Spot the Difference

An Analysis of the New LODEINFO Campaign by Earth Kasha

Hiroaki Hara / Masaoki Shoji /
Yuka Higashi / Vickie Su / Nick Dai

TREND MICRO

Authors

Hiroaki Hara

Masaoki Shoji

Yuka Higashi

Vickie Su

Nick Dai

Table of Contents

Introduction

Earth Kasha

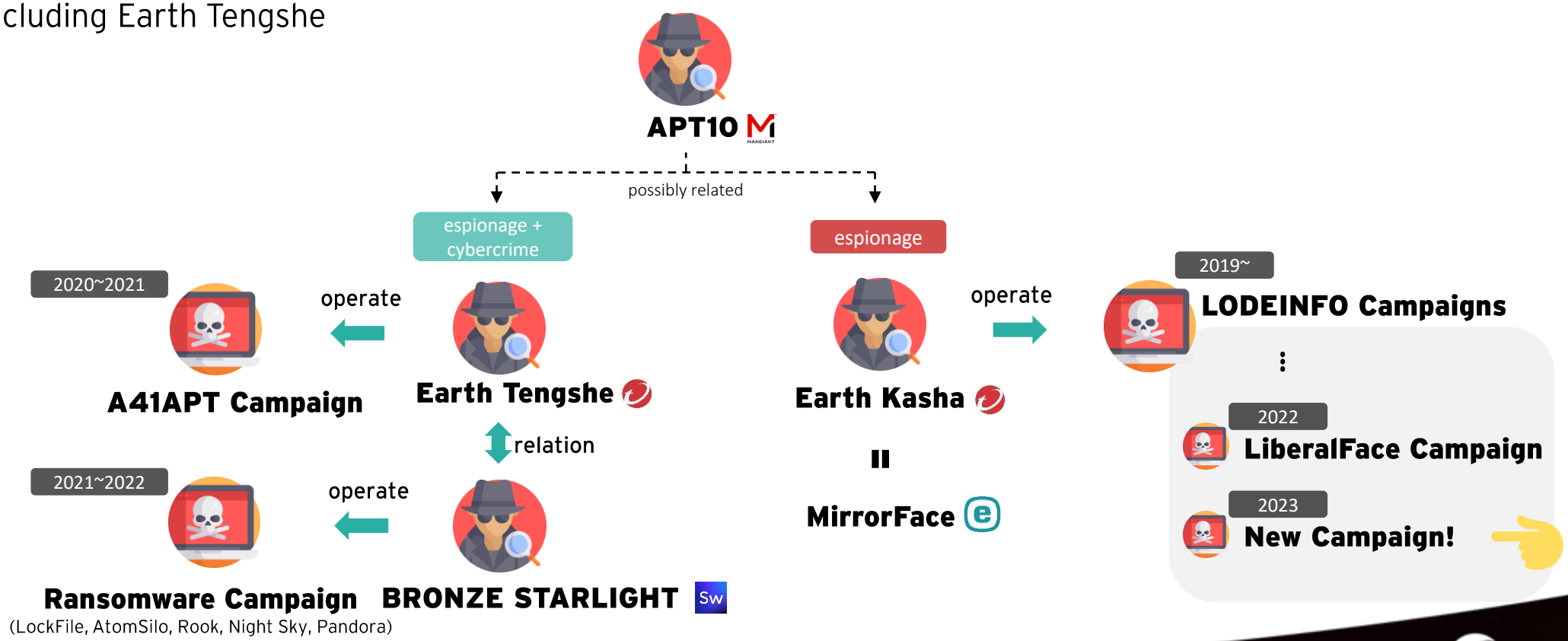
- Trend Micro's official name of Intrusion Set using LODEINFO since 2019
- Until early 2023, Earth Kasha had been employing Spear-phishing email campaign to deliver LODEINFO targeting Japan
- Since early 2023, we have observed a new campaign with different TTPs and toolsets

火車 (Kasha)

https://en.wikipedia.org/wiki/Kasha_(folklore)#/media/File:SekienKasha.jpg

APT10 Umbrella

- Earth Kasha (= LODEINFO actor) has been reported to have connections with APT10 by Kaspersky and Macnica
- We believe Earth Kasha is not exactly same as the known APT10, but possibly belongs to "APT10 Umbrella" including Earth Tengshe



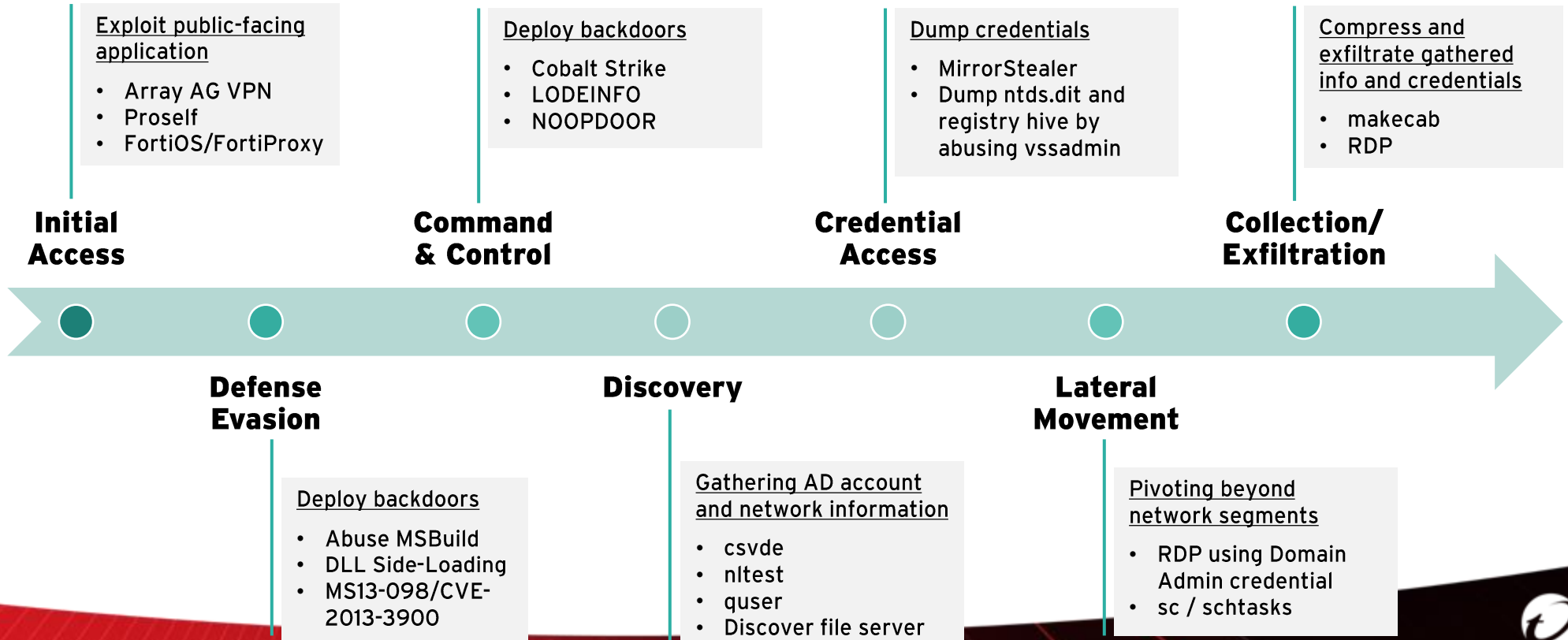
The New Campaign by Earth Kasha

LODEINFO, Long (known) Story Short

- LODEINFO is a notable backdoor exclusively utilized by Earth Kasha, and is continuing to be actively developed
- As far as observed until early 2023, in all incidents, LODEINFO had been distributed using Spear-phishing email

New Campaign!

- Significant change on Initial Access that exploited public-facing application
- Targeting organizations related to advanced technology and government agencies in Japan, Taiwan and India since early 2023

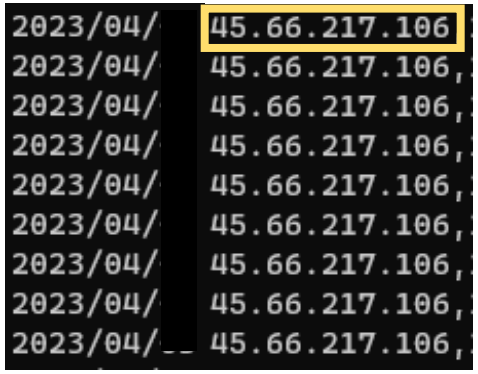


Initial Access

- Initial Access
- Defense Evasion
- Command and Control
- Discovery
- Credential Access
- Lateral Movement
- Collection / Exfiltration

- Exploited public-facing applications for Initial Access
  - SSL-VPN: Array AG (CVE-2023-28461), FortiOS/FortiProxy (CVE-2023-27997)
  - Online Storage: Proself (CVE-2023-45727)
- JPCERT has also released an alert about abusing these products
  - https://www.jpcert.or.jp/at/2023/at230029.html

access log to Array AG



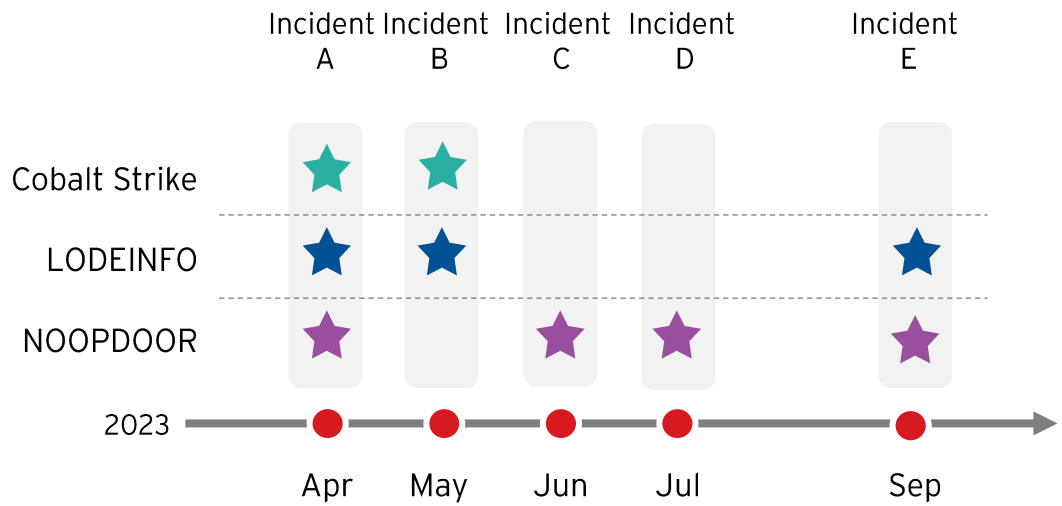
attacker's IP shared by JPCERT

攻撃時期：2023年3月以降
インディケータ情報：不正アクセス元のIPアドレス
- 154[.]31[.]112[.]129
- 173[.]249[.]201[.]243
- 35[.]229[.]146[.]251
- 45[.]32[.]49[.]130
- 45[.]32[.]252[.]239
- 45[.]66[.]217[.]106
- 45[.]32[.]33[.]120
- 95[.]85[.]91[.]15
- 139[.]162[.]127[.]90
- 167[.]71[.]207[.]51
- 176[.]97[.]70[.]81
- 194[.]102[.]36[.]128

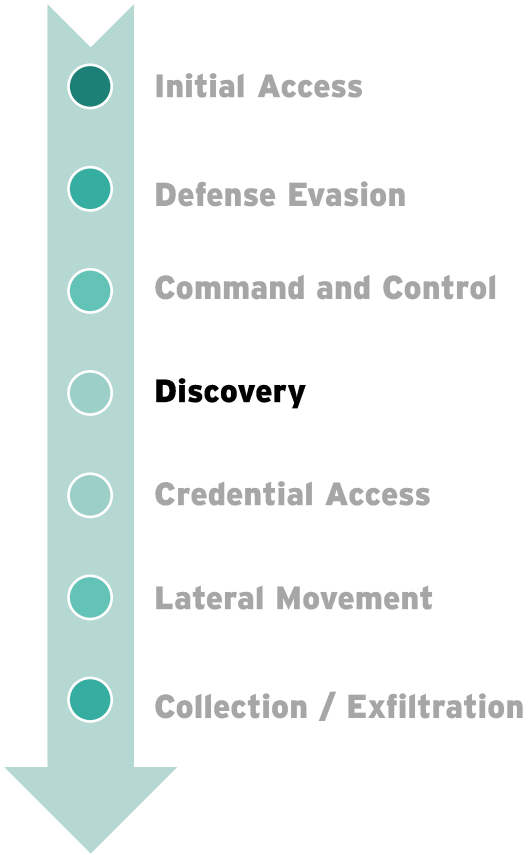
Command and Control

- As the payload, we observed 3 backdoors
  - Cobalt Strike
  - LODEINFO (v0.6.9, v0.7.1, v0.7.2 and v0.7.3)
  - NOOPDOOR NEW

Observed set of backdoors



Discovery



- Abuse legitimate tools shipped by MS for gathering AD network info

```
> csvde.exe  -f all.csv –u
> nltest.exe  /domain_trust
> quser.exe
```

- Discover file server to find system info

```
> dir \\<FILESERVER>\ /s
> copy \\<FILESERVER>\Network Diagram.xlsx
C:\Users\<USER>\Desktop\Network Diagram.xlsx
```

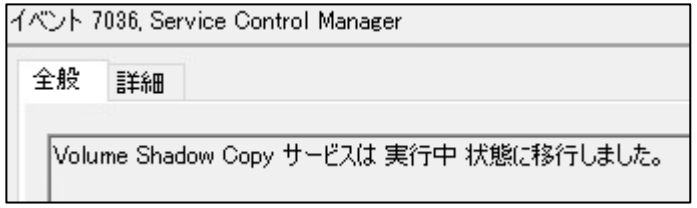
filenames are just an example, not actually observed one in incident

Credential Access (1)

- Initial Access
- Defense Evasion
- Command and Control
- Discovery
- **Credential Access**
- Lateral Movement
- Collection / Exfiltration

- In AD server, dump SYSTEM hive and ntds.dit by abusing vssadmin

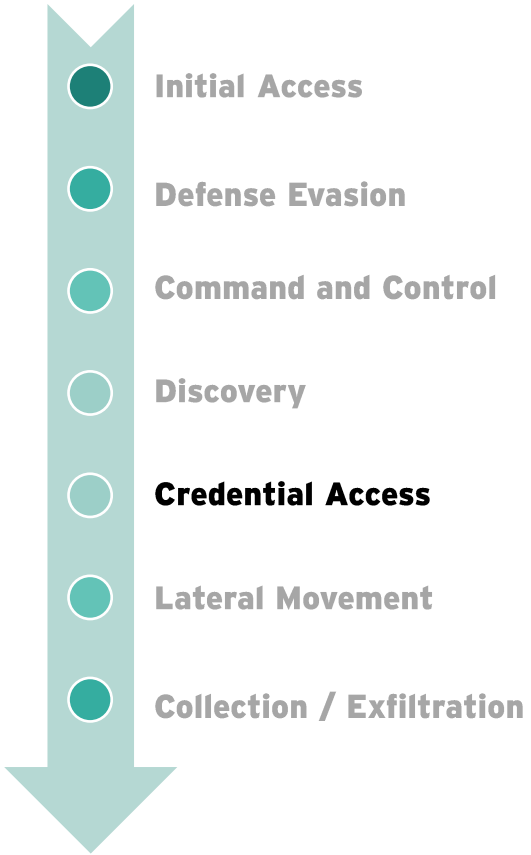
1. create ShadowCopy by vssadmin



2. Copy ntds.dit and SYSTEM hive from AD server to another machine

```
> copy  \\<DC_IP>\c$\windows\temp\ntds.dit .
> copy  \\<DC_IP>\c$\windows\temp\system .
```


Credential Access (2)



MirrorStealer

- Multi-purpose credential stealer originally documented by ESET
- Target applications
  - Saved credentials in browsers (Chrome, Firefox, Edge, IE)
  - Saved credentials in Email client (Outlook, Thunderbird, Becky, Live Mail)
  - Saved credentials in Group Policy Preferences
  - Recent accessed server and saved credentials in SQL Server Management Studio (mru.dat, SqlStudio.bin)
- Stolen information will be saved in `%temp%\31558.TXT`

target application class

```
.?AVCBecky2@@
.?AVCCredit@@
.?AVCGpp@@
.?AVCIE@@
.?AVCLiveMail@@
.?AVCMozilla@@
.?AVCMssql@@
.?AVCOutlook@@
.?AVCVault@@
```


Lateral Movement

- Initial Access
- Defense Evasion
- Command and Control
- Discovery
- Credential Access
- **Lateral Movement**
- Collection / Exfiltration

- Copy components to the target machine over admin share and create scheduled task or service by schtasks.exe and sc.exe

Copy LODEINFO components and create scheduled task

```
> copy SfsDllSample.exe \\<IP>\c$\windows\temp\SfsDllSample.exe
> copy SfsDll32.dll     \\<IP>\c$\windows\temp\SfsDll32.dll
> copy vcruntime140.dll \\<IP>\c$\windows\temp\vcruntime140.dll
```

Copy NOOPDOOR components and create scheduled task

```
> copy mssitlb.xml  \\<IP>\C$\Windows\system32\UIAnimation.xml
> copy ShiftJIS.dat \\<IP>\C$\Windows\system32\ComputerToastIcon.contrast-white.dat
```

```
<Location>\Microsoft\Windows\Media Center\PBDADiscoveryW3</Location>
<ItemName></ItemName>
<LaunchString>"C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe" C:\Windows\system32\UIAnimation.xml</LaunchString>
```


Collection / Exfiltration

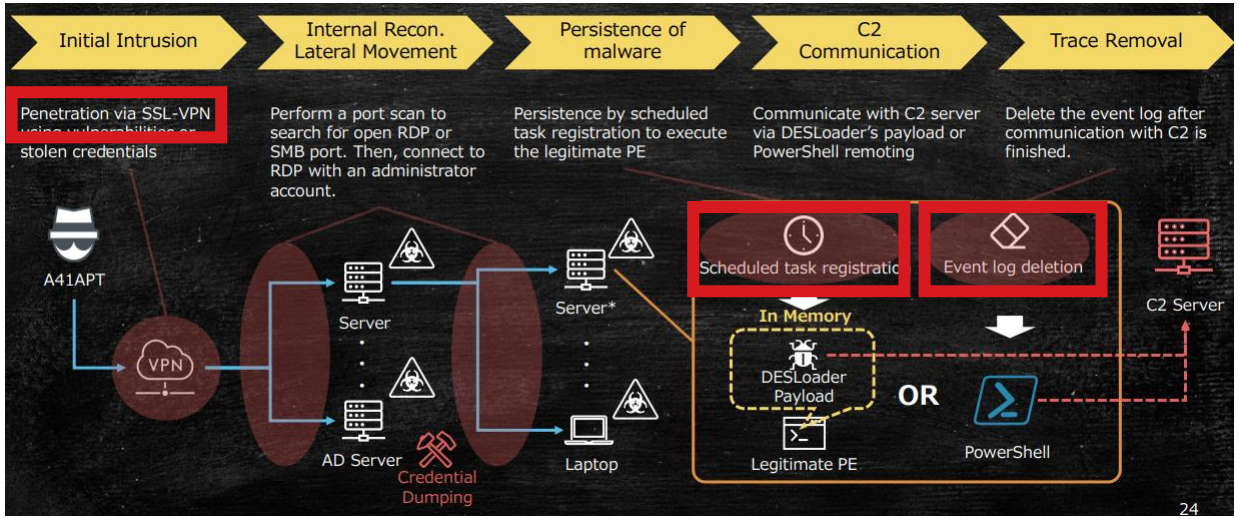
- Gathering information, including ntds.dit, SYSTEM hive and other data, in single victim machine and compress them by makecab

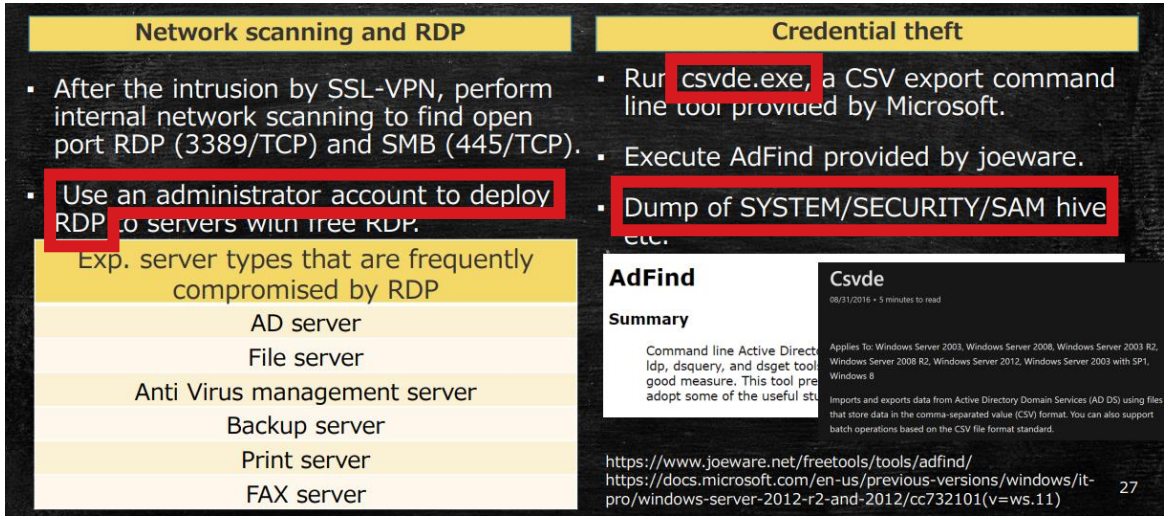
```
> makecab  system
> makecab  ntds.dit
> makecab  people.txt
```

- Exfiltrate interesting files and archives over RDP (tsclient is a source host connecting RDP)

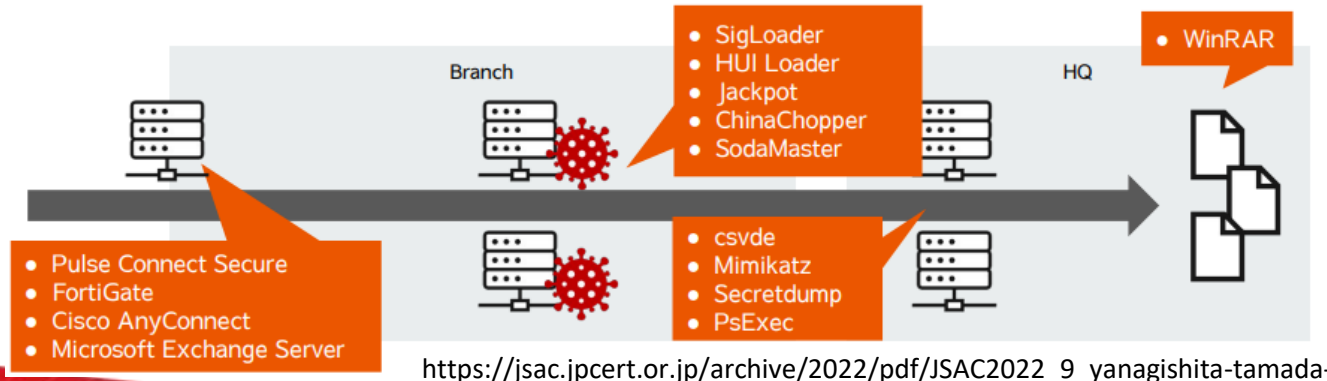
```
\\tsclient\C\aaa\All PC List.xlsx
\\tsclient\C\aaa\All IP List.xlsx
\\tsclient\C\aaa\Network Diagram.xlsx
```


TTPs Overlaps in the Known A41APT Campaign





https://jsac.jpcert.or.jp/archive/2021/pdf/JSAC2021_202_niwa-yanagishita_en.pdf



https://jsac.jpcert.or.jp/archive/2022/pdf/JSAC2022_9_yanagishita-tamada-nakatsuru-ishimaru_en.pdf

Spot the Difference of Campaigns

	A41APT Campaign	LODEINFO Campaign #1	LODEINFO Campaign #2
Attribution	Earth Tengshe	Earth Kasha	Earth Kasha
Timeline	2020 ~ 2021	2019 ~	2023 ~
Region	Japan, Taiwan, Thailand and United States (but main target is the entity in Japan)	Japan	Japan, Taiwan and India
Industry	private sector including electronics, energy, automotive, defense industries	public sector, individuals associated international affairs, politicians and researchers in academic sector	• private sector including manufacturing and aviation • Hi-tech related organizations • government agencies
TTPs	• Exploit public-facing application • DLL Side-Loading • MS13-098/CVE-2013-3900 to embed encrypted payload	• Spear-phishing email • DLL Side-Loading • MS13-098/CVE-2013-3900 to embed encrypted payload	• Exploit public-facing application • DLL Side-Loading • MS13-098/CVE-2013-3900 to embed encrypted payload
Tools	• SigLoader • HUI Loader • SodaMaster • P8Rat • FYAnti • Cobalt Strike • Jackpot	• LODEINFO • NOOPDOOR • DOWNIISSA • Lilim RAT • MirrorStealer	• LODEINFO • NOOPDOOR • Cobalt Strike • MirrorStealer

Malware Analysis in the New Campaign

TREND MICRO

Malwares

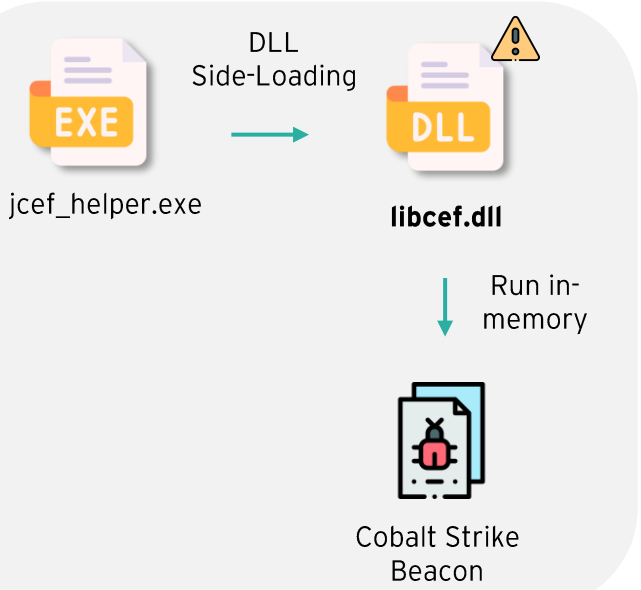
- Shellcode Loaders
  - GOSICLOADER / BINSCOUT / POWERPROVIDER
- LODEINFOLDR / LODEINFO
- NOOPLDR / NOOPDOOR

Shellcode Loaders

- We have observed several types of shellcode loaders

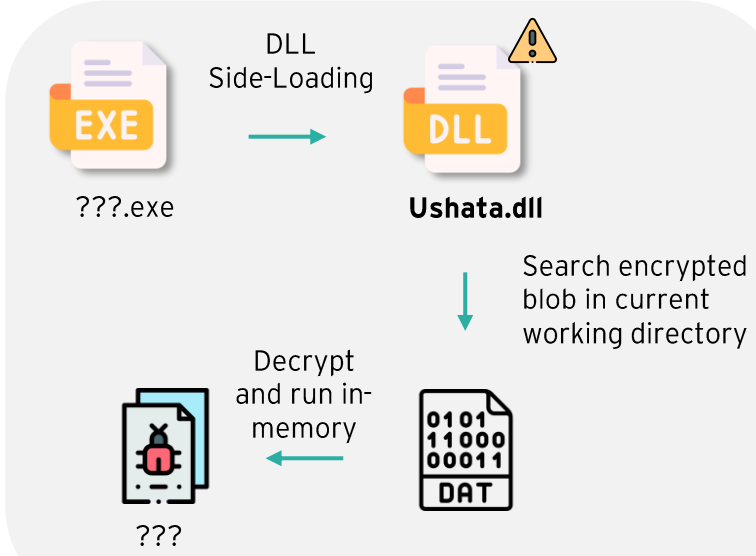
GOSICLOADER

- written in Go
- Decrypt the embedded payload by Base64+AES



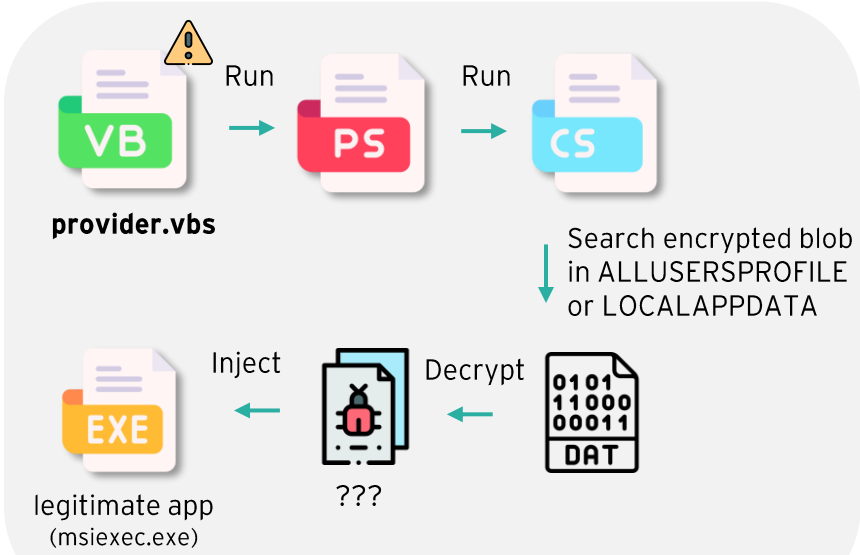
BINSCOUT

- written in C, partially obfuscated
- Decrypt the encrypted payload in CWD by custom algorithm
- Use CRC32 checksum to verify



POWERPROVIDER

- VBS -> PS -> C#
- Decrypt the encrypted payload in another directory by RC4
- Delete "Windows PowerShell" event log



Cracked version of Cobalt Strike

- Based on the embedded watermark, it could possible be a cracked version of Cobalt Strike aka CSAgent, which is shared among Chinese-speaking hacking community

Config in Cobalt Strike

```
Watermark_Hash                  - MYhXSMGVvcr7PtOTMdABvA==
Watermark                       - 666666
```

CSAgent config described in the hacking community blog in Chinese

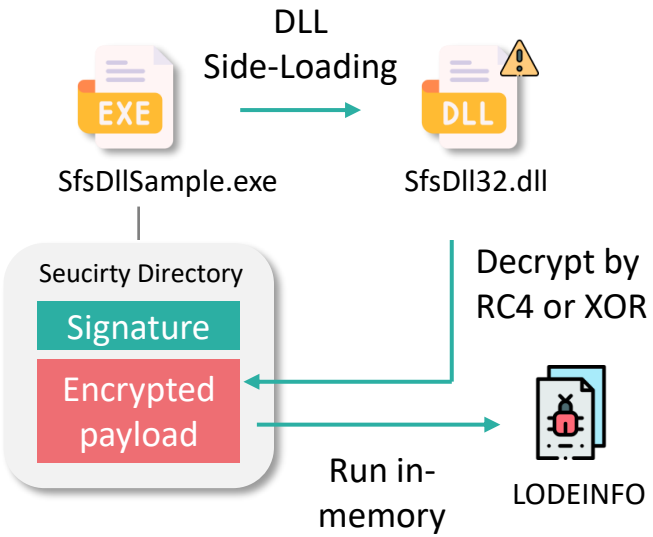
另外，CSAgent这个版本更新了配置方式，现在将配置统一放到CSAgent.properties文件里

```
sleeved.decryption.key=f38eb3d1a335b252b58bc2acde81b542
beacon.watermark.value=666666
beacon.watermark.hash=MYhXSMGVvcr7PtOTMdABvA==
# 翻译开关，是否启用翻译功能，值：Y/N
need.translation=Y
# 是否只使用翻译功能，用于加载自己魔改过的版本，值：Y/N
translation.only=N
```

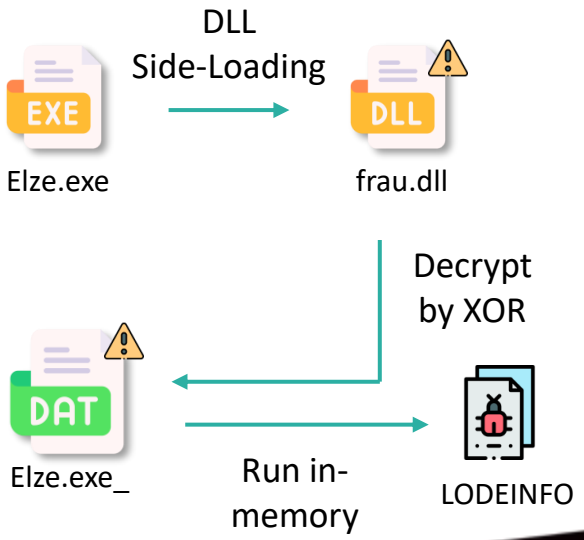

LODEINFOLDR Type 2 (aka FaceLoader)

- LODEINFO observed in Campaign #2 is loaded by LODEINFOLDR Type 2
- Type 2 is different from Type 1 (in Campaign #1) on how to embed and decrypt LODEINFO
- Type 2 decrypts an encrypted payload embedded in a parent process' digital signature abusing MS13-098/CVE-2013-3900

LODEINFOLDR Type 2 loading process



LODEINFOLDR Type 1 loading process




```
cmp     dword ptr [esi+ebx], IMAGE_NT_SIGNATURE
jnz     loc_1000ADA5
mov     ecx, [esi+ebx+98h] ; ecx = RVA of Security Directory
add     ecx, ebx           ; ecx = offset to Secutiry Directory
mov     edx, [ecx]         ; edx = size of signature
mov     eax, [edx+ecx-4] ; eax = offset to encrypted payload
sub     edx, eax           ; edx = size of added data behind legitimate sig
lea     ebx, [eax+ecx]  ; ebx = calc offset to encrypted payload
lea     ecx, [edx-4]    ; ecx = calc actual size of payload by -4 which is a size of offset to payload
mov     [ebp+p_enc], ebx
movzx   eax, byte ptr [ecx+ebx-1]
sub     ecx, eax
mov     eax, [ebp+var_10]
push    ecx
mov     [eax], ecx
call    call_new
```

Offset	Name	Value
28200	Length	28E98
28204	Revision	200
28206	Type	2
28208	Cert. Content	8230, 2A42, 906,...

0x28200

add

0x4238

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00051050	67	2B	99	A9	AA	63	82	96	6C	D1	8E	18	E9	19	3C	C2	g+™©ªc,–lÑŽ.é.<Â
00051060	EE	A5	E0	39	9E	81	0F	62	A0	FF	A0	72	80	DB	6C	03	î¥à9ž..b ÿ r€Ûl.
00051070	95	A4	4A	BD	C9	9A	D4	0B	1D	AF	65	D2	4A	2B	49	80	•¤J½ÉšÔ..¯eÒJ+I€
00051080	49	E6	51	40	A4	DE	DC	8F	C7	41	76	E4	18	C5	F7	64	IæQ@¤ÞÜ.ÇAvä.Å÷d
00051090	98	03	03	03	38	42	00	00									˜...8B..

0x2C438

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0002C3F0	EA	5A	AF	18	80	1E	8C	B8	05	69	B5	A3	9F	C5	91	EC	êZ¯.€.Œ¸.iµ£ŸÅ‘ì
0002C400	8D	04	3D	C0	97	B2	6F	B5	DD	5D	54	64	97	4A	B7	18	.=À—²oµÝ]Td—J·.
0002C410	72	10	42										64	52	90	59	r.Bz..VÁ™±OÌdR.Y
0002C420	1B	E9	38	87	9B	60	45	56	96	7F	35	7A	CB	E7	42	98	.é8‡›`EV–.5zËçB˜
0002C430	D4	7B	38	50	F9	65	00	00	44	99	58	44	91	B4	34	8A	Ô{8Pùe..D™XD‘´4Š
0002C440	AE	E7	24	F3	39	F9	95	ED	D3	7E	76	7E	56	BC	7B	69	®ç$ó9ù•íÓ~v~V¼{i
0002C450	65	25	BB	C9	3							0	72	6F	63	C7	e%»É9&W»E.ProcÇE
0002C460	94	65	73	73	20	C7	45	98	25	75	20	65	C7	45	9C	78	”ess ÇE˜%u eÇEœx
0002C470	69	74	20	C7	45	A0	77	69	FA	29	B9	CB	ED	9C	97	36	it ÇE wiú)¹Ëíœ—6
0002C480	2D	B9	63	23	92	1D	2C	EE	4D	A9	D3	31	95	D4	85	7E	-¹c#’.,îM©Ó1•Ô…~
0002C490	EE	CD	6D	60	AE	75	D5	39	38	E5	16	7D	D9	7C	3F	FD	îÍm`®uÕ98å.}Ù|?ý
0002C4A0	1A	F3	E0	75									BE	18	8C	1F	4E	.óàuˆ,ÆÇ.4á¾.Œ.N
0002C4B0	7D	54	33	39	8E	87	98	DC	C9	34	4E	7B	B8	52	99	05	}T39Ž‡˜ÜÉ4N{¸R™.
0002C4C0	F9	1C	96	0F	04	DC	5D	00	76	D5	86	D7	81	23	A8	AC	ù.–..Ü].vÕ†×.#¨¬
0002C4D0	2A	46	F6	C4	37	52	71	18	54	79	40	D3	7F	5C	8E	56	*FöÄ7Rq.Ty@Ó.\ŽV

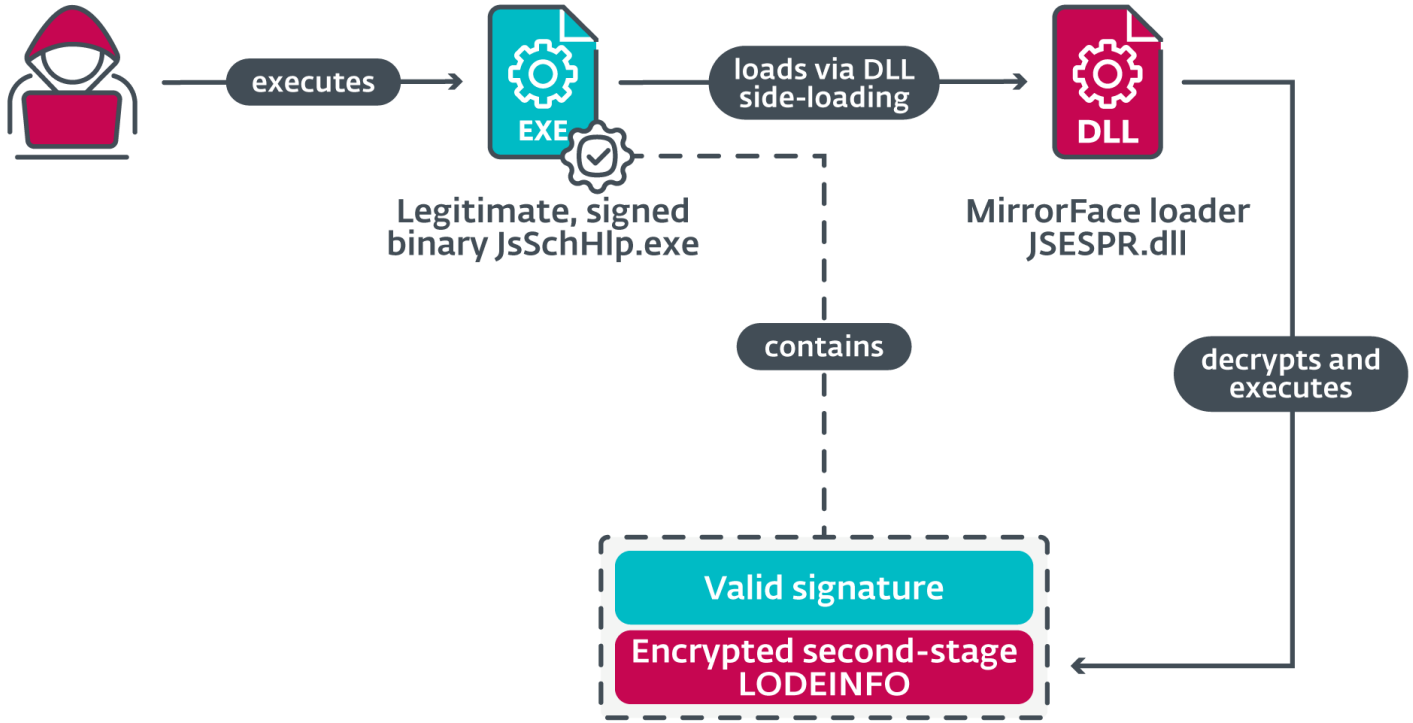
Legitimate signature

RC4 key (0x40)

Encrypted payload

LODEINFOLDR Type 2

- Looks like LODEINFOLDR Type 2 is same as the loader of LODEINFO used in LiberalFace Campaign



https://www.eset.com/jp/blog/welivesecurity/unmasking-mirrorface/

LODEINFO

- We have observed **v0.6.9, v0.7.1, v0.7.2 and v0.7.3** in Campaign #2

v0.6.9	v0.7.1	v0.7.2	v0.7.3
command	command	command	command
ls	ls	ls	ls
rm	rm	rm	rm
mv	mv	mv	mv
cp	cp	cp	cp
cat	cat	cat	cat
mkdir	mkdir	mkdir	mkdir
send	send	send	send
recv	recv	recv	recv
memory	memory	memory	memory
kill	kill	kill	kill
cd	cd	cd	cd
ver	ver	ver	ver
print	print	print	print
ransom (not implemented)	ransom (not implemented)	ransom (not implemented)	ransom (not implemented)
comc	comc	comc	comc
config	config	config	config
pkill	pkill	pkill	pkill
ps	ps	ps	ps
keylog	keylog	keylog	keylog
autorun	autorun	autorun	autorun
	runas	runas	runas

Variant of LODEINFO

- LODEINFO observed in Campaign #2 can be distinguished from LODEINFO that was used in Initial Access in Campaign #1
  - support to run DLL in memory without backdoor command
  - support to run shellcode in memory without backdoor command
- ESET distinguishes this type of LODEINFO as "**The 2nd stage LODEINFO**"

1st stage LODEINFO

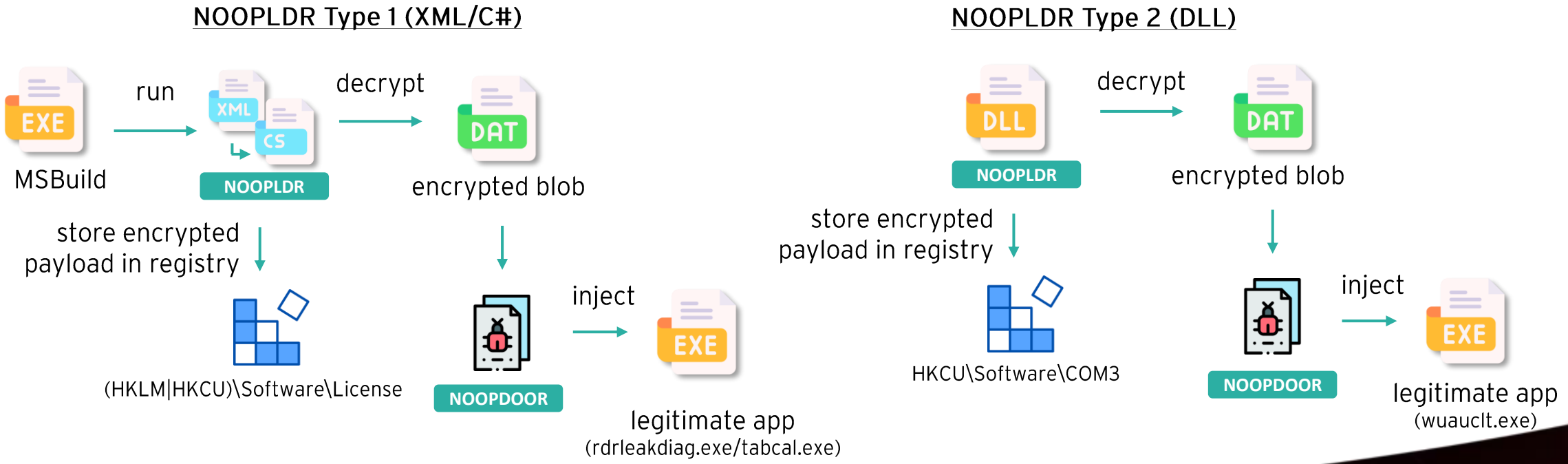
```
if ( a1 )
{
  v1 = (_DWORD *)a1[1];
  v2 = *a1;
  v3 = (_DWORD *)v1[2];
  v5 = v3;
  if ( *v3 )
  {
    init_encstr_table(v6, v2);
    if ( process_backdoor_command(v6, (int)v1) )
      *(_DWORD *)(*v1 + 40) = (*(int (**)(void))(v2 + 160))();
    sub_1DB25(v6);
    v3 = v5;
  }
  (*(void (__cdecl **)(_DWORD *))(v2 + 104))(v3);
  (*(void (__cdecl **)(_DWORD *))(v2 + 104))(v1);
  (*(void (__cdecl **)(int *))(v2 + 104))(a1);
}
```

2nd stage LODEINFO's backdoor command process routine

```
if ( v4 == 'M' && *(v15 + 5) == 'Z' )      // run DLL
{
  v13 = v1;
  v6 = sub_6615(v5, v3, v15);
  v7 = v6;
  if ( v6 )
  {
    if ( *(v6 + 20) )
    {
      strcpy(v14, "MyExpFunc");
      v8 = (sub_2485)(v6, v14);
      if ( v8 || (v8 = (sub_2485)(v7, 0)) != 0 )
        v8(0);
    }
    else
    {
      v11 = *(v6 + 36);
      if ( v11 && *(v7 + 24) )
        v11();
    }
  }
}
else
{
  if ( v4 != 0xE9 )                        // process backdoor commands
  {
    init_encstr_table(v16, v1);
    v9 = v12;
    if ( (process_backdoor_command)(v12) )
      *(*v12 + 40) = (*(v1 + 160))();
    sub_6245(v16);
    goto LABEL_11;
  }
  v13 = 0;
  if ( (*(v1 + 52))(v5, v3, 64, &v13) )
  {
    *v5 = 0xE9;
    v5(0);                                 // run shellcode
  }
}
```

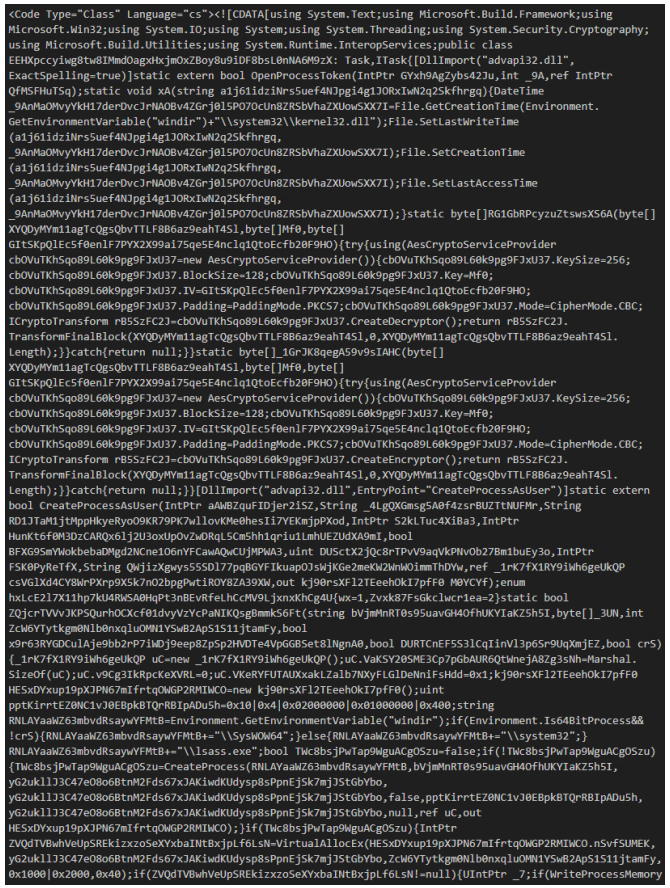

NOOPLDR (aka FakeXInjector)

- Custom loader of NOOPDOOR
- We have observed 2 types of NOOPLDR, which is totally different (XML vs DLL), but both types implement similar strategy to decrypt/store the encrypted payload with using the Device ID of the machine



NOOPLDR (XML/C#)

MSBuild task containing
obfuscated C# to load NOOPDOOR



1. Get MachineId value (=machine Device ID)
   - HKLM\Software\Microsoft\SQMClient\MachineId
2. Read encrypted payload from registry
   - HKCU\Software\License\{HEX}
3. Decrypt by AES128-CBC with key generated by SHA384 of MachineId + computer name
4. Inject the decrypted shellcode into legitimate app (rdrleakdiag.exe/tabcal.exe)

NOOPLDR (DLL)

- Designed to be registered as a Windows Service
- Implements stealthy techniques
  - Control Flow obfuscation
  - Junk codes
  - String encoding with XOR
  - Syscall on process injection

```
00007FFC790C9D10   48:894C24 08      mov qword ptr ss:[rsp+8],rcx
00007FFC790C9D15   48:895424 10      mov qword ptr ss:[rsp+10],rdx
00007FFC790C9D1A   4C:894424 18      mov qword ptr ss:[rsp+18],r8
00007FFC790C9D1F   4C:894C24 20      mov qword ptr ss:[rsp+20],r9
00007FFC790C9D24   48:83EC 28        sub rsp,28
00007FFC790C9D28   B9 832B5041       mov ecx,41502B83
00007FFC790C9D2D   E8 1C120200       call symsrv.7FFC790EAF4E
00007FFC790C9D32   48:83C4 28        add rsp,28
00007FFC790C9D36   48:8B4C24 08      mov rcx,qword ptr ss:[rsp+8]
00007FFC790C9D3B   48:8B5424 10      mov rdx,qword ptr ss:[rsp+10]
00007FFC790C9D40   4C:8B4424 18      mov r8,qword ptr ss:[rsp+18]
00007FFC790C9D45   4C:8B4C24 20      mov r9,qword ptr ss:[rsp+20]
00007FFC790C9D4A   4C:8BD1           mov r10,rcx
00007FFC790C9D4D   0F05              syscall                          NtWriteVirtualMemory
00007FFC790C9D4F   C3                ret
```

Control Flow obfuscation and junk codes

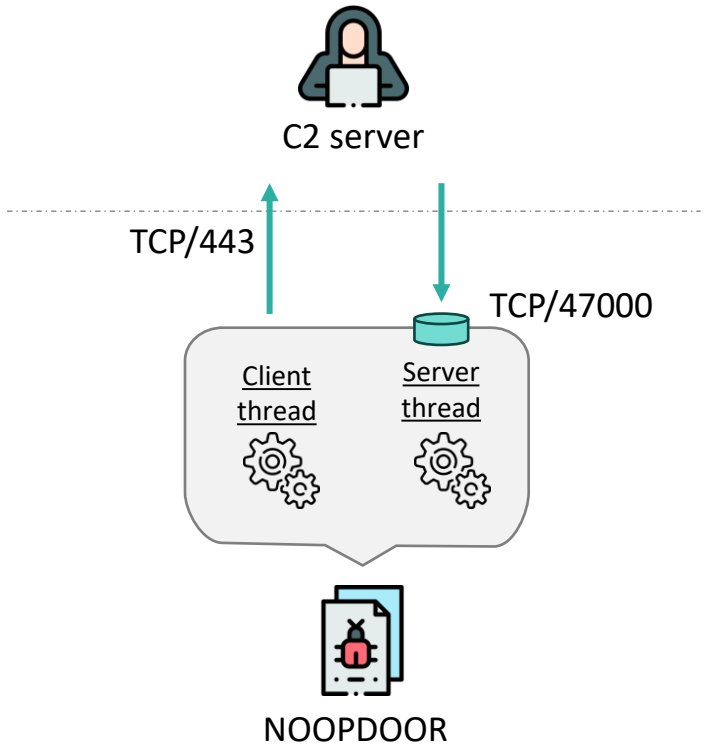
```
__int64 __fastcall sub_7FFAF2FC0AE0(LPCSTR lpSrc, __int64 a2, size_t a3, const CHAR *a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v58 = -618811393;
  v7 = 127828533;
  while ( 1 )
  {
    while ( 1 )
    {
      while ( 1 )
      {
        while ( 1 )
        {
          while ( 1 )
          {
            while ( v7 <= -30399256 )
            {
              if ( v7 > -1095341677 )
              {
                if ( v7 <= -550330022 )
                {
                  if ( v7 <= -802998124 )
                  {
                    if ( v7 <= -947375653 )
                    {
                      if ( v7 > -1031782135 )
                      {
                        if ( v7 <= -982615776 )
                        {
                          if ( v7 <= -1004208705 )
                          {
                            if ( v7 <= -1015941117 )
                            {
                              if ( v7 == -1031782134 )
                              {
                                v7 = -330262518;
                              }
                              else
                              {
                                v223 = PtInRect(0i64, 0i64);
                                v7 = -390545418;
                              }
                            }
                            else if ( v7 == -1015941116 )
                            {
                              v7 = -1198009038;
                            }
                            else if ( v7 == -1009911998 )
                            {
                              v7 = 363348955;
                            }
                            else
                            {
                              v7 = 608672795;
                            }
                          }
```

```
while ( 1 )
{
  IsIntlEqualA = StrIsIntlEqualA(0, 0i64, 0i64, 0);
  v2 = 0i64;
  switch ( IsIntlEqualA )
  {
    case 0i64:
      break;
    case 1i64:
      LoadCursorW(0i64, 0i64);
      v3 = !PtInRect(0i64, 0i64);
      v2 = 0i64;
      switch ( v3 )
      {
        case 0i64:
          BroadcastSystemMessageExA(0, 0i64, 0, 0i64, 0i64, 0i64);
          PrettyW = PathMakePrettyW(0i64);
          v2 = 0i64;
          switch ( PrettyW )
          {
            case 0i64:
              goto LABEL_6;
            case 1i64:
              v2 = 1i64;
              break;
          }
          break;
        case 1i64:
          goto LABEL_6;
      }
      break;
  }
LABEL_6:
  switch ( v2 )
  {
    case 0i64:
      while ( 1 )
      {
        v5 = 0i64;
        switch ( StrIsIntlEqualA(0, 0i64, 0i64, 0) )
        {
          case 0:
            break;
```


NOOPDOOR (aka HiddenFace)

- 64-bit modulable backdoor over TCP with proxy support
- NOOPDOOR is probably designed as the further payload of LODEINFO
- Capabilities
  - Client thread: polling C&C server (main thread)
    - Run program
    - Upload/Download file
    - Change timestamp of specific file
    - Read specific file (%SystemRoot%\System32\msra.tlb)
  - Server thread: listening on TCP/47000
    - Run program
    - Upload/Download file
    - Change working directory
    - Execute shellcode

NOOPDOOR is basically active backdoor but also supports passive backdoor



DGA

- NOOPDOOR's C&C server will be generated based on the current time, therefore it can be changed everyday (by default, but domain's lifespan can be changed based on the seed URL)

e.g. http://$j[].srmbr.com:443/#180

action	example
1. Calculate the today's FILETIME and get the lower 4 bytes	2023/12/01 00:00:00 -> 0x365692200 -> 0x65692200
2. If the seed URL contains "#<NUM>", calculate NUM * the value in Step #1 // NUM	0x65692200 -> 0x650a3600
3. If the seed URL contains "[]", replace it with the current hostname	hxxp://$j[TEST].srmbr[.]com:443/#180
4. Calculate SHA256 of the value in Step #3 in little endian	217826c36e994d097eadcf856fdcadb21372e5a0845e496dbb6015e1a8d42867
5. Calculate SHA512 of the hash in Step #4 + seed URL	dd5b50e5e0405a221fc3c45f8d40ae37733f190b62e2b64e5de10cd190224453d90eaf…
6. Base64 encode the value in Step #5 and get the first 17 bytes	3VtQ5eBAWiIfw8RfjUCuN3M/GQti4rZOXeEM0ZAiRFPZDq8hG4tUb0ce7kFR4zdmLx+xTZe2Z0P…
7. Remove digits and symbols, and make it lower letter	3VtQ5eBAWiIfw8Rfj -> VtQeBAWiIfwRfj -> vtqebawiifwrfj
8. Remove adjacent duplicated letters	vtqebawiifwrfj -> vtqebawifwrfj
9. Replace "$<KEY>" with the generated string	hxxp://vtqebawifwrfj[.]srmbr[.]com:443/

"#<NUM>" and *"[]"* is optional, and the letter in *"$<KEY>"* can be changed

NOOPDOOR vs. ANEL Loader

- ANEL is a 32bit backdoor used by APT10 in 2017~2019
- We confirmed that there're several code similarities between NOOPDOOR and ANEL Loader
- We guess the author(s) of NOOPDOOR and ANEL Loader might be same individual or share TTPs/tools

	NOOPDOOR	ANEL
timeline	2021~	2017~2019
arch	64bit	32bit
form	shellcode	shellcode (Loader) -> DLL
protocol	TCP	HTTP

NOOPDOOR vs. ANEL Loader (#1)

- The way of managing global variables in shellcode is same
  - It calls interesting function to get the address of global variable storage
  - In this function, it calls another function which does just “POP" the return address of this function
  - The return address should be the start address of global variables

NOOPDOOR

```
0000000001FA60                  ; struct_a1_1 *__fastcall get_config_addr()
0000000001FA60                  get_config_addr proc near               ; CODE XREF: sub_10+ABfp
0000000001FA60                                                          ; sub_100+1Afp ...
0000000001FA60 E8 40 00 00 00                   call    sub_1FAA5
0000000001FA65 90                               nop
0000000001FA66 90                               nop
0000000001FA67 90                               nop
0000000001FA68 90                               nop
0000000001FA69 90                               nop
0000000001FA6A 90                               nop
0000000001FA6B 90                               nop
0000000001FA6C 90                               nop
0000000001FA6D 90                               nop
0000000001FA6E 90                               nop
0000000001FA6F 90                               nop
0000000001FA70 90                               nop
0000000001FA71 90                               nop
0000000001FA72 90                               nop
0000000001FA73 90                               nop
0000000001FA74 90                               nop
0000000001FA75 90                               nop
0000000001FA76 90                               nop
0000000001FA77 90                               nop
0000000001FA78 90                               nop
0000000001FA79 90                               nop
0000000001FA7A 90                               nop
0000000001FA7B 90                               nop
0000000001FA7C 90                               nop
0000000001FA7D 90                               nop
0000000001FA7E 90                               nop
0000000001FA7F 90                               nop
0000000001FA80 90                               nop
0000000001FA81 90                               nop
0000000001FA82 90                               nop
0000000001FA83 90                               nop
0000000001FA84 90                               nop
0000000001FA85 90                               nop
0000000001FA86 90                               nop
0000000001FA87 90                               nop
0000000001FA88 90                               nop
0000000001FA89 90                               nop
0000000001FA8A 90                               nop
0000000001FA8B 90                               nop
0000000001FA8C 90                               nop
0000000001FA8D 90                               nop
0000000001FA8E 90                               nop
0000000001FA8F 90                               nop
0000000001FA90 90                               nop
0000000001FA91 90                               nop
0000000001FA92 90                               nop
0000000001FA93 90                               nop
0000000001FA94 90                               nop
0000000001FA95 90                               nop
0000000001FA96 90                               nop
0000000001FA97 90                               nop
0000000001FA98 90                               nop
0000000001FA99 90                               nop
0000000001FA9A 90                               nop
0000000001FA9B 90                               nop
0000000001FA9C 90                               nop
0000000001FA9D 90                               nop
0000000001FA9E 90                               nop
0000000001FA9F 90                               nop
0000000001FAA0 90                               nop
0000000001FAA1 90                               nop
0000000001FAA2 90                               nop
0000000001FAA3 90                               nop
0000000001FAA4 90                               nop
0000000001FAA4                  get_config_addr endp
0000000001FAA4
0000000001FAA5
0000000001FAA5                  ; =============== S U B R O U T I N E =======================
0000000001FAA5
0000000001FAA5
0000000001FAA5                  ; __int64 sub_1FAA5()
0000000001FAA5                  sub_1FAA5       proc near               ; CODE XREF: get_config_
0000000001FAA5 58                               pop     rax
0000000001FAA6 C3                               retn
```

ANEL Loader

```
seg000:00000005                 ; int sub_5()
seg000:00000005                 sub_5           proc near               ; CODE XREF: sub_6C+58↓
seg000:00000005                                                         ; sub_186+6↓p ...
seg000:00000005 E8 26 00 00 00                  call    sub_30
seg000:0000000A 90                              nop
seg000:0000000B 90                              nop
seg000:0000000C 90                              nop
seg000:0000000D 90                              nop
seg000:0000000E 90                              nop
seg000:0000000F 90                              nop
seg000:00000010 90                              nop
seg000:00000011 90                              nop
seg000:00000012 90                              nop
seg000:00000013 90                              nop
seg000:00000014 90                              nop
seg000:00000015 90                              nop
seg000:00000016 90                              nop
seg000:00000017 90                              nop
seg000:00000018
seg000:00000018                 loc_18:                                 ; DATA XREF: sub_186+1A
seg000:00000018                                                         ; sub_71B+2↓r
seg000:00000018 90                              nop
seg000:00000019 90                              nop
seg000:0000001A 90                              nop
seg000:0000001B 90                              nop
seg000:0000001C 90                              nop
seg000:0000001D 90                              nop
seg000:0000001E 90                              nop
seg000:0000001F 90                              nop
seg000:00000020 90                              nop
seg000:00000021 90                              nop
seg000:00000022 90                              nop
seg000:00000023 90                              nop
seg000:00000024 90                              nop
seg000:00000025 90                              nop
seg000:00000026 90                              nop
seg000:00000027 90                              nop
seg000:00000028 90                              nop
seg000:00000029 90                              nop
seg000:0000002A 83 E8 01                        sub     eax, 1
seg000:0000002D 83 C0 01                        add     eax, 1
seg000:0000002D                 sub_5           endp
seg000:0000002D
seg000:00000030
seg000:00000030                 ; =============== S U B R O U T I N E =======================
seg000:00000030
seg000:00000030
seg000:00000030                 ; int sub_30(void)
seg000:00000030                 sub_30          proc near               ; CODE XREF: sub_5↑p
seg000:00000030 58                              pop     eax
seg000:00000031 C3                              retn
```


NOOPDOOR vs. ANEL Loader (#2)

- API hashing algorithm is same
  - both uses ROR7 + 4bytes XOR
  - XOR key is different for each API

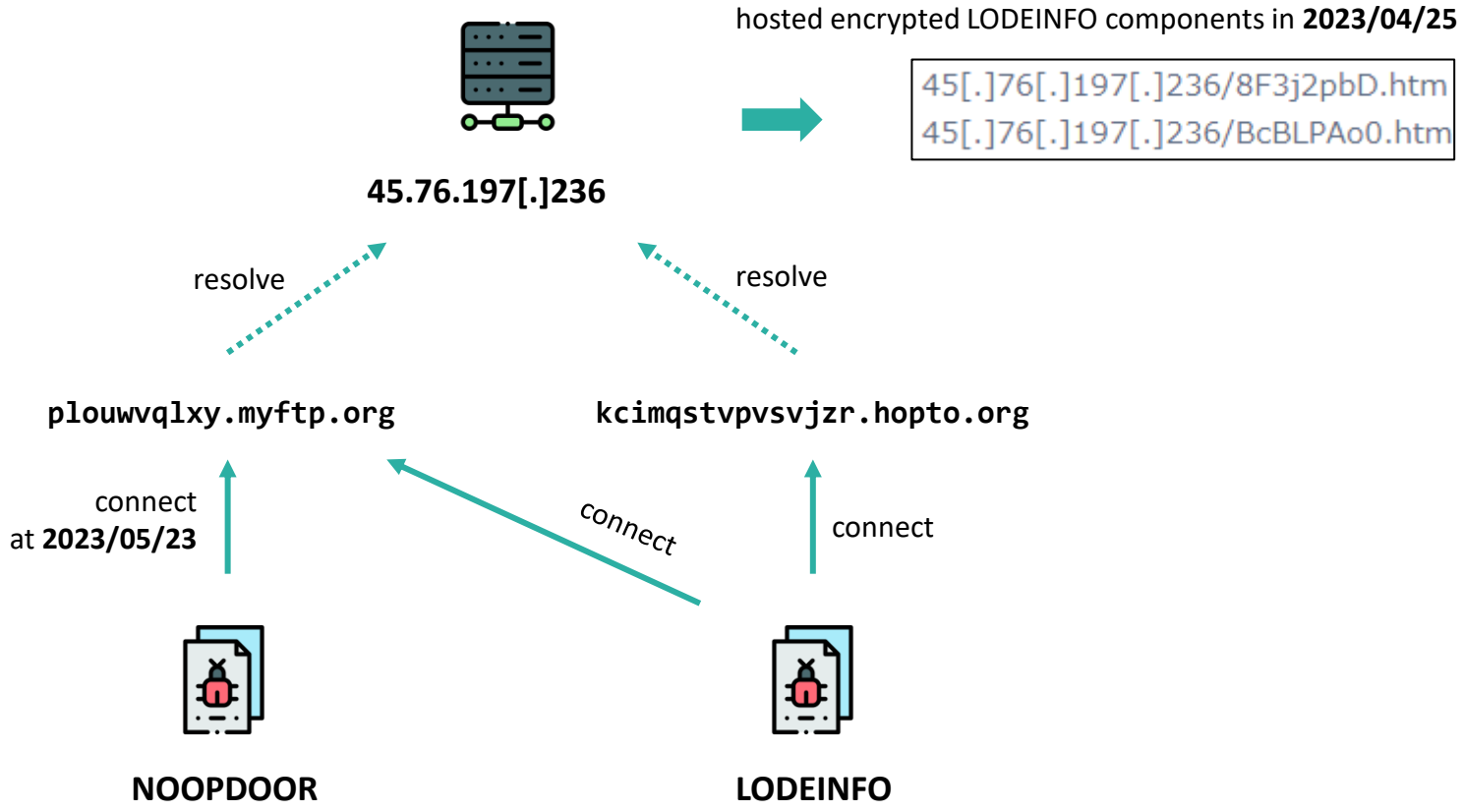
NOOPDOOR

```
BYTE *__fastcall find_api(PBYTE dllbase, int hash)
{
  imports *import_table; // r13
  _IMAGE_NT_HEADERS *v5; // rcx
  __int64 v6; // rdx
  unsigned int v7; // ebp
  __int64 v8; // rdi
  __int64 v9; // r14
  BYTE *i; // rsi
  BYTE *c; // rcx
  int value; // eax
  __int64 v14; // r8

  import_table = get_import_table();
  if ( !dllbase )
    return 0i64;
  if ( *dllbase != 'ZM' )
    return 0i64;
  v5 = *(dllbase + 15);
  if ( !*&dllbase[v5 + 140] )
    return 0i64;
  v6 = *&dllbase[v5 + 136];
  if ( !v6 )
    return 0i64;
  v7 = *&dllbase[v6 + 24];
  v8 = 0i64;
  if ( !v7 )
    return 0i64;
  v9 = *&dllbase[v6 + 32];
  for ( i = &dllbase[v9]; ; i += 4 )
  {
    c = &dllbase[*i];
    value = *c ? ror4_7bit(c + 1, *c) : 0;
    if ( !(hash ^ value ^ 0x3B4055) )
      break;
    v8 = (v8 + 1);
    if ( v8 >= v7 )
      return 0i64;
  }
  if ( import_table->kernel32_GetProcAddress )
    return (import_table->kernel32_GetProcAddress)(dllbase, &dllbase[*&dllbase[4 * v8 + v9]]);
  v14 = *&dllbase[*(dllbase + 15) + 136];
  return &dllbase[*&dllbase[4 * *&dllbase[2 * v8 + *&dllbase[v14 + 36]] + *&dllbase[v14 + 28]]];
}
```

ANEL Loader

```
char *__cdecl find_api(char *a1, int a2)
{
  int v3; // eax
  int v4; // esi
  char *v5; // esi
  char *j; // ebx
  char *i; // ebx
  int (__stdcall *v10)(char *, int); // [esp+8h] [ebp-Ch]
  int v11; // [esp+Ch] [ebp-8h]
  char *v12; // [esp+1Ch] [ebp+8h]

  v11 = 0;
  if ( a1 )
  {
    if ( *a1 == 'ZM' )
    {
      v3 = *(a1 + 15);
      if ( *&a1[v3] == 'EP' )
      {
        if ( *&a1[v3 + 124] )
        {
          v4 = *&a1[v3 + 120];
          if ( v4 )
          {
            v5 = &a1[v4];
            if ( *(v5 + 6) )
            {
              v10 = *sub_5();
              v12 = 0;
              if ( v10 == 0x90909090 )
              {
                if ( *(v5 + 6) )
                {
                  for ( i = &a1[*(v5 + 8)]; ror4_7bit(0, &a1[*i]) != (a2 ^ 0x70CC57); i += 4 )
                  {
                    if ( ++v12 >= *(v5 + 6) )
                      return v11;
                  }
                  return &a1[*&a1[4 * *&a1[2 * v12 + *(v5 + 9)] + *(v5 + 7)]];
                }
              }
              else if ( *(v5 + 6) )
              {
                for ( j = &a1[*(v5 + 8)]; ror4_7bit(0, &a1[*j]) != (a2 ^ 0x70CC57); j += 4 )
                {
                  if ( ++v12 >= *(v5 + 6) )
                    return v11;
                }
                return v10(a1, *(v5 + 4) + *&a1[2 * v12 + *(v5 + 9)]);
              }
            }
          }
        }
      }
    }
  }
  return v11;
}
```


Infra Overlap with NOOPDOOR and LODEINFO



Attribution and Insights

Diamond Model Comparison (Two LODEINFO Campaigns)

- Since no strong inconsistency, we believe the same group has updated TTPs in 2023

LODEINFO Campaign #1

Technical Axis

- ✓ Spear-phishing
- ✓ DLL Side-Loading
- ✓ MS13-098/CVE-2013-3900

Social-Political Axis

- ✓ Rise of tension in East Asia

ADVERSARY

Group:
- ✓ related-to APT10
- ✓ Believed to be China-nexus

CAPABILITY

Tools:
- ✓ LODEINFO
- ✓ NOOPDOOR
- ✓ DOWNIISSA
- ✓ Lilim RAT
- ✓ MirrorStealer

INFRASTRUCTURE

- ✓ VPS/Hosting Service (CHOOPA, Vultr, Linode)
- ✓ IP located in Japan as C2

VICTIM

Geography:
- ✓ Japan

Industry:
- ✓ Targeting public sector, individuals associated international affairs, politicians and researchers in academic sector

LODEINFO Campaign #2

Technical Axis

- ✓ Exploit public-facing application
- ✓ DLL Side-Loading
- ✓ MS13-098/CVE-2013-3900

Social-Political Axis

- ✓ Rise of tension in East Asia

ADVERSARY

Group:
- ✓ related-to APT10
- ✓ Believed to be China-nexus

CAPABILITY

Tools:
- ✓ LODEINFO
- ✓ NOOPDOOR
- ✓ Cobalt Strike
- ✓ MirrorStealer

INFRASTRUCTURE

- ✓ Compromised website as C2
- ✓ Abuse of DDNS

VICTIM

Geography:
- ✓ Japan, Taiwan and India

Industry:
- ✓ private sector including manufacturing and aviation
- ✓ Hi-tech related organization
- ✓ government agency

Diamond Model Comparison (Earth Tengshe vs. Earth Kasha)

- No overlaps in Capability, but many overlaps in TTPs of Pre/Post-Exploitation

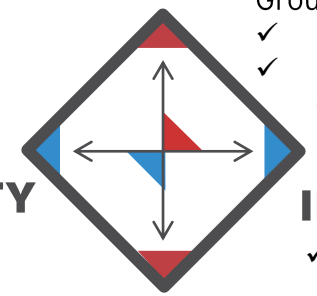
A41APT Campaign

Technical Axis

- ✓ Exploit public-facing application
- ✓ DLL Side-Loading
- ✓ MS13-098/CVE-2013-3900
- ✓ Lateral Movement via RDP
- ✓ Defensive Evasion by wevutil
- ✓ Collection by csvde
- ✓ Credential Access by registry hive

Social-Political Axis

- ✓ Rise of tension in East Asia



ADVERSARY

Group:
- ✓ related-to APT10
- ✓ Believed to be China-nexus

CAPABILITY

Tools:
- ✓ SigLoader
- ✓ HUI Loader
- ✓ SodaMaster
- ✓ P8Rat
- ✓ FYAnti
- ✓ Cobalt Strike
- ✓ Jackpot

INFRASTRUCTURE

- ✓ Heavy use of IP address as C2

VICTIM

Geography:
- ✓ Japan, Taiwan, Thailand and United States (but main target is entity in Japan)

Industry:
- ✓ Targeting private sector, including electronics, energy, automotive, defense industries

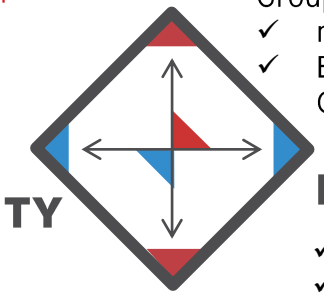
LODEINFO Campaign #2

Technical Axis

- ✓ Exploit public-facing application
- ✓ DLL Side-Loading
- ✓ MS13-098/CVE-2013-3900
- ✓ Lateral Movement via RDP
- ✓ Defensive Evasion by wevutil
- ✓ Collection by csvde
- ✓ Credential Access by registry hive

Social-Political Axis

- ✓ Rise of tension in East Asia



ADVERSARY

Group:
- ✓ related-to APT10
- ✓ Believed to be China-nexus

CAPABILITY

Tools:
- ✓ LODEINFO
- ✓ NOOPDOOR
- ✓ Cobalt Strike
- ✓ MirrorStealer

INFRASTRUCTURE

- ✓ Compromised website as C2
- ✓ Abuse of DDNS

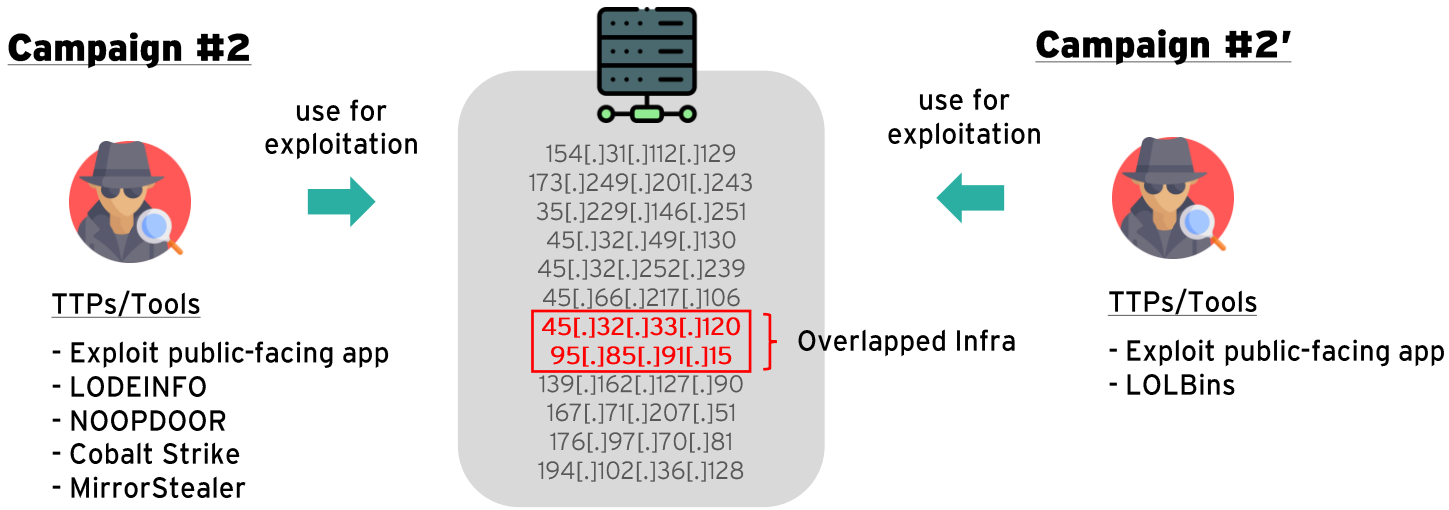
VICTIM

Geography:
- ✓ Japan, Taiwan and India

Industry:
- ✓ private sector including manufacturing and aviation
- ✓ Hi-tech related organization
- ✓ government agency

Yet another campaign by Earth Kasha? Or another attacker?

- We observed several incidents that use infrastructure common to "LODEINFO Campaign #2", but NO malware was used



Assumption

1. Malware such as LODEINFO and NOOPDOOR might possibly be shared with other groups?
2. There might possibly be a team focusing on initial compromise (like access broker)?

Other Related Groups and Campaigns?

- Several vendors reported the same exploits were abused in near timeline
  - Microsoft reported Volt Typhoon also employs FortiOS/FortiProxy (CVE-2023-27997)
    - https://www.microsoft.com/en-us/security/blog/2023/05/24/volt-typhoon-targets-us-critical-infrastructure-with-living-off-the-land-techniques/
  - LAC reported multiple campaigns abusing Array AG (CVE-2023-28461) and Citrix (CVE-2023-3466, CVE-2023-3467, CVE-2023-3519)
    - https://www.lac.co.jp/lacwatch/pdf/20231108_lsi_vol6.pdf

No overlaps in toolsets and TTPs in Post-Exploitation
thus, these exploits were possibly shared among several groups

Conclusion

Summary

- We believe Earth Kasha has been deploying the new campaign with updated malware and different TTPs
  - Observed overlaps in terms of TTPs with Earth Tengshe's campaign and overlaps in terms of malwares with the previous Earth Kasha's campaign
  - This could indicate that two groups, who both are believed to be related to APT10 Umbrella, might possibly be accorporating or sharing TTPs and tools
  - Furthermore, exploits could possibly be shared among Chinese actors not only APT10 Umbrella

Recommendation

- Review possible Attack Surface of your network
- Use IoC and Yara in Appendix for threat hunting

Thank you

Appendix

IoC #1

SHA256	malware
9c681493c81581995e6a48b96411a7004fe77558d7ca863e26398538ad78f385	NOOPLDR (DLL)
8574a494425825958c1e978ca7f66a467954fa90c7c898eebac49928519f0eae	LODEINFOLDR (Type 2)

domain/IP	malware	comment
ns1[.]tlsart[.]com	Cobalt Strike	
{DGA}[.]hopto[.]org	NOOPDOOR	DDNS, blocking all sub-domains is not recommended
{DGA}[.]gotdns[.]ch	NOOPDOOR	DDNS, blocking all sub-domains is not recommended
{DGA}[.]myftp[.]org	NOOPDOOR / LODEINFO	DDNS, blocking all sub-domains is not recommended
{DGA}[.]tw8sl[.]com	NOOPDOOR	
{DGA}[.]srmbr[.]com	NOOPDOOR	
45[.]76[.]197[.]236	NOOPDOOR / LODEINFO	IP related to domain used by NOOPDOOR and LODEINFO

IoC #2

observed filepath	malware
C:\Windows\System32\hello.xml	NOOPLDR
C:\Windows\System32\hello.bin	encrypted NOOPDOOR
C:\Windows\System32\stdole2.xml	NOOPLDR
C:\Windows\system32\certmgr.dat	encrypted NOOPDOOR
C:\Windows\System32\C_G18030.xml	NOOPLDR
C:\Windows\system32\C_950.dat	encrypted NOOPDOOR
C:\Windows\System32\UIAnimation.xml	NOOPLDR
C:\Windows\System32\ComputerToastIcon.contrast-white.dat	encrypted NOOPDOOR
C:\Windows\System32\mssitlb.xml	NOOPLDR
C:\Windows\System32\ShiftJIS.dat	encrypted NOOPDOOR
C:\Windows\System32\HashtagDS.xml	NOOPLDR
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\WsatConfig.dat	encrypted NOOPDOOR
C:\Windows\System32\capisp.xml	NOOPLDR
C:\Windows\System32\ddrawex.dat	encrypted NOOPDOOR
C:\Windows\System32\symsrv.dll	NOOPLDR
C:\Windows\System32\symstore.exe_config	encrypted NOOPDOOR

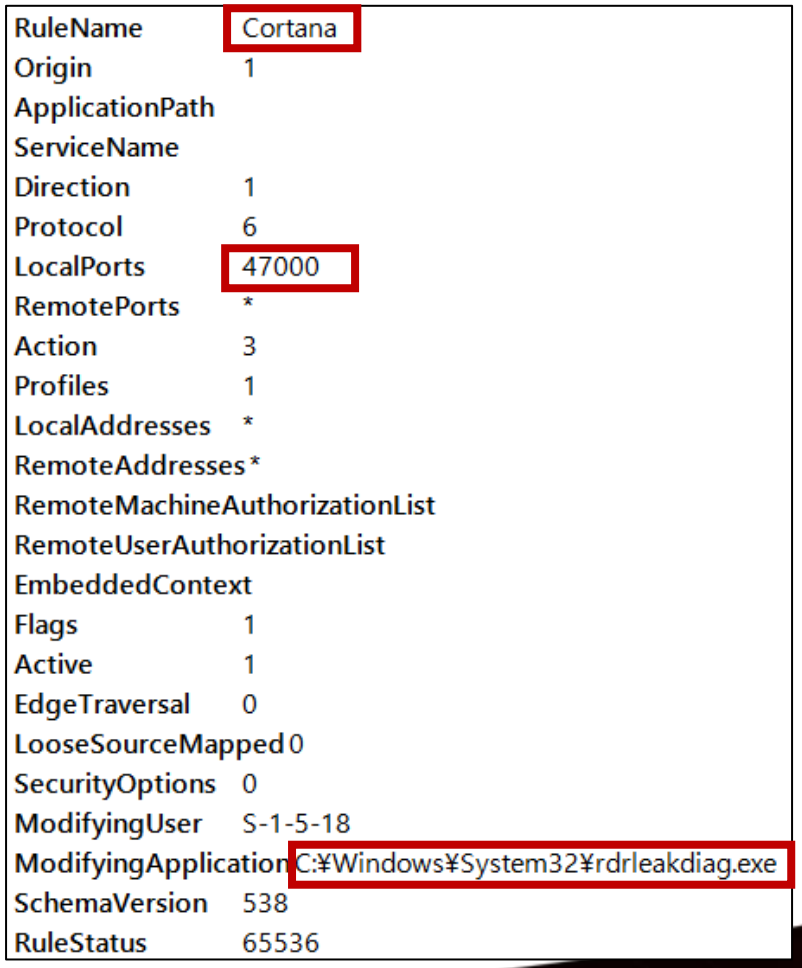
observed filepath	malware
C:\Windows\Branding\shellbrd\tedutil.dll	LODEINFOLDR
C:\Windows\Temp\SfsDll32.dll	LODEINFOLDR
C:\Windows\Temp\JSESPR.dll	LODEINFOLDR
C:\Windows\Temp\K7AVWScn.dll	LODEINFOLDR
C:\Windows\Temp\libcef.dll	GOSICLOADER
C:\Windows\Temp\Ushata.dll	BINSCOUT
C:\Windows\DiagTrack\provider.vbs	POWERPROVIDER
%Temp%\31558.txt	output of MirrorStealer

IoC #3

value	type
\Microsoft\Windows\Media Center\PBDADiscoveryW3	Scheduled Task name related to NOOPDOOR
\Microsoft\Windows\Media Center\PBDADiscoveryW4	Scheduled Task name related to NOOPDOOR
\Microsoft\Windows\Media Center\Presentation	Scheduled Task name related to NOOPDOOR
msupdate	Service name related to NOOPDOOR
(HKLM|HKCU)\Software\License	Registry key that contains encrypted NOOPDOOR
HKCU\Software\COM3	Registry key that contains encrypted NOOPDOOR

Notable Event Log

- Event Id: 2004
- Source: Microsoft-Windows-Windows Firewall with Advanced Security
- Description: A rule has been added to the Windows Firewall exception list.
- RuleName = 'Cortana' AND LocalPorts = 47000
  - NOOPDOOR adds a new rule "Cortana" in local Firewall setting to allow incoming connection to port 47000 for passive mode



RuleName	Cortana
Origin	1
ApplicationPath	
ServiceName	
Direction	1
Protocol	6
LocalPorts	47000
RemotePorts	*
Action	3
Profiles	1
LocalAddresses	*
RemoteAddresses	*
RemoteMachineAuthorizationList	
RemoteUserAuthorizationList	
EmbeddedContext	
Flags	1
Active	1
EdgeTraversal	0
LooseSourceMapped	0
SecurityOptions	0
ModifyingUser	S-1-5-18
ModifyingApplication	C:¥Windows¥System32¥rdrleakdiag.exe
SchemaVersion	538
RuleStatus	65536

Yara Rules

We will release a blog with Yara rule soon

```
rule Trojan_LODEINFOLDR_generic
{
    meta:
        Author = "Trend Micro"
        Created_Time = "2024-01-26"
    strings:
        $chunk_1 = {
            8A 02
            34 ??
            88 01
            8A 42 01
            34 ??
            88 41 01
            8A 42 02
            34 ??
            88 41 02
            8A 42 03
            34 ??
            88 41 03
            8A 42 04
            34 ??
            88 41 04
            8A 42 05
            34 ??
            88 41 05
            8A 42 06
            34 ??
            88 41 06
            8A 42 07
            34 ??
            88 41 07
            8B C1
            C6 41 08 00
        }
    condition:
        uint16(0) == 0x5A4D and
        all of them
}
```

```
rule Trojan_NOOPLDR_xml
{
    meta:
        Author = "Trend Micro"
        Created_Time = "2024-01-26"
    strings:
        $s1 = "<Code Type=\"Class\" Language=\"cs\"><![CDATA[using "
        $s2 = "Software\\\\Microsoft\\\\SQMClient"
        $s3 = ".GetValue(\"MachineId\").ToString()"
        $s4 = "SHA384.Create();"
        $s5 = "new byte[32];Array.Copy("
        $s6 = "new byte[16];Array.Copy("
    condition:
        all of them
}
```

```
rule Trojan_NOOPLDR_dll
{
    meta:
        Author = "Trend Micro"
        Created_Time = "2024-01-26"
    strings:
        $chunk_1 = {
            48 8D 05 ?? ?? ?? ??
            48 8D 0D ?? ?? ?? ??
            0F B6 0C 0E
            32 0C 06
            0F B6 D1
            48 8D 8D ?? ?? ?? ??
            E8 ?? ?? ?? ??
        }

        $chunk_2 = {
            3D ?? ?? ?? ??
            7E ??
            3D ?? ?? ?? ??
            7F ??
            3D ?? ?? ?? ??
            0F 84 ?? ?? ?? ??
            3D ?? ?? ?? ??
            75 ??
            B8 ?? ?? ?? ??
            EB ??
        }

    condition:
        uint16(0) == 0x5A4D and
        all of them
}
```

