0002	0000	.0
0000110:	0400	0000	0000	0000	0400	0000	0000	0000	
0000120:	0040	0000	0004	0000	0000	0000	0200	0000	.@
0000130:	0000	1000	0010	0000	0000	1000	0010	0000	
0000140:	0000	0000	1000	0000	0000	0000	0000	0000	
0000150:	4c28	0000	dc00	0000	0000	0000	0000	0000	L(
0000160:	0000	0000	0000	0000	0000	0000	0000	0000	
0000170:	0030	0000	c000	0000	0000	0000	0000	0000	.0
0000180:	0000	0000	0000	0000	0000	0000	0000	0000	
0000190:	0000	0000	0000	0000	0000	0000	0000	0000	
00001a0:	0000	0000	0000	0000	0010	0000	0001	0000	
00001b0:	0000	0000	0000	0000	0000	0000	0000	0000	
00001c0:	0000	0000	0000	0000	2e74	6578	7400	0000	text
00001d0:	041e	0000	0010	0000	0020	0000	0004	0000	
00001e0:	0000	0000	0000	0000	0000	0000	2000	00e0	
00001f0:	2e72	656C	6f63	0000	1a01	0000	0030	0000	.reloc0
0000200:	0002	0000	0024	0000	0000	0000	0000	0000	\$
0000210:	0000	0000	4000	0042	0000	0000	0000	0000	@B

Once the payload is copied to the newly allocated memory, the shellcode resolves the needed API's, and calls the OEP (original entry point) of the payload in memory.

Payload

Environment Checks

Once it's loaded, the payload creates a thread that performs the core functionality of the malware. It performs a few checks at the onset of the environment and the user privileges. The malware employs the function **msvcrt.time** to record the current time of the malware. It then uses **IcmpCreateFile** and **IcmpSendEcho** to send an IPv4 ICMP echo to an invalid IP address, with a timeout of 601 seconds. This is meant to delay the execution of the malware by 601 seconds; this delay is then measured by calling **msvcrt.time** again, and ensuring that more than 600 seconds have elapsed between the first and second calls to the function. It

should be noted that if the call to **IcmpCreateFile** fails, the malware will just sleep for 600 seconds. These steps are measures against debugging and/or sandboxing. It also invokes **IsUserAnAdmin** to ensure that the current user is member of the administrator's group. If either of these checks fails, the malware exits immediately.

It uses a decoding scheme as the one described above to decode strings during runtime in memory. It is important to note that these dynamically decoded strings are zeroed out in memory before each function using them exits. The strings dynamically decoded throughout the execution of the malware are listed in the Appendix section of this blog.

The malware also checks the privilege levels of its own process; if the process does not have administrative privileges, it uses **AdjustTokenPrivileges** to enable the **SeDebugPrivilege** value for the process. This enables the process to either debug or adjust memory for a process owned by another account.

Registry Checks

The malware checks for the following registry key:

HKLM\SOFTWARE\Piriform\Agomo\TCID. The key value is supposed to hold a system time value; if the value is greater than the current time, the malware will terminate. It also checks the value of **HKLM\SOFTWARE\Piriform\Agomo\MUID**. If the key does not exist, the malware will set its value using a pseudo-random number derived in the following manner:

```
// Pseudocode to calculate MUID val
DWORD MUID;
unsigned int seed, rand1, rand2;
seed = GetTickCount();
srand(seed);
rand1 = rand();
rand2 = rand() * rand1;
MUID = GetTickCount() ^ rand2;
```

Gathering Victim Information

Once the checks are completed, the malware gathers the following information about the victim machine:

- OS major version
- OS minor version
- OS architecture
- Computer name
- Computer DNS domain

- IPv4 addresses associated with the machine. This information is gathered by calling GetAdaptersInfo, and then enumerating through each adapter to search for the IP_ADAPTER_INFO → IpAddressList → IpAddress field.
- Installed applications. The malware accesses the registry key HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall, and enumerates through each key, and compares the *Publisher* value with "Microsoft Corporation." If there is a match, it moves on to the next value. If not, it will attain the DisplayName value using **SHGetValueA**, and insert it into memory. Each name is prepended with an "S" in memory.
- Full name of the executable image of each running process. The malware calls
 WTSEnumerateProcessA to get a pointer to an array of WTS_PROCESS_INFO
 structures, which are then used to get the *ProcessName* field for each process. Each
 process name is prepended with a "P" in memory.

This information is stored in a data structure in memory in the following manner:



The MUID_Val is used as a unique identifier for the victim machine.

Next, the structure is encoded in memory in two steps:

- Step 1: Aforementioned scheme
- Step 2: Modified version of base64

The image below displays the data structure as it goes through each encoding step.

0000000: 5043 3a5c 5769 6e64 6f77 735c 5379 7374 PC:\Windows\Syst	0000000: cl2c bl3a 0509 3cc2 d2ba 6e63 b284 221a .,.: <nc".< td=""><td>0000000: 5136 7677 6673 3154 4d53 6d72 2a54 6846 Q6vwfslTMSmr*ThF</td></nc".<>	0000000: 5136 7677 6673 3154 4d53 6d72 2a54 6846 Q6vwfslTMSmr*ThF
0000010: 656d 3332 5c73 706f 6f6c 7376 2e65 7865 em32\spoolsv.exe	0000010: 1887 d6cf 68ce f9a4 3313 580d e73a 1260h3.X:.`	0000010: 4c4e 3167 2a43 7666 2a61 3256 4d51 7061 LN1g*Cvf*a2VMQpa
0000020: 0000 0000 0000 0000 0000 0000	0000020: 6e57 75a7 ebbl 84bb 78b7 c2c2 f008 8741 nWuxA	0000020: 5578 7664 654a 5a32 636c 5747 7338 384f UxvdeJZ2c1WGs880
0000030: 0000 0000 0000 0000 0000 0000	0000030: 7688 27ae 0190 f8f3 d8d0 79al bb2c 6465 v.'vde	0000030: 3678 7a73 4b38 6d42 5476 3136 6158 666b 6xzsK8mBTv16aXfk
0000040: 0000 0000 0000 0000 0000 0000	0000040: 3178 cld4 8358 cfb6 4b3b 3bed 4c72 llcb 1xXK:.Lr	0000040: 7433 4b66 6c53 7876 3245 7768 716c 7961 t3KflSxv2Ewhglya
0000050: 0000 0000 0000 0000 0000 0000	0000050: c2ef a37e 135f 5484 efe7 4f36 71d8 49a0~. TO6g.I.	0000050: 7366 306b 3362 6153 5530 782a 7248 7462 sf0k3baSU0x*rHtb
0000060: 0000 0000 0000 0000 0000 0000	0000060: 7d17 48f6 cc4b ea5f 5d2e 4539 3f2a 3299 }.HK. 1.E9?*2.	0000060: 4b6a 6471 6a33 7449 4849 5846 6574 4530 Kidgi3tIHIXFetE0
0000070: 0000 0000 0000 0000 0000 0000	0000070: 8ce2 4730 f6cc fe3f 1d05 95a1 9442 07f8G0?B	0000070: 3257 3052 644e 6d54 5841 5465 7141 7344 2WORdNmTXATegAsD
0000080: 0000 0000 0000 0000 0000 0000	0000080: b02b 3434 c997 3350 ec3a 0232 01c9 fdfc .+443P.:.2	0000080: 4564 4354 6272 4534 4930 4c42 3579 3855 EdCTbrE4I0LB5v8U
0000090: 0000 0000 0000 0000 0000 0000	0000090: 4c2c ef2c df1b f573 3ca8 300c 9285 0351 Ls<.00	0000090: 6744 6671 2a6f 4a63 786e 3250 4d4f 3434 gDfg*oJcxn2PM044
00000a0: 0000 0000 0000 0000 0000 0000	00000a0: 1260 1662 4f8a b5e0 e8d3 7228 a496 a970 b0r(p	00000a0: 5158 4b63 6853 6445 6576 6a2a 696a 5855 OXKchSdEevi*iiXU
00000b0: 0000 0000 0000 0000 0000 0000	00000b0: 9a9a 7546 ad6d a041 3a79 dd71 f25b e8f8uF.m.A:v.g.(00000b0: 6f38 3732 2130 6639 3739 3164 316e 4330 087210f9791d1nC0
00000c0: 0000 0000 0000 0000 0000 0000	00000c0: 83f1 0d4a f04c 31bc 5b39 82cd a920 e8a8J.L1.[9	00000c0: 5663 6e73 3930 4459 6756 6241 534f 6472 Vcna90DYgVbAS0dr
00000d0: 0000 0000 0000 0000 0000 0000	00000d0: 0b4c c2b9 516b 20b5 25e6 dc5b de0e 5f30 .L.Ok .%.(0	00000d0: 6b6a 4534 4f59 706a 2a6e 4679 3270 636a kiE40Ypi*nFv2pci
00000e0: 0000 0000 0000 0000 0000 0000	00000e0: adab d77e cd5e 84f4 b727 61f3 4665 e9a7	00000e0: 7a4e 656e 446e 4853 6c34 696b 5374 556e zNenDnHS14ikStUn
00000 0000 0000 0000 0000 0000 0000 0000	00000f0: 35c3 90c3 e288 6e2c eb71 7e0f e684 c06b 5pg~k	00000f0: 4736 7771 7646 3338 7068 4672 4e41 4678 G6wgyF38phFrNAFx
0000100: 5043 3a5c 5769 6e64 6f77 735c 5379 7374 PC:\Windows\Syst	0000100: f236 5ede 175a 7b88 1940 4bb3 0a97 2e8f .6^2{8K	0000100: 2163 5931 7254 7567 475a 744d 7972 7879 lcYlrTugGZtMvrxv
0000110: 656d 3332 5c73 7663 686f 7374 2e65 7865 em32\svchost.exe	0000110: a5ce d9ad dfd6 54ba c2e9 5c7c 6dcd c704	0000110: 4661 7a4d 3343 6e49 5552 6578 7441 7552 FazM3CnIURextAuR
0000120: 0000 0000 0000 0000 0000 0000 0	0000120: c19e 6a5f 3e2a c09c f6ca 6f0e 6588 5a89i >*o.e.Z.	0000120: 2a57 6b78 5347 4251 766e 632a 694c 6133 *WkxSGBOvnc*iLa3
0000130: 0000 0000 0000 0000 0000 0000 0	0000130: 4382 2e8c 7551 c099 60dc fb94 9909 8b6a Cu0	0000130: 4e64 4b47 2a69 6138 6153 7467 4959 414e NdKG*ia8aStgIYAN
0000140: 0000 0000 0000 0000 0000 0000 0	0000140: 9773 f68f 81fa cb1b bd8c 219a 51c0 0193 .s	0000140: 304e 594f 4e78 367a 796c 6171 4561 5075 0NYONx6zylagEaPu
0000150: 0000 0000 0000 0000 0000 0000 0	0000150: d253 29bd f51a 79c1 4a15 939a 62e9 aa50 .S)v.JbP	0000150: 6e65 6872 4a74 5743 6755 6674 4261 6273 nehrJtWCguftBabs
0000160: 0000 0000 0000 0000 0000 0000 0	0000160: 6708 963b eeb7 c4b6 0cc0 aca3 5d2a c52b g	0000160: 534b 4a62 4147 6b4b 5265 6956 4d37 4a43 SKJbAGkKReiVM7JC
0000170: 0000 0000 0000 0000 0000 0000 0	0000170: 8a2f 00eb 17e5 fa4d f03d 57ae 21f5 0b0b ./M.=W.!	0000170: 2a59 7149 626d 7636 3366 5674 6f38 4d21 *YgIbmy63fVto8MI
0000180: 0000 0000 0000 0000 0000 0000 0	0000180: dcba 4ed8 d8b5 1642 863a a23a d23a 67a0	0000180: 3677 2a4a 4c33 544b 584f 4638 6e55 3656 6w*JL3TKXOF8nU6V
0000190: 0000 0000 0000 0000 0000 0000 0	0000190: ffa7 5f78 469c 378a 42d4 fb18 5e57 c4fc xF.7.B^W	0000190: 6935 3231 5537 6162 6c4f 7654 3078 4c7a i521U7abl0vT0xLz
00001a0: 0000 0000 0000 0000 0000 0000 0	00001a0: fe51 c8f3 445f bcaf 2ba5 b425 6410 e2d7 .0D+8d	00001a0: 3351 3842 594a 5630 6670 566e 4864 4a63 308BYJV0fpVnHdJc
00001b0: 0000 0000 0000 0000 0000 0000 0	00001b0: be5d 03f8 51e4 0dfb 2fc7 6d2f b37c d6d6 .1.0/.m/.	00001b0: 4639 6166 7969 586f 6c6b 3845 5164 6165 F9afviXolk8E0dae
00001c0: 0000 0000 0000 0000 0000 0000 0	00001c0: 2f73 63cc f45d 3d53 5c40 61ee f5cd 0d29 /sc.,1=S\&a)	00001c0: 6258 6c79 3557 3243 5142 4832 4175 456a bX1v5W2COBH2AuE1
00001d0: 0000 0000 0000 0000 0000 0000 0	00001d0: b4be cab3 fb16 cde4 68cd 1b42 8c41 5634bB.AV4	00001d0: 7661 3931 3441 4c4c 706b 7442 2179 7a32 va914ALLpktBlvz2
00001e0: 0000 0000 0000 0000 0000 0000 0	00001e0: d83a 2282 5a42 b435 572a fa94 110e 22fe .: ".ZB.5W*".	00001e0: 3972 6358 6863 5579 494d 4d32 6c4d 4962 9rcXhcUvIMM21MIb
00001f0: 0000 0000 0000 0000 0000 0000 0	00001f0: 2912 d48d 0243 7d66 a0b2 ffae 2f8c 5bbb)C)f/.[.	00001f0: 6b46 724b 6676 6d36 3546 634e 6577 6b41 kFrKfvm65FcNewkA
0000200: 5043 3a5c 5072 6f67 7261 6d20 4669 6c65 PC:\Program File	0000200: 5fa0 6875 dc50 89e1 e3b4 96fe 13c3 67e8	0000200: 466f 766d 354e 4d49 4265 5870 6e62 7656 Foym5NMIBeXpnbvV
0000210: 735c 436f 6d6d 6f6e 2046 696c 6573 5c41 s\Common Files\A	0000210: 9cb1 6b6b 20f9 9d38 04c1 305a e93d 1186kk80Z.=	0000210: 3148 5330 554a 3335 7331 5678 7877 4754 1HS0UJ35s1VxxwGT
0000220: 646f 6265 5c41 524d 5c31 2e30 5c61 726d dobe\ARM\1.0\arm	0000220: 8fa6 74bc dd4d c0f8 df42 846e d5d9 7c46t.MB.DF	0000220: 7047 2a31 652a 495a 794d 7550 564c 3541 pG*le*IZyMuPVL5A
0000230: 7376 632e 6578 6500 0000 0000 0000 0000 svc.exe	0000230: deb4 d7c9 80a3 24e0 2de3 2b89 eac2 60bc\$.=.+`.	0000230: 5544 5361 4750 6637 5773 3352 4c35 6c4c UDSaGPf7Ws3RL51L
	THE PARTY AND A THE PARTY PARTY AND A VALUE AND A THE PARTY AN	

The custom base64 encoding scheme uses a modified Base64 index table. Rather than the regular table that has the following values:

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/; its table has the following values:

abcdefghijkImnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!*.

C2 Communication

Once the victim machine information has been encoded, the malware queries the registry key **HKLM\SOFTWARE\Piriform\Agomo\NID**. Upon the initial run, the registry key does not exist; however, the malware eventually inserts an IP address computed via a DGA (Domain Generating Algorithm) later in the execution flow. It is interesting to note that even if the registry key exists, the malware extracts the IP address from the registry value, but does not do anything with it. After the registry check, it decodes the hard-coded IP address **216.126.225[.]148**, and attempts to send the encoded data struct to it via an HTTP POST request on port 443. It uses **InternetSetOptionA** to set the following option flags on the HTTP handle:

- SECURITY_FLAG_IGNORE_CERT_DATE_INVALID → Ignores bad or expired SSL certificates from the server
- SECURITY_FLAG_IGNORE_CERT_CN_INVALID → Ignores incorrect SSL certificate common names
- SECURITY_FLAG_IGNORE_WRONG_USAGE → Ignores incorrect usage problems
- SECURITY_FLAG_IGNORE_UNKNOWN_CA → Ignores unknown certificate authority problems
- SECURITY_FLAG_IGNORE_REVOCATION → Ignores certificate revocation problems

The malware also calls **HttpAddRequestHeadersA** to append the domain **speccy.piriform[.]com** to the POST request. This is performed to appear inconspicuous and make it harder to detect. It is also likely an attempt to confuse the analyst performing dynamic analysis of the malware. Once the information is sent to the C2, the malware expects to receive a stage 2, which it reads into a locally allocated memory block. Analysis shows that once stage 2 is received, it is decoded using the same custom Base64 and the decoding algorithm. Once decoded, the functions **GetProcAddress** and **LoadLibraryA** are pushed to the stack, and the EP of stage 2 is called. At the time of analysis, stage 2 was not available.

DGA

If the malware cannot connect to the C2, it employs a Domain Generating Algorithm, or DGA, to generate a domain. The DGA is dependent on the current year and month; therefore, it generates a new domain on a monthly basis. Below is the code, reproduced in C, displaying the DGA utilized by the malware.

```
#include "stdafx.h"
#include <Windows.h>
#include <stdio.h>
void main()
{
SYSTEMTIME st;
DWORD r1, r2, r3, seed;
char buf[100];
const char *format = "ab%x%x.com";
GetLocalTime(&st);
seed = st.wYear * 10000 + st.wMonth;
srand(seed);
r1 = rand();
r2 = rand();
r3 = rand() * r2;
sprintf_s(buf, format, r3, r1);
}
```

The list of domains calculated for all months in the years 2017 and 2018 are listed in the Appendix.

Once the DGA domain for the current month and year has been calculated, the malware calculates an IP address using that domain in the following steps:

- Get a *hostent* structure by calling **gethostbyname** on the generated domain
- Get the *h_addr_list*, which is a NULL terminated list of IP addresses associated with the domain

These A records (127.100.183[.]225 and 10.158.168[.]171) for the domain ab1145b758c30[.]com, as highlighted in the PCAP screenshot below will be used to calculate a new C2 IP address. If there are more than two A records, the malware will only utilize the first two on the list.

📕 *Loo	cal Area Connec	tion [Wireshark	2.2.5 (v2.2.5-0-g	440fd4d)]	=	-		
<u>F</u> ile	<u>E</u> dit <u>V</u> iew <u>G</u> o) <u>C</u> apture <u>A</u> r	nalyze <u>S</u> tatistic	s Telephon <u>y T</u> ools	<u>I</u> nternals <u>H</u> elp			
•) 🔟 🔳 🖉	🖻 🛅 🕽	४ 🔁 🔍 -	🗧 🕸 🌍 ዥ 👱		ର୍ ତ୍ 🖭 🌌	🖻 🍢 💥 🕅	
Filter:	frame[54:23] =	= 0d:61:62:31:31	:34:35:62:37:35:3	8:63:33:30:03:63:6f:6d:00:	Expression	Clear Apply Save		
Protoco	l Length	Info						▲
DNS	77	Star	ndard query	0x764b A ab1145	b758c30.com			
DNS	109	Star	ndard query	response 0x764b	A ab1145b758	c30.com A 127	.100.183.225 A 10	.158.168.171
	Questions: : Answer RRs: Authority R Additional I Queries ab1145b75	1 2 Rs: 0 RRs: 0 8c30 com: 1	tvne∆ cla	ss TN				~
	Name: ab1145b758c30.com [Name Length: 17]							
	[Label (Count: 2]	2055) (1)					=
	Class: IN (0x0001)							
/	Answers							
E	ab1145b75	8c30.com: t	type A, cla	ss IN, addr 127.	100.183.225			
	ab1145b75	8c30.com: t	type A, cla	ss IN, addr 10.1	58.168.171			Ŧ
0020 0030 0040	5d 83 00 00 02 00 0 38 63 33	35 ef bd 00 00 00 00 00 30 03 63 6f	4b 2e 93 61 62 31 6d 00 00	76 4b 81 80 00 31 34 35 62 37 01 00 01 c0 0c	01]5к 35а	vK b1145b75		•
0050	01 00 01 0	00 00 00 05	5 00 04 7f	64 b7 e1 c0 Oc	00	d		E
0060	01 00 01 (00 00 00 05	00 04 0a	9e a8 ab				*
0 💅	This is a respon	se to the DNS q	uery in this fr	Packets: 28 · Displayed:	2 (7.1%) · Dropp	Profile: Default		

The Python code below reproduces the algorithm to calculate the new C2 IP address from the A records of the newly generated domain.

The new C2 IP address derived from the records of the domain ab1145b758c30[.]com is 211.158.54[.]161. The malware will attempt to connect to this C2 as shown below. If the connection is successful, it will subsequently send the encoded data structure and await stage 2.

*Local Area Connection [Wireshark 2.2.5 (v2.2.5-0-g440fd4d)]	x
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>G</u> o <u>C</u> apture <u>A</u> nalyze <u>S</u> tatistics Telephony <u>I</u> ools <u>I</u> nternals <u>H</u> elp	
● ● ◢ ■ ∅ ⊨ 🖹 🗙 😂 9, ⇔ ⇒ ⊋ 7 🖢 Ε 🗐 9, 9, 9, 17 ₩ 🛛 幆 ※ 13	
Filter: tcp.stream eq 0 Expression Clear Apply Save	
Destination Protocol Info	
211.158.54.161 TCP 51682 - 443 [SVN] Seq=0 win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1	
🗷 Frame 16: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0	
Ethernet II, Src: Vmware_4f:89:9d (00:0c:29:4f:89:9d), Dst: Vmware_ec:3a:7c (00:50:56:ec:3a:7c)	
 Internet Protocol Version 4, Src: 192.168.93.131, Dst: 211.158.54.161 0100 = Version: 4 0101 = Header Length: 20 bytes (5) Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 52 	Ŧ
0000 00 50 56 ec 3a 7c 00 cc 29 4f 89 9d 08 00 45 00 .PV.:].)0E. 0010 00 34 0f 47 40 00 80 06 00 00 cc as 5d 83 d3 9e .4.G@] 0020 36 al c9 e2 01 bb 63 23 44 d6 00 00 00 80 02 6c# D 0030 20 00 28 92 00 00 02 04 05 b4 01 03 03 08 01 01 03 08 01 01 01 03 03 08 01	
😑 💅 File: "C:\Users\apollo\AppData\Local\Temp 🛛 Packets: 26 · Displayed: 2 (7.7%) · Dropped: 0 (0.0 🗍 Profile: Default	

Initial (Buggy) Registry Modifications

Once the C2 communication subroutine has ended, the malware makes two registry modifications:

Encodes the newly calculated C2 IP address and attempts to save it in **HKLM\SOFTWARE\Piriform\Agomo\NID**. The encoding scheme is the same as the one mentioned before. Analysis shows that before the registry key string is built, a function is called to change the endianness of 0x44494E (DIN) to 0x4E4944 (NID). However, due to a bug in the code the function incorrectly changes it to 0x004E4944 (prepended with a NULL value). Subsequently, function **SHSetValueA** is called with the following parameters:

- **hKey =** HKEY_LOCAL_MACHINE
- **Subkey =** "SOFTWARE\Piriform\Agomo"
- Value = ""
- ValueType = REG_DWORD
- **Data =** ...
- **DataLength =** 0x4

The parameter **Value** should be "NID"; however, since the string is incorrectly prepended with a NULL value, the function doesn't read the string at all. The C2 IP address is instead saved in **HKLM\SOFTWARE\Piriform\Agomo\Default** as shown below.



Takes the current time value as determined by the earlier call to **msvcrt.time** and adds 172,800 seconds (2 days) to the value. Saves the new value in HKLM\SOFTWARE\Piriform\Agomo\TCID.

Recommendations

Falcon Endpoint will notify you of any additional activity through our Falcon Intelligence detections. The intent behind the malicious packages was to collect an initial set of reconnaissance data; we urge you to block the known IP address and domains at your network perimeter to prevent any communication to the collection server. In addition, we recommend you update to the latest version of the Avast CCleaner software to ensure the embedded malicious code is removed.

For additional information on CrowdStrike's threat intelligence offerings, visit the <u>Falcon</u> <u>Intelligence product page</u>.

Appendix

Hashes

Information regarding the CCleaner binaries that were affected:

 Size:
 9791816

 SHA256:
 1A4A5123D7B2C534CB3E3168F7032CF9EBF38B9A2A97226D0FDB7933CF6030FF

 Compiled:
 Tue, Dec 29 2015, 21:34:49 UTC - 32 Bit EXE

 Version:
 5.33.00.6162

 Signature Valid
 Subject:

 Piriform Ltd
 Issuer:

 Symantec Class 3 SHA256 Code Signing CA

 Size:
 7680216

 SHA256:
 6F7840C77F99049D788155C1351E1560B62B8AD18AD0E9ADDA8218B9F432F0A9

 Compiled:
 Thu, Aug 3 2017, 9:25:13 UTC - 32 Bit EXE

 Version:
 5, 33, 00, 6162

 Signature Valid
 Subject:

 Piriform Ltd
 Issuer:

 Symantec Class 3 SHA256 Code Signing CA

 Size:
 7781592

 SHA256:
 36B36EE9515E0A60629D2C722B006B33E543DCE1C8C2611053E0651A0BFDB2E9

 Compiled:
 Thu, Aug 3 2017, 9:37:49 UTC - 32 Bit EXE

 Version:
 5, 33, 00, 6162

 Signature
 Valid

 Subject:
 Piriform Ltd

 Issuer:
 Symantec Class 3 SHA256 Code Signing CA

The following is the information about the decoded payload in memory: **Size**: 16384 **SHA256**: FA8A55A05CA9E6587C941354628A0E818DCBF42ED3D98C40689F28564F0BFA19 **Compiled**: Tue, Aug 1 2017, 8:24:34 UTC – 32 Bit DLL

Network Artifacts

The following is the infrastructure associated with the CCleaner backdoor:

Infrastructure Connection Type Description

216.126.225[.]148 Port 443 / TCP C2

DGA Domains

Month, Year	Domain	Month, Year	Domain
January, 2017	abde911dcc16[.]com	January, 2018	ab3c2b0d28ba6[.]com
February, 2017	ab6d54340c1a[.]com	Feburary, 2018	ab99c24c0ba9[.]com

March, 2017	aba9a949bc1d[.]com	March, 2018	ab2e1b782bad[.]com
April, 2017	ab2da3d400c20[.]com	April, 2018	ab253af862bb0[.]com
May, 2017	ab3520430c23[.]com	May, 2018	ab2d02b02bb3[.]com
June, 2017	ab1c403220c27[.]com	June, 2018	ab1b0eaa24bb6[.]com
July, 2017	ab1abad1d0c2a[.]com	July, 2018	abf09fc5abba[.]com
August, 2017	ab8cee60c2d[.]com	August, 2018	abce85a51bbd[.]com
September, 2017	ab1145b758c30[.]com	September, 2018	abccc097dbc0[.]com
October, 2017	ab890e964c34[.]com	October, 2018	ab33b8aa69bc4[.]com
November, 2017	ab3d685a0c37[.]com	November, 2018	ab693f4c0bc7[.]com
December, 2017	ab70a139cc3a[.]com	December, 2018	ab23660730bca[.]com

Dynamically Decoded Strings

The following are the strings that are dynamically decoded during the malware's execution. It should be noted that each string is promptly zeroed out in memory after use.

SOFTWARE\Piriform\Agomo kernel32.dll IsWow64Process SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall Publisher Microsoft Corporation DisplayName QueryFullProcessImageFileNameA SeDebugPrivilege %u.%u.%u ab%x%x.com speccy.piriform.com