When Threat Actors Fly Under the Radar: Vatet, PyXie and Defray777

unit42.paloaltonetworks.com/vatet-pyxie-defray777/

Ryan Tracey, Drew Schmitt

November 7, 2020

By Ryan Tracey and Drew Schmitt

November 6, 2020 at 6:15 PM

Category: Malware, Ransomware, Unit 42

Tags: Defray777, PyXie, Vatet



This post is also available in: <u>日本語 (Japanese)</u>

Executive Summary

As security practitioners, we spend a lot of time focusing on the threat actors and malware families that leverage the most impactful exploits or affect the highest number of victims. But what happens when a threat actor goes "low and slow" to fly under the radar? One could argue that, in that situation, the threat actor may end up having more impact than some of the more prolific threat groups.

We first noticed that there may be a relationship between the Vatet loader, PyXie Remote Access Tool (RAT) and Defray777 ransomware when there were remnants and/or detections of all three in various <u>Incident Response</u> and <u>Managed Threat Hunting</u> engagements. After

digging deep into each malware family, it became apparent that Vatet, PyXie and Defray777 are all associated with the same financially motivated threat group that has been operating since as early as 2018.

That threat group, sometimes referred to as <u>PyXie</u> by BlackBerry Cylance and <u>GOLD</u> <u>DUPONT</u> by SecureWorks, has been actively conducting successful ransomware operations that have impacted organizations in a number of sectors including healthcare, education, government and technology while remaining under the radar. This blog aims to shed light on this threat group and to disrupt their operations through awareness of their malware families and operating methodologies. In essence, we want to get them *on* the radar.

During our research, we discovered that this threat group has developed and maintained the Vatet loader. This loader has evolved as this threat group has taken advantage of multiple open source tools by altering the original application to execute payloads such as PyXie and/or Cobalt Strike. Next, the threat group uses a tailored version of PyXie, which we call PyXie Lite, to conduct reconnaissance and to find and exfiltrate files that are likely sensitive to the victim organization. In a number of incidents we investigated, the actors established an initial foothold into the victim's network through common banking trojans such as IcedID or Trickbot. From there, they deployed Vatet, PyXie and Cobalt Strike before executing Defray777 ransomware entirely in memory. This results in encrypted files on local drives and file shares before exiting. Additionally, the ransomware leaves no evidence of execution except for the encrypted files and ransom notes. In regard to Defray777, the group behind this malware has also ported their ransomware from Windows to Linux, something that, before Defray777, has yet to be seen in the targeted ransomware space. Before this discovery, ransomware that had the ability to impact both Windows and Linux systems was limited to cross-functional ransomware written in Java or scripting languages such as Python. With the port to Linux, Defray777 ransomware has become the first ransomware variant to have standalone executables for Windows and Linux.

With three different malware families to cover, we realize there is a lot of content to digest. We have a lot of great details on each of these, but we also realize that you may be interested in one malware family over the others, or you may just prefer to choose your own adventure. If desired, use the links below to skip to the malware family that interests you most, or to get right to the IOCs that will get you hunting for, and detecting, this threat group in action.

Table of Contents

- First Up: Vatet Loader
- Next Up: PyXie Lite
- Last, but Not Least: Defray777
- Linking Vatet, PyXie and Defray777
- Indicators of Compromise (IOCs)

First Up: Vatet Loader

Vatet is a custom loader that executes XOR encoded shellcode from the local disk or a network share. The loaders are typically open source applications found on GitHub, or other repositories, that the actors modify to load their shellcode. In most cases, the payload winds up being Cobalt Strike beacons and/or stagers, but some of the more recent payloads have been an updated version of the PyXie RAT. Vatet is often a precursor to enterprise-wide ransomware attacks.

<u>Microsoft wrote about the Vatet loader in April 2020</u> and said the loader had been in use as early as November 2018 for the purpose of loading Cobalt Strike into memory for execution. This loader continues to be seen in the wild using multiple versions of open source applications to load shellcode including:

Version	First Seen
Recompiled Tetris game	2019-06-28
Recompiled Notepad	2020-05-03
Recompiled desktop customization app, Rainmeter	2020-06-24
Recompiled Notepad++	2020-09-24

Table 1. Vatet versions.

In our research, we have seen Vatet samples with compile times as early as 2019, although this variant has implemented several variations since then.

In the earliest versions of Vatet that we analyzed, the malicious payload was loaded via a network share using a path with the following format: **{IP}\{EPOCHTIME}\{PAYLOAD}**.dat. However, in the latest samples analyzed, the malicious payload was loaded locally from disk. Additionally, we have seen variations in the XOR keys used to decode the payload during execution time. Our research also determined that the Vatet loader has expanded its payload capabilities to load PyXie in addition to the previously seen Cobalt Strike beacons and stagers. Finally, the Vatet loaders we analyzed have evolved and begun taking steps to improve their anti-forensics capabilities by deleting malicious payloads after they have been loaded into memory for execution.



Figure 1. Vatet execution flow. Let's take a deeper look at Vatet using a malicious version of Rainmeter.

Inner Workings of the Vatet Loader: A Rainmeter Review

Rainmeter is a desktop customization tool that allows users to customize their desktops through the use of "skins." During a legitimate installation, Rainmeter creates an executable, rainmeter.exe, and a corresponding DLL, rainmeter.dll. Under normal conditions, rainmeter.dll is responsible for reading configuration files and facilitating a customized desktop. Under the observed circumstances, a signed, legitimate version of rainmeter.exe and a malicious version of rainmeter.dll could be simply copied onto the victim system, then used to load and execute a Cobalt Strike beacon in memory under the context of a signed, legitimate executable.

Taking a Look at the Static Properties

We first reviewed the suspicious rainmeter.exe and rainmeter.dll files and compared them to versions that would be installed on a system through the official September 2019 release of the Rainmeter installer, which can be found on its <u>public GitHub page</u>.

Reviewing rainmeter.exe did not produce many interesting findings. Examining both executables in <u>PEStudio</u> confirmed that the sample recovered during a ransomware scenario was the same executable generated by the standard Rainmeter installer, based on the SHA256 hash. We also verified that both executables had the same valid digital signature.

gestudio 9.05 - Malware Initial Assessment - www.w	initor.com [c:\cases\archive	\legit_rainmeter\rainmeter.exe] — 🗆 🗙	Z pestudio 9.05 - Malware Initial Assessment - www.w	initor.com [c:\cases\archive\	\options_1\rainmeter.exe] - 🗆 🗙
file settings about			file settings about		
₩ E X B ?			₩ E × E 💡		
	property md5 sha1 sha256 md5-without-overlay sha1-without-overlay sha256-without-overlay first-bytes-hex file-size size-without-overlay entropy imphash signature entry-point file-version description file-type cpu subsystem compler-stamp resources-stamp	value 204E002E10571660C.08C00E200E28EEA 49900C.0A4E001545130009559118E34002E07FE 36670451702E351820293A3E031001F055A652647.443E6C3852657E085663 101a 101a 40 554 90 00 50 00 00 00 00 00 00 00 00 FF FF 00 00 80 00 00 00 00 00 00 00 00 40 00 475356 (bytes) 101a 2532 9A66E55F22017F3D79503820AF36E546 101a 101a 2532 9A66E55F22017F3D79503820AF36E546 101a 2532 9A66E55F22017F3D79503820AF36E546 101a 2532 9A66E55F22017F3D79503820AF36E546 101a 2532 9A66E55F22017F3D79503820AF36E546 101a 2532 9A65E57F2017F3D79503820AF36E546 101a 101a 101a 101a 101a 101a 101a 101a 101b 101b 101b 101b 101b 10	Classes and how options. Tyteinmeter and An indicators (220) Winated (disabled) Societad (disabled) Soci	property md5 sha1 sha256 md5-without-overlay sha1-without-overlay sha256-without-overlay first-bytes-hes first-bytes-text file-size size-without-overlay entropy imphash signature entry-point file-version description file-type cpu subsystem compile-stamp debugger-stamp resources-stamp	value yelstpozz 10571666C C66C0E2DDE28EEA 49800C-0A4EDD154153D2006550110E14003EETF 366704370CE35188293A3ED11D01F055AE5549243EC2852657ED05563 1/4 1/4 1/4 1/4 4D 54 90 00 50 00 00 00 00 00 0F FF 00 00 B8 00 00 00 00 00 00 00 00 00 00 00 00 00
	version-stamp	empty		version-stamp	empty
	certificate-stamp	0x98C04000 (Tue Feb 06 18:00:00 2018)	1	certificate-stamp	0x98C04000 (Tue Feb 06 18:00:00 2018)

Figure 2. Initial comparison of static properties of "rainmeter.exe".

Comparing the rainmeter.dll samples provided more interesting findings. Initially, it was obvious that the two samples were not the same, since the hashes did not line up. The sizes of the files were significantly different from one another and the compile dates were also quite different. Additionally, there was some variability in the imports, exports, strings and other properties. Further, the suspected malicious DLL was not digitally signed and had additional sections not present in the legitimate Rainmeter DLL.

ive\legit_rainmeter\rainmeter.dll	roperty	value	C:\cases\archive\options_2\rainmeter.dll	property	value
iit)	45	READA7CCRA0210E52074287C4EE6A1E7	Jul indicators (wait)	mdS	32DAE47577CDA09DEC93E65E1317CREE
)	105	DDD702222E02D29E29D1D02C20A0EECCA55D54D	···• virustotal (disabled)	cha1	235/AE47577CE/A0057C62263E1217C6EE
31	101	9EEC A3ADC16CE8AE41BD846E5E132D28DDEC05088D76CE828DC363DACC	> dos-header (64 bytes)	sha256	47D6CC0A05218D0C1078DA8E8D0CA7878424CDD73
	d5-without-overlay	wait	dos-stub (wait)	md5-without-overlay	wsit
	al-without-overlay	wait	> file-header (Jul.2020)	sha1-without-overlay	wait
4	a256-without-overlay	wait	optional-header (GUI)	sha256-without-overlay	wait
fi	rst-hytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FE FE 00 00 B8 00 00 00 00 00 00 00 40 00	directories (time-stamp)	first-bytes-bex	4D 5A 90 00 03 00 00 00 04 00 00 00 FE FE 00 00 B8 00 0
fi	rst-bytes-text	M7	Sections (blacklist)	first-bytes-text	M7
fi	le-size	1805200 (bytes)	imports (wait)	file-size	1882112 (bytes)
si	ze-without-overlay	wait	exports (duplicated)	size-without-overlay	wait
	ntropy	6.656		entropy	6.648
in	nphash	n/a	resources (12)	imphash	n/a
si	onature	n/a	abc strings (wait)	signature	n/a
er	ntry-point	55 8B EC 83 7D 0C 01 75 05 E8 24 06 00 00 FF 75 10 FF 75 0C FF 75 08 E8 B3 F		entry-point	55 8B EC 83 7D 0C 01 75 05 E8 28 06 00 00 FF 75 10 FF
fil	le-version	4.3.1.3321	,🗊 manifest (aslnvoker)	file-version	4.4.0.3321
d	escription	n/a	in version (Rainmeter.dll)	description	n/a
fil	le-type	dynamic-link-library	🖼 certificate (n/a)	file-type	dynamic-link-library
c c	pu	32-bit	· 🗋 overlay (wait)	cpu	32-bit
SL	ubsystem	GUI		subsystem	GUI
cc	ompiler-stamp	0x5D875A7D (Sun Sep 22 06:26:53 2019 - UTC)		compiler-stamp	0x5F01CBA2 (Sun Jul 05 07:46:26 2020 - UTC)
d	ebugger-stamp	0x5D875A7D (Sun Sep 22 06:26:53 2019)		debugger-stamp	0x5F01CBA2 (Sun Jul 05 07:46:26 2020)
re	sources-stamp	empty		resources-stamp	empty
0	orts-stamp	0xFFFFFFFF (Sun Feb 07 00:28:15 2106)		exports-stamp	0xFFFFFFFF (Sun Feb 07 00:28:15 2106)
ve	ersion-stamp	empty		version-stamp	empty
	ertificate-stamp	0x98C04000 (Tue Feb 06 18:00:00 2018)		certificate-stamp	n/a

file settings about								
₩ 🛛 X 🖹 💡								
□-III c:\cases\archive\legit_rainmeter\rainmeter.dll ∧	property	value	value	value	value	value		
Jul indicators (wait)	name	.text	.data	.idata	.rsrc	.reloc		
··· 🔪 virustotal (disabled)	md5	F678288D9143CE9EFA0DC6D	37C65C7D43ACDAF4B7AF14	BB6CDF86E9CE020B5D4FAF	8B6652E00914EC8B1B81C8C	045CE2F2AC7399BA8D77C8		
> dos-header (b4 bytes)	entropy	6.592	4.706	5.590	3.436	6.702		
dos-stub (216 bytes)	file-ratio (99.47%)	94.11 %	1.13 %	0.60 %	0.37 %	3.26 %		
mile-neader (Sep.2019)	raw-address	0x00000400	0x0019F000	0x001A4000	0x001A6A00	0x001A8400		
directories (time stamp)	raw-size (1795584 bytes)	0x0019EC00 (1698816 bytes)	0x00005000 (20480 bytes)	0x00002A00 (10752 bytes)	0x00001A00 (6656 bytes)	0x0000E600 (58880 bytes)		
h rections (99.47%)	virtual-address	0x10001000	0x101A0000	0x101AB000	0x101AE000	0x101B0000		
libraries (7/23)	virtual-size (1816940 bytes)	0x0019EBD8 (1698776 bytes)	0x0000A888 (43144 bytes)	0x000028B4 (10420 bytes)	0x00001850 (6224 bytes)	0x0000E408 (58376 bytes)		
imports (108/385)	entry-point	0x00168F66						
	writable	•	x		÷			
	executable	x						
	shareable							
abc strings (174/12666) V	discardable					х		
< >>	initialized-data	-	x	x	X	x		
pestudio 9.05 - Malware Initial Assessment - www.w	initor.com [c:\cases\archive\op	otions_2\rainmeter.dll]						- 0
file settings about								
⊯∃ X8 ?								
□-== c:\cases\archive\options_2\rainmeter.dll	property	value	value	value	value	value	value	value
	name	.text	.data	.idata	testdata	testvers	rsrc	.reloc
···• virustotal (disabled)	md5	EB344977B74F4A7404B8A76	7D6B2388B6464B72948B5CB	CE28C601240E2AD7B6F8CA	3B564C2CF07F7B243BEC045	8D5DC1EA2D1C955D1E8791	93D3D25BB577B5876E2B801	CC68CD0ACDE43A5B18297
> dos-header (64 bytes)	entropy	6.596	4.647	5.612	2.134	0.020	3.433	6.693
dos-stub (208 bytes)	file-ratio (99.95%)	94.61 %	1.14 %	0.57 %	0.03 %	0.03 %	0.35 %	3.21 %
The neader (Jul.2020)	raw-address	0x00000400	0x001B3000	0x001B8400	0x001BAE00	0x001BB000	0x001BB200	0x001BCC00
directories (time stame)	raw-size (1881088 bytes)	0x001B2C00 (1780736 bytes)	0x00005400 (21504 bytes)	0x00002A00 (10752 bytes)	0x00000200 (512 bytes)	0x00000200 (512 bytes)	0x00001A00 (6656 bytes)	0x0000EC00 (60416 bytes)
h rections (blacklist)	virtual-address	0x10001000	0x101B4000	0x101BF000	0x101C2000	0x101C3000	0x101C4000	0x101C6000
libraries (7/24)	virtual-size (1901624 bytes)	0x001B2B42 (1780546 bytes)	0x0000AC84 (44164 bytes)	0x00002926 (10534 bytes)	0x000000C0 (192 bytes)	0x00000004 (4 bytes)	0x00001850 (6224 bytes)	0x0000EA38 (59960 bytes)
imports (108/394)	entry-point	0x0017A822			-		-	
exports (duplicated)	writable		x					
tis-callbacks (n/a)	executable	x						
resources (12)	shareable				x	x		
string and (175 (12005)								

Figure 4. Comparing sections between "rainmeter.dll" samples.

It is important to note that the <u>code base for Rainmeter</u> is publicly available on GitHub under the GNU General Public License v2.0. This would have allowed the threat actor to openly review/modify the existing rainmeter.dll file contents and compile it into the suspected malicious DLL we saw during our investigation.

After completing these comparisons to confirm that the Rainmeter DLL was likely malicious, it was time for a deeper and more focused look at the samples using a debugger for dynamic analysis.

Dynamic Analysis of the Malicious Rainmeter Sample

Now that we had identified samples for deeper inspection, we stopped the comparisons to the legitimate Rainmeter application and focused on the analysis of the suspicious samples recovered. We placed the samples of rainmeter.exe and rainmeter.dll recovered from the investigation into our analysis environment and began debugging Rainmeter. Shortly after starting analysis, rainmeter.exe loaded rainmeter.dll as expected, and subsequently called its ordinal 1 exported function. Continuing the execution, there were calls to CreateFileA, where the sample was looking for the hardcoded path C:\Windows\help\options.dat.



Figure 5. Call to "CreateFileA" for a hardcoded path.

After the call to CreateFileA, there is a comparison of the result of the call to CreateFileA to FFFFFFF to determine if it has a valid handle to the file or not. If there is no valid handle, the program exits.

Originally it was not obvious that options.dat was necessary for the analysis of the malicious Rainmeter sample as .dat files are not part of the normal Rainmeter application. However, a version of options.dat was recovered in order to continue analysis. Once the "dat" file was placed in the expected location, the program then allocated space on the heap and read the contents of options.dat into memory. After the contents of options.dat were read into memory, the sample performed a first-level decoding of the contents by XOR-ing the contents with the value FE.

🕷 Rainmeter.exe - PID: 5E4 -	- Module: rainmeter.dll - Thread: Main Thread	1514 - x32dbg [Elevated]
FILE VIEW DEBUG	TRACE PLUGINS FAVOURITES OPTIC	ONS HELP JUN 4 2020
ii ← ≡ c' 	🕇 🏊 🐲 🌲 🏦 📓	🥒 🚍 🛷 🥓 j× # 🗛 🎚 🗐 🌻
🖾 CPU 🏾 🍄 Graph	🍃 Log 📑 Notes 📍 Breakpoints	🛲 Memory Map 🗐 Call Stack 😪 SEH 🧕 Script 🎽 Symbols 🗠 Si
	71121189 53 7112118A FF7424 24 7112118 FF7424 24 7112118 FF15 64F12271 711211C4 85C0 711211C5 884C24 10 711211C6 33C0 711211C7 33C0 711211C8 885C9 71121100 Y 74 09 71121102 803418 FE 71121105 40 71121107 3BC1 71121109 Y 72 F7	<pre>push ebx push dword ptr ss:[esp+24] call dword ptr ds:[<&ReadFile>] test eax,eax je rainmeter.711211E7 mov ecx,dword ptr ss:[esp+10] xor eax,eax test ecx,ecx je rainmeter.711211DB xor byte ptr ds:[eax+ebx],FE inc eax cmp eax,ecx jb rainmeter.711211D2 </pre>

Figure 6. Initial XOR decoding loop.

Once the first decoding routine is completed, the malicious Rainmeter application closes the handle to options.dat. When the program closes the handle to options.dat, it is removed from the file system. This is a built-in anti-analysis technique employed to hinder recovery of the .dat file for analysis. At this point, the data read into the program was still a blob of unrecognizable code. However, at the end of the XOR decoding routine, there is a CALL EBX instruction that transfers execution to the recently decoded data. Following EBX in the disassembled view shows that this is valid code. At this stage of analysis, Rainmeter has decoded its options.dat payload, loaded it into memory and executed it. Future analysis confirmed that this was the end of the Vatet loader routine, and execution was passed to the Cobalt Strike shellcode loader.

🕷 Rai	€ Rainmeter.exe - PID: 698 - Thread: Main Thread 1F2C - x32dbg [Elevated]																								
FILE	VIE	w	DEBUG	TR/	ACE	PLUGIN	IS F	AVOUR	RITES	OPTIC	DNS	HELP	JUN 4	4 2020											
-	ອ		→ I	1	• 🏘	-		1	÷≞		1	= 4	× 🥠			۵.	9								
<u></u> 0	PU	?				<u>i</u>										-	o		🔮 Symbols						•
EBX				 05 05 05 05 05 05 05 05 05 	60004 60004 60005 60005 60005 60005	8 9 5 A F	FC E8 10 EA 32 BE 42 05 A 27 5 F	00000 2335B 75C36 A122F	DO 65 A30 FD EB	02	cld call jmp mov add daa	<mark>56000</mark> far D2 esi,FD eax,EB	5E A3:659 365C47 2F12A	5 8333 ; 7 A	2		Desti esi:&	nation «L"C:\\U	of CALL aft sers\\5ynax	er initial \\AppData\	l XOR d ∖\Roami	decoding rout ing\\Rainmete	ne ·\\Raiı	nmeter.ini	

Figure 7. Transfer of execution to valid code after XOR decoding.

By this point, we realized that the Vatet loading mechanism was completed, but we wanted to validate the identity of the final payload, so we pressed on. Further along in the execution, there is a second decoding routine where an additional dynamic XOR loop is used to decode and rewrite the contents of the executable code. If this routine looks familiar, it's probably because you are noticing the Cobalt Strike decoding mechanism. This routine begins by obtaining a pointer to the first four bytes of the imported executable code and setting it as the starting XOR key. The code then executes a loop acting on four bytes at a time, XORing the imported code with the starting XOR key. Next, the loop writes the XOR'd value back into the data segment, followed by setting a new XOR key. The new XOR key is determined by XOR'ing the current XOR key with the value decoded by the current key. Once this loop is finished, the sample then uses JMP ECX to transfer execution to the recently decoded executable contents.

🕷 Rainmeter.exe - PID: 698 - Thread: Main Thread 1F2C - x32dbg [Elevated]

FILE VI	IEW	DEBUG	т	RACE	P	LUGIN	S F	AVOUR	ITES	OP	TIONS	HEL	P	JUN 4 2	2020								
e 🔊		→ 1	I	•	A	*	\$	•	÷₹	8	/	• =	41	11				۵,		9			
🔛 CPU	2	Graph) Lo	og	1	Notes	•	Brea	akpoint	ts I	🋲 Me	mory №	1ap	٦	Call St	ack	~ 2	SEH	C	Scrip	t (📔 Symł
			-• (• (0560 0560 0560	005E 0060 0061	Ť	EB 27 5F 8B07	,			jm po mo	p <mark>5600</mark> p edi v eax.	0087 . dwor	d ptr	ds:	[edi]							
			•	05 60 05 60 05 60 05 60	0063 0066 0068 006A		83C7 8B2F 31C5 83C7	04 04			ad mo xo ad	d edi, v ebp, r ebp, d edi,	4 dwor eax 4	d ptr	ds:	[edi]				Star	ting	XOR K	ey
EIP			•	05 60 05 60	006D 006E 0070		8B1F 31C3				mov xoi	v ebx,	dwor eax	d ptr	ds:	[edi]				Dyna	mic X	DR de	coding
201				05 600 05 600	0072 0074 0076 0075 007C 007E 0080 0080 0085 0085 0087 0092 0092 0092 0092 0092 0092 0092 009	~ ~	891F 8307 8320 8307 8320 8307 8320 8307 8320 8307 8027 8027 8027 8027 8027 8027 8027 80	04 04 2 2 3 3 40 5 3 000000 3 0 4 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	FF 8880 00 3C 52 A2 58 41	5 E	mo add sul xol cm je jm po cm sbi cm cm sbi cm cm cm cm cm cm cm cm cm cm cm cm cm	v dwor r eax, d edi, b ebp, 56000 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5600 5700 5600 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700 5700	d pt ebx yebx b84 b84 b84 b84 b84 b84 b84 b84	ds:[ptr s 044B tr ds tr ds 2:F2E F5F2 d ptr	[edi eax- s:[ed ::	7FC7E bp+5A x-14] x+13] 6 [A1F5 [A1F5	; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	, dh]		call esi:	\$0 &L"C:	\\Use	rs\\Syt
ebx=0			•	560	00007		A1 60	ESEE/	1		moy		dwor	d ote	de	EA1EE	526P	1		es i.	al C.	(\USE	- S \ \S y
dword ptr	[edi]]=[056	50009	C]=	44B35	FC																	
Dump	1	🛄 Dun	np 2	1	Dur	mp 3	.0	Dump	4	100	Dump 5	6) Wa	tch 1	[x =	Loca	ls	2	Struct				
Address	Hex										ASC	II											
05 600048 05 600058 05 600058 05 600078 05 600078 05 600088 05 600088 05 600088 05 600008 05 600008 05 600008 05 600018 05 600118 05 600128	FC E8 36 FD 31 C5 04 F2 00 00 6A 13 F2 F5 D1 33 CF 4B 88 F8 01 35 72 44	10 00 05 A/ 83 C7 ED 04 FF Ff 00 58 FC ED A1 66 16 E9 38 80 2E A1 87 87 88 85 16 25 84 85 16 25 84 85 17 65	0 00 A 12 7 04 4 31 5 38 5 FC 9 9 A 3 FC 9 9 A 5 4C 0 EC 1 97 5 37 5 20 8 C 4	00 2F 57 B0 35 A6 F5 72 6A 17 84 13 24	EA 32 EB 27 SB 1F 39 DD F1 1A 48 04 5E BB A1 68 A1 68 A16 A16 A16 A16 A16 A16 A16 A16 A16 A16	33 5F 31 74 38 F2 6 F2 0A 44 D6 F2 75 DF A8	5B 65 8B 07 C3 89 02 EB 80 F2 BC AE A2 5E F5 A1 F5 A1 F5 A1 EA F2 A6 2F 2B 9C 2D DE BB F8 A9 90 C5 80 C5	A3 D 83 C 1F 3 EA 5 64 5 68 F 68 F 68 4 3F 2 21 3 4A D 7E 0 7E 0 7C E CA B	2 BE 7 04 1 D8 8 FF D 5A A EC 2 F5 2 F5 2 F5 8 EE 5 6D C 3A F 76 C 6D E 18 3 14	47 5 88 2 E0 E E8 0 2F 8 A1 0 84 F F0 1 F5 F 37 4 9B 4 C6 E	GC Ше. 2F Gý. 7 1A. 88 88 900 Öÿÿ 35 58 j 58 j 58 j 55 òõi 56 òõi 57 \$	<pre></pre>	23[ef(_1Å (t.ëê) 8.ò.M @¼º ò¢^»(òčiki öiki #êò?? Öi/!! "#êò?? Öi/!! "#êò?? Öi/!! "#êò?? Öi/!! "#ôï#	0%G\ C/ 10.C Xÿàė MZè. jì/. bõįk bõįk bõįk Hî4n %m.û <:0. Bvöó .m7C î0									

Figure 8. Entering into the second decoding loop. Note the memory space in Dump 1.

🕷 Ra	inmeter	r.exe	- PID	: 698	- Tł	nread	l: Mai	n Thr	ead	1F2C	- x3	2dbg	[Elev	ated]																			
FILE	VIE	W	DEB	UG	Т	RACE	F	PLUGI	NS	FA	VOUF	UTES	c	OPTIC	ONS	H	IELP	ງເ	JN 4	2020														
-	ອ 🛯		÷	Ш		•	A	-	- 4		•	÷£		8	1	Ę	- 4	>	//				₽,		9									
32	CPU	-	Gra) La	og	1					eakpoi										9	SEH		o S		P						. R€
812-				→ →		55 600 155	0005 E 000 60 000 60 000 60 000 60 000 60 000 60 000 60 000 60 000 72 000 72 000 70 000 72 000 70 7	~	EBFB838B311357 8833113374 833113374 EB85FE8859255 8911383311374 EB85FE8859255 89157 FF00000000000000000000000000000000000	27 07 2F 2F 2F 2C 2F 2C 27 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	94 94 94 94 95 9000 9000 9000	FF 38800 400 00000 56 00			jmp popp add mov xor add puss mov xor cmp sbb pop pop mov cal pop puss inc cmp sbb cal puss mov cal add cmp sbb cal add add add add add add add add add a	56 ed ed ebb ebb ebb ebb ebb ebb ebb ebb e	0008 ii, dw ii, dw	ptr pr pr pr pr pr pr pr pr pr p	ds:[ds: ds: ds: ds: ds: ds: ds:	eax eax eax eax	[ed [ed],et ,et	i] i] i) x],dh		Sta Dyn Tra cal	urtin masfe 1 \$0	r of	R Key deco exec	, uti	g on to	decc	ded e	execut	table
eax=	05 6000	94																																
05 60	0085					_																												
1	Dump 1	ę	Щ. І	Dump	2	i.	. Du			i	Dump		i i i i i i i i i i i i i i i i i i i				60						2											
Addr	ess H	ex .		00	00	00.0	0.5			5.0	45 5		0.55		ASCI	II	F 01		_															
05 600 05 600	00A4 C 00B4 0 00C4 0 00C4 8 00C4 8 00C4 8 00C4 8 00C4 8 00C4 3 0114 7 0114 7 0124 3 0124 3 0124 8 0124 8 0154 8 0154 8 0164 9 0164 9	Image: Solution Solution 3 500 Solution 10 57 00 00 10 57 00 00 12 18 87 00 50 11 87 60 50 94 40 11 83 75 50 88 75 50 </td <td>81 FF 000 95 76 05 69 69 63 56 56 84 56 84 56 84 56 84 56 84 56 84 85 85 85 85 85 85 85 85 85 85</td> <td>00 00 00 05 C4 37 B9 3D 39 25 00 51 2D 50 50</td> <td>00 00 99 36 91 84 EF 8F EF AEF CS F DAEF</td> <td>50 00 00 00 00 00 00 00 00 00 00 00 00 0</td> <td>00 50 50 00 00 00 00 00</td> <td>8 F0 0 00 0 00 8 9D 8 FF 7 D3 4 58 6 BF 7 D3 4 58 6 BF 7 D3 8 57 6 B2 9 37 6 B2 9 37 6 A 21 6 A 21 7 A 21 6 A 21 7 A 21 7 7 7 7 7 7 7 7 7 7 7 7 7</td> <td>B5 00 BC BA 30 5C 26 07 18 F1 9F</td> <td>A2 00 00 B0 39 45 55 D1 3F 28 E8 39 FC 77</td> <td>+56 00 00 F 000 F CC E 34 3 5E 9 F3 E 9A 8 BD E 6D E 6D E 6D 2 6D 2 6D 2 6D 2 6D 2 6D 2 6D 2 6D 2</td> <td>5 8 0 8 0 8 0 8 0 8 0 8 0 8 0 8 0</td> <td>4 00 0 00 5 54 4 2D 1 F7 A 3C F A3 F C9 E E7 3 11 0 D0 2 00 2 00 5 54 4 2D 1 F7 A 3C F A3 B3 F C9 E 7 1 1 0 00 2 00 0 00 0 00 0 00 0 00 0 00</td> <td>00 00 16 95 85 0A 6C 95 DE AE 7B 10</td> <td>AP AP WÿE B P B Q. Q. Q. Q. Q. Q. Q. Q. Q.</td> <td>Ý 46¥ 7 = 6Â ∧ 20È 14 14</td> <td>. [.157 Óhoµ '</td> <td>¢Vh. • Ìèc 94:(E^./ Uóà: ? (½) èmÜ 9ÿ ü.t</td> <td>eT. ∂ A÷μ :<. ?£1 ÿÉÞ 4Ç⁰ </td> <td></td>	81 FF 000 95 76 05 69 69 63 56 56 84 56 84 56 84 56 84 56 84 56 84 85 85 85 85 85 85 85 85 85 85	00 00 00 05 C4 37 B9 3D 39 25 00 51 2D 50 50	00 00 99 36 91 84 EF 8F EF AEF CS F DAEF	50 00 00 00 00 00 00 00 00 00 00 00 00 0	00 50 50 00 00 00	8 F0 0 00 0 00 8 9D 8 FF 7 D3 4 58 6 BF 7 D3 4 58 6 BF 7 D3 8 57 6 B2 9 37 6 B2 9 37 6 A 21 6 A 21 7 A 21 6 A 21 7 A 21 7 7 7 7 7 7 7 7 7 7 7 7 7	B5 00 BC BA 30 5C 26 07 18 F1 9F	A2 00 00 B0 39 45 55 D1 3F 28 E8 39 FC 77	+56 00 00 F 000 F CC E 34 3 5E 9 F3 E 9A 8 BD E 6D E 6D E 6D 2 6D 2 6D 2 6D 2 6D 2 6D 2 6D 2 6D 2	5 8 0 8 0 8 0 8 0 8 0 8 0 8 0 8 0	4 00 0 00 5 54 4 2D 1 F7 A 3C F A3 F C9 E E7 3 11 0 D0 2 00 2 00 5 54 4 2D 1 F7 A 3C F A3 B3 F C9 E 7 1 1 0 00 2 00 0 00 0 00 0 00 0 00 0 00	00 00 16 95 85 0A 6C 95 DE AE 7B 10	AP AP WÿE B P B Q. Q. Q. Q. Q. Q. Q. Q. Q.	Ý 46¥ 7 = 6Â ∧ 20È 14 14	. [.157 Óhoµ '	¢Vh. • Ìèc 94:(E^./ Uóà: ? (½) èmÜ 9ÿ ü.t	eT. ∂ A÷μ :<. ?£1 ÿÉÞ 4Ç⁰ 															

Figure 9. After the completion of the second decoding routine, note the executable contents now decoded in Dump 2.

At this stage of analysis, we confirmed the content included in options.dat was shellcode that was later decoded via a dynamic XOR routine to create executable code in Rainmeter's process memory.

Now that we had the executable contents of the XOR'd executable code from options.dat available in memory, we dumped the contents from the memory map section in <u>x64bdg</u> for additional analysis to determine this code's potential functionality.

Moving to our dumped sample of the executable code, we conducted an analysis of the strings to determine if there was anything obvious to correlate dynamic analysis findings. In doing this, we identified a reference to beacon.dll, which is most often associated with the DLL version of Cobalt Strike's beacon. Additionally, loading the isolated PE into PeStudio showed the following references to an exported function _ReflectiveLoader@4, which is a known exported function of Cobalt Strike.

Xii ?		
:\cases\archive\options_2\rainmeter	property	value
indicators (5/29)	md5	459D38FDF4A448BBC062F516B5C778F4
virustotal (disabled)	sha1	EB80AB3CDDDDA233A077AC9A294C6AA7F43397F4
dos-header (64 bytes)	sha256	BDA65B4BA61404CB7FBBBB70B4404460723DB17D835A5692333012DC97BB7ECB
dos-stub (184 bytes)	md5-without-overlay	59289EF96738F25F4C2BE93EEFC76955
file-header (number-of-symbols)	sha1-without-overlay	F3076E32B061D87CB346B6E69E5F0C041B88CD4D
optional-header (GUI)	sha256-without-overlay	ED8F92D699311B776990D9C61F3C0050BF0048B693A6489CAA1AD1F3730E31C4
directories (5)	first-bytes-hex	4D 5A E8 00 00 00 00 5B 89 DF 52 45 55 89 E5 81 C3 50 81 00 00 FF D3 68 F0 B5 A2 56 68 04 00 00 00
libraries (A)	first-bytes-text	MZ[REUPhVh
imports (204)	file-size	216940 (bytes)
exports (ReflectiveLoader@4)	size-without-overlay	208896 (bytes)
allbacks (n/a)	entropy	6.693
sources (n/a)	imphash	n/a
gs (13/1454)	signature	n/a
ug (n/a)	entry-point	8B FF 55 8B EC 83 7D 0C 01 75 05 E8 C4 6C 00 00 FF 75 08 8B 4D 10 8B 55 0C E8 EC FE FF FF 59 5D 0
anifest (n/a)	file-version	n/a
rsion (n/a)	description	n/a
tificate (n/a)	file-type	dynamic-link-library
lay (unknown)	cpu	32-bit
	subsystem	GUI
	compiler-stamp	0x58266B7B (Fri Nov 11 19:08:11 2016 - UTC)
	debugger-stamp	n/a
	resources-stamp	n/a ·
	exports-stamp	0x5DE8F170 (Thu Dec 05 06:00:48 2019)
	version-stamp	n/a
	certificate-stamp	n/a

sha256: BDA65848A61404CB/FB88B/084404400/23DB17063JA30523330120C37001202 Figure 10. Extracted PE analysis in PeStudio.

To confirm whether the extracted payload was a Cobalt Strike beacon or not, we utilized a Cobalt Strike beacon parser, which dumped the beacon's decoded configuration.

•		Beacon Dumper
Θ	BEACON METADATA	
~	VERSION	10
	EXPORT TIMESTAMP	The OF Dee 2010 19:00:49 LTC
	EXPORT_TIMESTAMP	
-	EXPORT_NAME	_ReflectiveLoader@4
Θ	BEACON_SETTINGS	
	SETTING_PROTOCOL	HTTPS Beacon (windows/beacon_https/reverse_https)
	SETTING PORT	443
	SETTING SLEEPTIME	15000
	SETTING MAXGET	1048576
	SETTING UTTER	
	SETTING_JITTER	
	SETTING_MAXDINS	
	SETTING_PUBKEY	MIGTMAUGCSQGSIb3DQEBAQUAA4GNADCBIQKBgQDC142R2XDk0t41BaAng7kggQbQ2gKtt9J0HunWGbbHc2dWd32moqfQurJ53NsjMvGrDkwXGokAV2GaGnCCb1GHK1NigibuBcoke5seiXnny94nDmeEu4eEdY
	SETTING_DOMAINS	192.169.7.160,/s/ret=nb_sb_noss_1/167-3294888-0262949/tield-keywords=books
	SETTING_USERAGENT	Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
	SETTING_SUBMITURI	/N4215/adj/amzn.us.sr.aps
	SETTING_C2_RECOVER	bytearray(b'\x04')
		Accept: */*
		Host: www.amazon.com
		session-token=
	SETTING_C2_REQUEST	skin=noskin;
		csm-hit=s-24KU11BB82RASYGJ3BDK 1419899032996
		Cookie
		Accept: */*
		Content-Type: text/xml
		X-requested-with: XMLHttpkequest
		Host: www.amazon.com
	SETTING_C2_POSTREQ	
		00=00=150-8859-1;
		SII e_2117
	DEPRECATED_SETTING_SPAWNTO	
	SETTING_SPAWNTO_X86	%windir%lsyswow64\WerFault.exe
	SETTING SPAWNTO X64	%windir%\sysnative\WerFault.exe
	SETTING PIPENAME	
	SETTING CRYPTO SCHEME	
	SETTING DNS IDLE	
	SETTING_DINS_SLEEP	
	SETTING_CZ_VERB_GET	
	SETTING_C2_VERB_POST	POST
	SETTING_C2_CHUNK_POST	0
	SETTING_WATERMARK	0x12345678 (305419896)
	SETTING_CLEANUP	1
	SETTING_CFG_CAUTION	0
	Setting_ID_54	
	Setting_ID_50	1
	SETTING_PROXY_BEHAVIOR	PROXY_PRECONFIG
	Setting ID 55	
	SETTING KILL DATE	
	SETTING GARGIE NOOK	154122
	SETTING GARGLE SECTIONS	
	SETTING DROCINU DEDME	ev nalizzi krziwradnintwzkrańwa kraiwa kraiwa kraika protoka protoka protoka protoka krziwa kraika krziwa kraika krziwa krz
	SETTING_PROCINJ_PERMS_I	
	SETTING_PROCINJ_PERMS	32
	SETTING_PROCINJ_MINALLOC	1/500
	SETTING_PROCINJ_TRANSFORM_X86	pAtesush(p./xn5/xan/xan.)

Figure 11. Cobalt Strike beacon configuration.

The confirmed Cobalt Strike beacon shows a typical implementation of Cobalt Strike's HTTPS beacon using malleable C2 profiles. Specifically, the <u>Amazon browsing traffic profile</u> created by harmjoy was used in this beacon.

Continue reading: Next Up: "PyXie Lite"

Get updates from Palo Alto Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

By submitting this form, you agree to our <u>Terms of Use</u> and acknowledge our <u>Privacy</u> <u>Statement</u>.