A short journey into DarkVNC attack chain

reaqta.com/2017/11/short-journey-darkvnc/



During an analysis of different <u>remote desktop trojans</u> we came across an interesting attack-chain which leverages an RTF that exploits **CVE-2017-8759** to deliver **DarkVNC**, a malicious version of the well-known *VNC*, designed to silently remote-control a victim.

DarkVNC Attack Chain

The DarkVNC chain as reconstructed by ReaQta-Hive can be seen below:



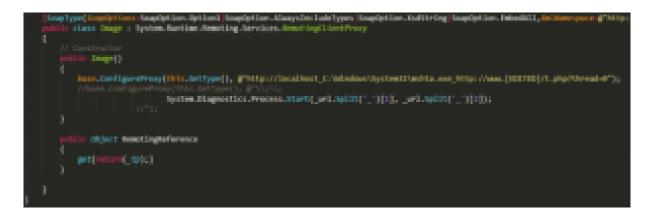
After opening the RTF document, one of the first processes to start is *csc.exe* which is a Command-Line build tool used to invoke the C# compiler, even though csc.exe is a perfectly legit, tool it can be abused for malicious purposes. The first step is to inspect the command-line of csc.exe to discover what is going to be compiled on-the-fly:



cam0snfh.cmdline should raise some suspicion: beside the "weird" name it also run from the user's directory:

```
/t:library /utf8output /R:"System.dll" /R:"System.Runtime.Remoting.dll"
/R:"System.Data.dll" /R:"System.Xml.dll" /R:"System.Web.Services.dll"
/out:"[EDITED].dll" /D:DEBUG /debug+ /optimize-
"C:\Users\User\AppData\Local\Temp\xyzlw5gj.0.cs"
```

The compilation will produce [EDITED].dll (we redacted the name of the DLL for safety reasons since it's in the form: *www.maliciousdomain.com*). To understand what this DLL does, we have to inspect the source-code located in the file *xyzlw5gj.0.cs*:



This block of code takes advantage of the <u>CVE-2017-8759</u> (WSDL Parser Code Injection) that allows an attacker to inject and execute arbitrary code. Specifically the csc.exe generated DLL will be executed by Office. The same technique has been <u>used in the wild to distribute FinSpy</u>. In our case *winword.exe* will finally execute *mshta.exe* that launches an **hta** script which invokes a *powershell*. The main purpose of powershell is to drop and execute **result.exe** whose scope is to deliver **DarkVNC** which we can consider the final payload. The convoluted process described above can be summarized with a simple image that gives us an immediate insight on what is happening on the victim's endpoint:

|--|--|

The Injector

As previously stated, *result.exe* acts as a loader, its goal is to decrypt and inject the malicious DLL that contains DarkVNC. From a static analysis point of view we have the following characteristics:

SHA256: 1D6F4CAC33FFF1B744DCE13BDF003B15D8EABCE53B0578E3B4BDBC5CBF001D78

SHA1:	2BB1BE823ED569EF3DAC008B2FEC4A8D04E46922
MD5:	22E2B492108F9D5517EE52C37912F24D
File size:	551.50 KB (564736 bytes)
File name:	result.exe

File type: Win32 EXE

The executable does not have a *Version Information* and from an initial inspection it's encrypted with some private PE cryptor. We will skip the detailed analysis of the packer and subsequent unpacking steps because we are more interested in the overall behavior. *result.exe* uses several layers of encryption but does not implement complex anti-reverse engineering countermeasures, so the fastest way track the core behavior by setting a breakpoint on *VirtualAlloc()* and following the various layers.

	100402646 FI	93 86 60 4	2.00	call dword ptr dat[sbox-dwirtualAlloc>]						
		A 00		push 0						
		9 06 24		mov dword ptr ss:[esp].ecx						
		1 09		sub ecx.ecx						
		1 64		add ecx, eax						
		5 C 8								
	and the second			nov edi,ecx						
				pop_ecx						
		6.99		push 0						
		9 20 24		moy dword ptr ssifesp],ebp						
		5 60		seb ebp.ebp						
	00402085 0	5 GF		add ebp, edi						
	00402067 8	9 AS 30 CO 4	7 00	mov debrid ptr ds:[ebx+470030],ebp						
	00402060 5	0		pop abp						
	00402065 6	A 00		push 0						
	00402070	9 14 24		mpy dword ptr as:[esp].edx						
	00402071 1	1 62		spr edx.edx						
		i 93 45 CO 4	7.00	or edx.dword ptr ds:[ebx+470045]						
		B #2		nov est.edx						
	00402070 5			pop edx						
		A 00		push 0						
	and the second			mov dword ptr ss:[esp],edi						
		3 11	2	xor edi.edi						
		5 55 C4 C0 4	7 00	or edi, dward ptr ds: [ebo+42c0c4]						
		5 CF		mov ecs, edi						
				pop edi						
	00402C8E F	5 44		repe movsh						
	00402090 5	2		push edx						
			F OF 00	moy dword ptr ss: espl.FFFFF						
	00402098 52	9		DOD HCX						
				push eax						
and the second s	DOCIDIONEUT C		e an on l	moy dword ptr as: eapl.result.40208F						
	00402CAL 5			pop ebx						
				push edi						
	00402643			nov edi.edx						
		6 P.A.		nov our, eux						
	4									
dword ptr Tebx+47c0ec	- DOMESSION AND A DOMESSION	an le deleren		-demoli2.virtualAlloc>						
and a built furgerationer	Telforence our cum	COLUMN STREET, SAME	the traces is	Prove the range of the second strength						
and the second se	and a Charles and a									
.text:00402c4c result	-6X_133CBC #200	£								
		-								
Ell Dump 1 Ell Dump 2	2 🔰 💵 Dump 3	Liu Dump 4	Dum	p S 🛛 🐯 Watch 1 🛛 IX-I Lacais 🛛 🌽 Struct 🖉						
Address hims				10000						
Address Hex				ASCII						
	03 00 00 00 04	33 00 00 FF	FF 00 00	Sec						
00200020 00 00 00 00	00 00 00 00 00									
00200020 00 00 00 00 00 00 00 00 00 00 0	00 00 00 00 00 00	00 00 00 80								
00200020 00 00 00 00 00200020 00 00 00 00 00200040 06 1F BA 06	00 00 00 00 00 00 00 00 00 00 00 00 00 84 09 CD 21	00 00 00 80 85 01 4C CD	00 00 00 21 54 66							
00200020 00 00 00 00 00200020 00 00 00 00 00200040 06 1F BA 06	00 00 00 00 00 00 00 00 00 00 00 00 84 09 CD 21	00 00 00 80		1. program canno						
002200020 00 00 00 00 00 002200030 00 00 00 00 002200040 0E 1F BA 0E 00220050 69 73 20 70	00 00 00 00 00 00 00 00 00 00 00 00 04 09 CD 21 72 6# 67 72 61	00 00 00 80 85 01 4c cp 60 20 63 61	21 54 66	1s program canno						
002200020 00 00 00 00 00 002200000 00 00 00 00 002200040 06 1F 8A 06 00220050 06 1F 8A 06 00220050 77 20 70 00220050 74 20 62 65	00 00 00 00 00 00 00 00 00 00 00 00 84 09 CD 21 72 6# 67 72 61 20 72 75 6# 20	00 00 00 80 88 01 4c CD 60 20 62 61 69 66 20 44	21 54 68 68 68 69	t be run in cos						
002200020 00 00 00 00 00 002200030 00 00 00 00 002200040 06 1F EA 06 002200050 06 73 20 70 00220050 74 20 62 65 00220070 65 6F 64 65	00 00 00 00 00 00 00 00 00 00 00 00 04 09 00 21 72 6F 67 72 61 20 72 5F 62 23 20 72 6B 6A 24	00 00 00 80 88 01 4c cb 60 20 63 61 69 66 20 44 00 00 00 00	21 54 68 68 68 6F 4F 53 20 00 00 60	t be run in cos						
002200020 00 00 00 00 00 002200030 00 00 00 00 002200040 06 3F 8A 06 002200050 69 72 20 70 002200500 69 72 20 70 002200500 69 62 65 002200070 60 6P 68 65 002200080 50 45 00 00	00 00 00 00 00 00 00 00 00 00 00 00 04 09 00 21 172 6# 67 72 61 20 72 75 68 20 18 00 00 04 24 40 01 04 00 15	00 00 00 80 85 01 40 CD 60 20 62 61 69 68 20 44 00 00 00 00 82 80 59 60	21 54 68 66 66 6F 4F 53 20 00 00 00 00 00 00	t be run in pos modeS PE						
002200020 00 00 00 00 00 002200030 00 00 00 00 002200040 06 3F 8A 06 002200050 69 72 20 70 002200500 69 72 20 70 002200500 69 62 65 002200070 60 6P 68 65 002200080 50 45 00 00	00 00 00 00 00 00 00 00 00 00 00 04 09 00 01 72 64 67 72 61 10 72 75 62 01 78 65 60 04 24 42 01 64 00 15 80 03 63 63 68	00 00 00 80 85 01 4c cb 60 20 63 61 60 66 20 46 80 00 00 00 82 80 59 05 81 02 05 65	21 54 68 68 68 6F 4F 53 20 00 00 60	1s program canno t be run in pos mode						

Execution will jump from layer to layer until we reach the last one where it's possible to get the most important aspects of the injector.

10	FR. OFF.	EA 2				_	pank 25	
	10001541	64.9					barth.	
	10001500		1.0	1.50	00.1	10 C	call deord ptr de:(categorinalderwather)	
1.1	10001807	80.0	8				Call desord ptr ch: (categories) dervather]	
1.00	PROPERCY.	- DF 8	ñ e	0.00	00.1	10 C	100014a1	
-	10000		0				parts 40	
	10001011	10					march sam	
	50001012	80 A	4.3	6.38			Tes extudeord ptr soc especial surge small, small	
	10.01616	L (M 5	t è	35			xmrps xmm0, xmm0	
	10001618						Dest eax	
	10001814	F 8 9	P 2	- 44	24.3		heydon semword ptr :::: esp+28, sem0	
10.	10001631	0.0 2	1.1	1.00				
100	10001831	10 B B C	8, 9				and exp.c may depend per scatterproc.,44 Tea was depend per scatterproc.	
-	10001838	C7 4	4.3	1.30	44.1		eav dabrid ptr stilletpr30,44	distant D.
-	30003438	80 A 68 7	4.3				Tea eas, depend ptr 55.5 espa25	
-	30003434	1.10	X 3	. 80	10^{-1}		Dester Indexts./%	<pre>intio00317EntToychest.exe -kT</pre>
100	20003334	- 12 -					pash eax	
10	10001634		H 3	U 80	02.1	0 00	Tes exc,dword ptr std esp+340	
	30003641	- 52 .					peit eas call desrd ptr ds:[edrathcombines;] tel eas, deard ptr dligetp-dc] peth sam	
	10001642		3 P				call exerciptrical: (deathcontrinue)	
100	10001848		8.3	1.10			Tes esc, deard per commerce	
-	10001840	32					puth sai	
	20003848	10		6, 99			Tea eax, doord ptr statespa30	
		1 A A					push sea	
	100155	- Et					push -	
	20001654	64.6	Å,	6.00	100		Dest 4000004	
1	30003854	66.0	6.1				Cuth 0	
1	10001810	5.5 0					Could D	
1.1	10001815						Cush D	
-	10001045	10.1	1. 1	1.98	00.1	o ao i	Tea eax, deard ptr statespi205	
- 2	10001648	10			-		push can	
	10001055	- 64 g	0 - 1				peak -	
			3.6	1.50	00.5	10 C	call deprd ptr do:[Celementerrecenses]	
		- <u>19</u> 3						

The svchost.exe process is created as a suspended process so the malicious code will be executed when the process is finally resumed. At this point we can extract DarkVNC from memory.

The DarkVNC Module

The static inspection of the module's PE shows the following:

Export Directory							
Import Directory	Ordinal	Function RVA	Name Ord	Name RVA	Name	Demangled	Forwarding
Relocation Directory	(n)						
Load Config Directory	1	000075E7	0	00030549	VncStartServer	NA	NA
IAT Directory	2	0000776C	1	00030558	VncStopServer	NA	NA

There are two exports whose meaning is self-explanatory, they are used to manage the VNC Server.

72C81FE3 72C81FE4 72C81FE9 72C81FE9 72C81FEF	50 PUSH EAX 68 02020000 PUSH 202 FF15 <u>1895CA7</u> ; CALL DWORD PTR DS:[<&WS2_32.#115>] 8045 10 LEA EAX.LARG.3]	
72C81FF2	66:8975 FØ MOV WORD PTR SS:[LOCAL.4],SI	
72C81FF6 • 72C81FF7 •	50 PUSH EAX 8D45 F0 LEA EAX,[LOCAL.4]	
72C81FFA	C745 10 1000(MOV DWORD PTR SS:[ARG.3],10	
72C82001 · 72C82002 ·	50 PUSH EAX 53 PUSH EBX	Ar
72C82003 · 72C82004 ·	56 PUSH ESI 68 8497CA72 PUSH OFFSET 72CA9784	Ar
72082009	FF15 1095CA7/CALL_DWORD_PTR_DS:[<&WS2_32.WSAStringToAddressW>]	LWS:



The first step is to convert from string to address the attacker's address, which in this case is in the form *IP:443*.



Obtains the *ComputerName* and an additional identifier in order to assemble the string that will identify the victim's endpoint, the final string will be: (*COMPUTER_NAME*)_*ADDITIONAL_ID*-**DARKVNC**. Immediately after, the VNC Server is started. We will not go through the analysis of the whole module for the sake of brevity, but from the inspection of strings we can speed-up the initial assessment.

75158DAE 61 79	4E 6F 74 69	66 79 57 6E 64 00 00	0 00 53 79 ayNotify	lind Syl
75158DBE 73 58	61 67 65 72	88 88 88 88 56 69 73	3 75 61 6C sPager	Uisgal
7515BDCE 45 66	66 65 63 74	73 88 88 88 29 88 86	0 00 23 68 Effects	• <u>#h</u>
7515800E 76 6E	63 00 00 00	SC 88 2A 88 86 88 86	3 88 2E 88 unc 🕓 1	• •
7515BDEE 00 00	2£ 00 2£ 00	5C 88 2A 88 86 88 86 86 86 86 86 86 86 86 86 86	6 00 60 00	s u l
	107 JULE 708 JULE	HERE REFEIRED FOR THE REFEIRE		

The string **#hvnc** is pretty indicative, this core shares many similarities with HVNC (HiddenVNC) a well-known Remote-Control Module whose source-code can be found in the <u>carberp leak</u>. This module shares with it a large amount of similarities like:

Hidden VNC capabilities: The module will create a *new Window Desktop to keep hidden the malicious VNC instance*. This technique is usually adopted to bypass anti-fraud engines on personal banking websites by impersonating the victim's computer and logging in with stolen credentials without raising alerts on the bank's side. Here's a quick representation of the above behavior taken from ReaQta-Hive's process-tree point of view:



We have a new *explorer.exe* instance and one of the child processes is Chrome! While there are also some basic differences between DarkVNC and HVNC, one of the most interesting is represented by the following:

SetEnvironmentVariableW("MOZ_DISABLE_CONTENT_SANDBOX", "1")

According to the <u>documentation</u> **MOZ_DISABLE_CONTENT_SANDBOX** disables content process sandboxing.

The threat from a higher perspective

So far we have identified the following DarkVNC samples: Collected samples:

- 1. deb02b28605a2b9c80b25c5fa1fa43ac8c71b10961f7517c1a0394531d3b0b40
- 2. 9a57cefbfcdf1b18cc31a2784a2ed3e0e11dd4a3c4608b1243b4141a475b182f
- 3. a67e96b01520183babfae285b5d692b5b3dda7edff7378b281ace7fd381d3c93
- 4. e0a73dd11f0f2c41859bf01cf8a5b7a2a9946303d6e7898f696037323d038f56
- 5. Delivered via Terror EK: http://www.malware-traffic-analysis.net/2017/10/17/index.html

Hashes of the sample analyzed in this post:

- 1. RTF: 7a641c8fa1b7a428bfb66d235064407ab56d119411fbaca6268c8e69696e6729
- 2. result.exe: 1d6f4cac33fff1b744dce13bdf003b15d8eabce53b0578e3b4bdbc5cbf001d78

Detection & Protection

Visibility over the endpoints is essential to quickly detect new threats as they're deployed by the attackers. Real-time behavioral analysis creates a window of opportunity to detect behaviors that are unusual, running VNC or Teamviewer is not a malicious activity by itself but those same tools can be abused to get control over an endpoint. Being capable of detecting such anomalies allows for a timely analysis and response before the severity of the incident escalates.

Check out <u>ReaQta-Hive</u> to understand how an *Endpoint Threat Response* platform can help your organization to secure the infrastructure from threats like the one just analyzed, track incidents and respond in real-time. Anomalous behaviors can be hard to understand manually and the help offered by the algorithms greatly increase the chances of detection and consequently the reaction time.