# **DPRK Malware Targeting Security Researchers**

norfolkinfosec.com/dprk-malware-targeting-security-researchers/

#### norfolk

January 26, 2021

Earlier today, Adam Weidemann from Google's Threat Analysis Group (TAG) <u>published</u> <u>research</u> regarding a threat actor targeting security analysts following a social engineering campaign. Google attributes this activity to DPRK threat actors. This blog has no evidence to corroborate or refute this claim, but considers Google to be a reputable source of information.

According to the published research, the threat actors would engage in a social engineering effort in which they would attempt to collaborate with security analysts on a Visual Studio project, ultimately leading to them delivering a malicious DLL that the researcher would unknowingly launch.

This post examines that DLL and parts of its second-stage workflow.

### **Technical Analysis**

MD5: 56018500f73e3f6cf179d3b853c27912 SHA-1: a3060a3efb9ac3da444ef8abc99143293076fe32 SHA-256: 4c3499f3cc4a4fdc7e67417e055891c78540282dccc57e37a01167dfe351b244

This file is a DLL that expects to be executed under the following conditions to initiate the malicious workflow:

- The operating system must be 64-bit
- The correct export must be called
- Exactly two additional command line arguments must be supplied alongside this export

Although Google provided multiple hashes, the file above was selected as a starting point because its exports matched the export shown in an image in Google's post (CMS\_dataFinal). The post used this image how the malware would execute in normal circumstances, which in turn allows us to supply two additional critical parameters to the file:

Bx9yb37GEcJNK6bt
 4901

Under the attacker's workflow, these would have been supplied through a PowerShell command initiated through a Visual Studio Build Event, but these can also be supplied through a normal command line.

Once executed under the specified conditions, the malware will move an encoded set of strings into memory and decode them.

000007FEEA063465	OFB605 04200100	movzx eax, byte ptr ds: [7FEEA075470]		*
000007FEEA06346C 000007FEEA063473	48:8D8D 31210000 0F1005 D61F0100	movups xmm0, xmmword ptr ds: [7FEEA075450]	000007FEEA075450:"A_jvCd%#mUS.a}}!Cq?5	a^Lh8n/t/<6\$="
000007FEEA06347A	0F100D DF1F0100	movups xmm1, xmmword ptr ds: [7FEEA075460]	000007FEEA075460:"q?5a^Lh8n/t/<6\$="	
000007FEEA063483	41:B8 FF030000	mov r8d,3FF		
000007FEEA063489 000007FEEA063493	0F1185 00200000 100271	mov dword ptr ss: rbp+2000, b8/10210 movups xmmword ptr ss: rbp+2100, xmm0		
000007FEEA06349A	0F118D 10210000 C785 04200000 056287	movups xmmword ptr ss: rbp+2110, xmm1		
000007FEEA0634AB	C785 08200000 4D4408	mov dword ptr ss: rbp+2008, 5A08444D		
000007FEEA063485 000007FEEA06348F	66:C785 0C200000 ACE585	mov word ptr ss:[rbp+2000],A3CD		
000007FEEA0634C8 000007FEEA0634D2	C785 90200000 2F0333 C785 94200000 205700	mov dword ptr ss: rbp+2090, BE33032F		
000007FEEA0634DC	C785 98200000 6E6E1C	mov dword ptr ss: rbp+2098,631C6E6E		
000007FEEA0634E6	C785 A0200000 CA567F	mov dword ptr sstrbp+209C,68D0F26E mov dword ptr sstrbp+20A0,CA7F56CA		
000007FEEA0634FA 000007FEEA063504	C785 A4200000 82E58F: C785 A8200000 BF0B530	mov dword ptr ss: rbp+20A4,108FE582 mov dword ptr ss: rbp+20A8,C4530BBF		
000007FEEA06350E	C785 AC200000 308FB3	mov dword ptr ss: rbp+20ACL, ACB3BF30		
000007FEEA063522	C785 B4200000 A70A4C	mov dword ptr ss: rbp+2084,484C0AA7		
000007FEEA06352C 000007FEEA063536	C785 B8200000 1E6C41 C785 BC200000 4C6934	mov dword ptr ss: rbp+2088,98416C1E mov dword ptr ss: rbp+208C,F134694C		
000007FEEA063540	C685 C0200000 A2	mov byte ptr ss: [rbp+2000], A2		
000007FEEA063551	C785 1C200000 2C5FC6	mov dword ptr ss: rbp+201C ,39C65F2C	Encoded Strings	
000007FEEA06355B 000007FEEA063565	C785 20200000 797134 C785 24200000 8BC0E7	mov dword ptr ss: rbp+2020,76347179 mov dword ptr ss: rbp+2024,ASE7C08B	Encouca strings	
000007FEEA06356F	C785 28200000 102C71	mov dword ptr ss: rbp+2028, 87712C10	Pushed to Memory	,
000007FEEA063583	C785 30200000 3A483D	mov dword ptr ss: rbp+2030,243D483A	r usheu to memory	/
000007FEEA06358D 000007FEEA063597	66:C785 38200000 E0A	mov word ptr ss:[rbp+2034],288BC4D7		
000007FEEA0635A0 000007FEEA0635AA	C785 40200000 170333 C785 44200000 2050C6	mov dword ptr sst rbp+2040, BE330317		
000007FEEA0635B4	C785 48200000 6D730A	mov dword ptr ss: rbp+20481,7D0A736D		
000007FEEA0635C8	C785 50200000 E57E4C	mov dword ptr sstrbp+204C,70F7CFA6		
000007FEEA0635D2 000007FEEA0635DC	C785 54200000 C6B6A1 C685 58200000 1A	mov dword ptr ss:[rbp+2054],29A1B6C6 mov byte ptr ss:[rbp+2058],1A		
000007FEEA0635E3	C785 60200000 290333	mov dword ptr sstrbp+2060, BE330329		
000007FEEA0635F7	C785 68200000 6D730A	mov dword ptr ss: rbp+2068,7D0A736D		
000007FEEA063601 000007FEEA06360B	C785 6C200000 A6CFF7 C785 70200000 E57E4C	mov dword ptr ss: rbp+206C,70F7CFA6 mov dword ptr ss: rbp+2070, EA4C7EE5		
000007FEEA063615 000007FEEA06361F	C785 74200000 C686A1 C785 78200000 912248	mov dword ptr ss: rbp+2074, 29A1B6C6		
000007FEEA063629	C785 7C200000 32B2F4	mov dword ptr ss: rbp+207C ,B3F4B232		
000007FEEA06363D	C785 84200000 F34A11	(mov dword ptr ss: rbp+2080, F7F86DAA (mov dword ptr ss: rbp+2084, 6114AF3		
000007FEEA063647 000007FEEA063650	66:C785 88200000 527: C685 8A200000 E3	mov word ptr ss: rbp+20881,7152 mov byte ptr ss: rbp+208A1.E3		
000007FEEA063657 000007EEEA06365D	8885 20210000 40:8885 30210000	mov byte ptr ss: rbp+21201,al		
000007FEEA063664	E8 671F0000	Call Cms2.7FEEA0655D0		
000007FEEA063669	E8 CBFAFFFF	call cms2.7FEEA063140		
000007FEEA063675 000007FEEA06367C	48:8D8D 90200000 E8 BFFAFFFF	lea rcx, gword ptr ss:[rbp+2090] call cms2, 7FEEA063140		
000007FEEA063681	48:8D8D 18200000	lea rcx, qword ptr ss: [rbp+2018]		
000007FEEA06368D	48:8D8D 28200000	lea rcx, gword ptr ss: [rbp+2028]		
000007FEEA063694 000007FEEA063699	48:8D8D 40200000	lea rcx, gword ptr ss: [rbp+2040]		
000007FEEA0636A0 000007FEEA0636A5	E8 98FAFFFF 48:8D8D 60200000	call cms2.7FEEA063140 lea rcx.gword ptr ss:[rbp+2060]		
000007FEEA0636AC	E8 SFFAFFF	call cms2.7FEEA063140		
000007FEEA063681	40:0000 40200000	vor adv adv		*
00000001EEDF0]=8				· ·
0000000122010]-0				
cms2.dll:\$365D #2A5D				
Dumo 3	🕮 Dumo 5 🛛 🚳 Watch	h 1 IX=Liocals 🖇 Struct		0000000001ECBC0 0000
e curp s		ASCII		00000000000000000000000000000000000000
71 DB 05 62 87 68 4D	44 08 5A AC E5 B5 73	qD.b.kMD.Z-áus Bx9yb37GEcJNK6bt		00000000001ECBD8 0000 00000000001ECBE0 0000
00 00 00 00 00 00 00 0E 34 76 88 C0 E7 A5 10	OF 4A B1 2C 5F C6 39 2C 71 B7 15 35 F7 06	yq4v.Ac¥.,q.5÷. p_find.lxUi5CZO		00000000001ECBE8 0000
3D 24 D7 C4 BB 28 E0 33 BE 20 50 C6 33 6D	AA 00 00 00 00 00 00 00 73 0A 70 A5 CF F7 70	:H=\$xA»(a* .3% P\$3ms.)'T+p C:\\ProgramData\		000000000001ECBF8 0000
4C EA C6 B6 A1 29 1A	6F 2B 77 00 00 00 00	à~Lꣶi).o+w	Second ed Stuinge	00000000001ECC00 0000 00000000001ECC08 0000
4C EA C6 B6 A1 29 91	22 4B CD 32 B2 F4 B3	a~Lꢶ;)."KI2*ô"	Jecoded Strings	00000000001ECC10 0000 0000000001ECC18 0000
F8 F7 F3 4A 11 06 52 33 BE 20 57 DD 32 6E	71 E3 00 00 00 00 00 6E 1C 63 BE F2 D0 68	*%0+0JRqa pdate.bin /.3% WY2nn.c%0Dh C:\\Windows\\Sys		00000000001ECC20 0000
7F CA 82 E5 8F 10 BF E4 E3 A7 0A 4C 48 15	0B 53 C4 30 BF B3 AC	ÊV.Ê.âSĂOJ - tem32\\rund1132.		00000000000000000000000000000000000000
OF 00 00 00 00 00 90	7F 30 00 00 00 00 00	¢		00000000001ECC38 0000 00000000001ECC40 0000
30 00 00 00 00 00 00 04		·.0		00000000001ECC48 0000
46 00 00 00 00 00 CD 6A 76 43 64 25 23 60	AF 80 FB FE 07 00 00 55 53 2E 61 7D 21 43	^FI.ûþ A_jvCd%#mUS.a}IC A_jvCd%#mUS.a}IC		00000000001ECC58 0000
35 61 5E 4C 68 38 6E	2F 74 2F 3C 36 24 3D	q?5a^Lh8n/t/<6\$= q?5a^Lh8n/t/<6\$=		00000000000000000000000000000000000000
00 00 00 00 00 00 44		D		00000000001ECC70 0000 00000000001ECC78 0000
00 00 00 00 00 00 00 AE	00 00 00 00 00 00 00 00			00000000001ECC80 0000
39 00 02 00 00 00 80 01 00 00 00 00 00 01	83 30 00 00 00 00 00 00 00 00 00 00 00 00 00			00000000001ECC90 0000
	BA 80 FB FE 07 00 00	°.ûþ	E	00000000000000000000000000000000000000
	FE 1E 00 00 00 00 00			00000000001ECCA8

The malware uses these strings to call the CreateDirectory API at C:\ProgramData\VMWare and later to specify a filename at this location for a dropped payload (vmnat-update.bin). These decoded strings also contain an export and an additional parameter for this dropped payload.

After this, the malware enumerates running processes via NtQuerySystemInformation and checks for the presence of avp.exe (Kaspersky) and avastui.exe (Avast). If either of these two processes are found, the malware will gracefully exit. If not, it continues with its workflow.



and avp.exe and avastui.exe (bottom)

Following this step, the malware decodes a second set of strings. These are used to create persistence via the CurrentVersion\Run key under an entry named "OneDrive Update."

48:8085 28200000 4C:8085 0820000 4C:8085 6020000 48:8095 90200000 48:8095 90200000 48:894724 28 48:894724 28 85:40000 E8 85FAFFF 85C0 0F85 45010000 48:8070 C820000	124 res.quord ptr :::: tp::200 d 124 res.quord ptr :::: tp::200 d 124 rfs.quord ptr :::: tp::200 d 124 rfs.quord ptr :::: tp::200 d 126 rfs.quord ptr :::: tp::200 d 126 rfs.quord ptr :::: tp::200 d 126 rfs.quord ptr ::: tp::200 d 126 rfs.quord ptr ::: tp::200 d 126 rfs.quord ptr ::: tp::200 d 127 rfs.quord ptr ::: tp::200 d 128 rfs.quord ptr ::: tp::200 d 129 rfs.quord ptr ::: tp::2	["\$\$+28]:"4901" ["\$\$+20]:"1xu1\$CZDIV45]89V"			
7785         7200000         33.346/           7785         7200000         38.776/           7785         7200000         86.776/           7785         720000         86.7671           7785         720000         86.7671           7785         720000         86.6761           7785         720000         86.6761           7785         720000         31.3621           7785         720000         31.3621           7785         720000         31.3621           7785         720000         31.3621           7785         720000         58.5643           7785         720000         58.5643           7785         720000         31.3621           7785         720000         31.5721           7785         720000         31.5721           7785         720000         59.7200           7785         720000         59.7200		Additional Encoded Data		Persistence for Dropped Binary	
		[rsp-20],ebx [rsp-20],rax [rsp-20],rax [rsp-20],rax [rsp-20],rax [rsp-20],rax [rsp-20],rax [rsp-20],ebx	[rsp+20]:"C:\//\wind eax:"C:///Windows//	0ws////5ysten32////rund1132.exe C:////ProgramData/// //Systen32////rund1132.exe C:///ProgramData////MM	<pre>\\\\vmat-update.bin,OCSP_resp_find lxuisC20IV45j89V 4901" ware\\\\vmat-update.bin,OCSP_resp_find lxuisC20IV45j89V 4901"</pre>
C685 1020000 0A r E8 3999FF 48:8080 00200000 E8 2079FFF 48:809 6424 30 48:809 6200200 41:85 06000200 43:85 06000200 F8:50 6620000 F8:50 6620000 F8:50 662000 F8:50 66200 F8:50 662000 F8:50 66200 F8:50 66200 F8:50 662000 F8:50	Nov Dreat ptr ::11("process") (************************************	A:'\0' [rsp-20]:")xu1sC20IV45389Y"			
401391438 401391438 431184622450 481189532210000 41189 0300000 41189 0300000 41189 0300000 41189 0300000 8500 8500 8500 75 08 4818842450 75 08 4818842450 75 08 481894250000 4618950210000		[[59-28]] <sup>+4001*</sup> [*59-20] <sup>+1</sup> NUTSCZOTV453899**			
E8 010 8FFF 4C:8005 5A5F0100 48:8015 535F0100 48:80 40 F0 41:89 08800500 E8 440AFFFF BA 50003700 B9 40000000					
Bit Strate         Bit Strate           Strate         Strate           Strat	1         1		,		

Once the malware has created persistence key, it writes a second-stage DLL to the "C:\ProgramData\VMWare\" directory and calls it using the previously decoded parameters.

# Second Stage

Analysis of the second-stage payload is in-flight and additional details beyond what is listed below will be provided when available. It is possible (and perhaps likely, due to time constraints) that another researcher will complete this analysis before I do, in which case those details will be corroborated and added below for completeness, along with the appropriate credit.

MD5 – f5475608c0126582081e29927424f338 SHA-1 – 8e88fd82378794a17a4211fbf2ee2506b9636b02 SHA-256 – a75886b016d84c3eaacaf01a3c61e04953a7a3adf38acf77a4a2e3a8f544f855

The second-stage malware performs a similar command line check to verify that it is running with two supplied parameters. The first of these parameters, IxUi5CZ0IV45j89Y, is used as to create a mutex to ensure that only one copy of the malware is running at a time. If the mutex already exists, the malware will exit.

The malware then resolves a long list of API calls before jumping in to a section in memory. While this list is extensive, they indicate potential functionality, including C2 operations (HttpOpenRequest, HttpAddRequestHeaders, etc) and host-based operations (GetDesktopWindow, WriteFile). The sample examined contains multiple C2 domains and URLs, each of which contain a different endpoint for communication.

hxxps:// codevexillium[.]org/image/download/download.asp hxxps:// www.dronerc[.]it/shop\_testbr/upload/upload.php hxxps:// transplugin[.]io/upload/upload.asp

The malware uses the OpenSSL library and communicates to these endpoints via POST request:



HTTP request (right click and open in new tab to zoom in)

As part of the POST request, the malware transmits the date and time of the malware's execution to the C2 server (encoded in Base64 format). This may be used for additional filtering by the attackers, to ensure that too much time hasn't passed between the malware's execution and communication (which could be indicative of an active debugging/reverse engineering effort).

At this stage, the actions available to the malware as a response to this POST request remain under analysis, although the malware presumably at a minimum provides basic reconnaissance and a channel for command-line execution (either directly or through creation of another payload). These are assumptions based on common malware characteristics, however, and not observed activity.\* (Updated below)

# \*Update (1/26):

Over the course of the last 24 hours, a lot of great research and analysis came to light from various parties. Most notably, I'd like to direct readers to three posts that offer additional context and demonstrate that the final action after this POST request is to download an additional payload onto the disk:

360 Threat Intelligence Center <u>provides additional operational context</u> for these attacks, including social engineering. This may be particularly valuable for threat hunters or threat intelligence practitioners. It also offers more details regarding the POST request and next-stage DLL in similar samples.

Qi'anxin Threat Intelligence Center <u>identified similar activity</u> (and malware) from this adversary in September 2020.

Anheng Threat Intelligence Center <u>provides additional context</u> regarding the social engineering and Visual Studio stages of this attack.