What We Can Do against the Chaotic A41APT Campaign

Hajime Yanagishita, Kiyotaka Tamada, You Nakatsuru, Suguru Ishimaru

2022/01/27

Who We Are

Hajime Yanagishita

Macnica

Kiyotaka Tamada

You Nakatsuru

Secureworks

Suguru Ishimaru

Kaspersky

Agenda

Recent A41APT campaign we've seen

- What A41APT Campaign is
- Continuous A41APT Campaign
  - Continued and Updated TTPs
  - News TTPs Observed in 2021
  - Attribution of The Threat Actor
- What We Can Do against the Chaotic A41APT Campaign

What A41APT Campaign is

A41APT Campaign

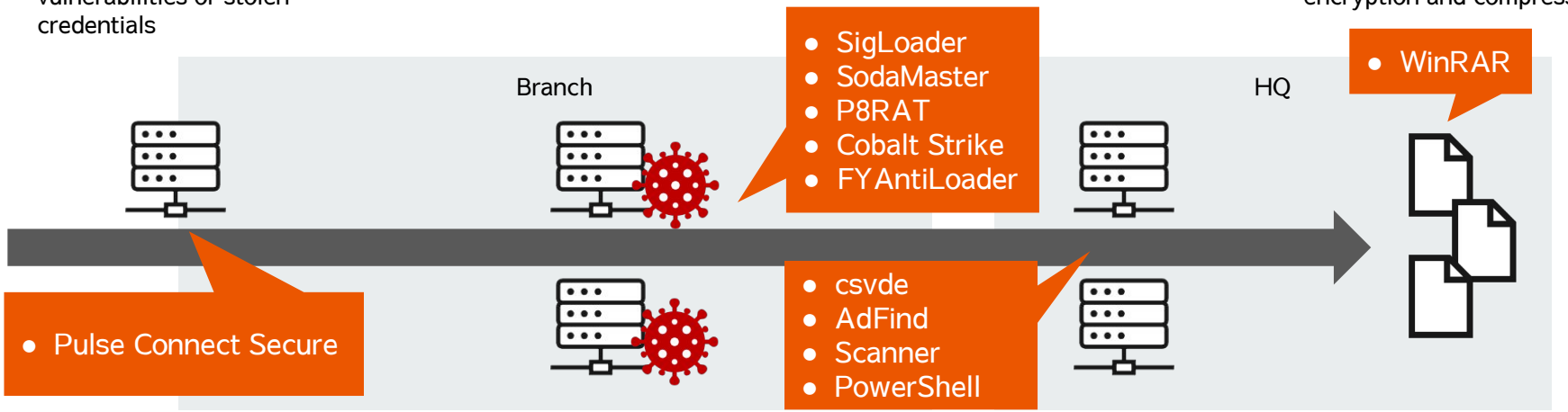
A sophisticated attack campaign revealed at JSAC 2021

- Period of activity
  - March 2019 to January 2021
- Target
  - Japan (Japanese companies and their overseas branches)
- Origin of the campaign name
  - Named after the actor's host name DESKTOP-A41UVJV used for remote connection

Distinguish attack campaign that threat actor intrudes via internet-facing system, deploy malware such as SigLoader/Sodamaster from others

TTPs Reported at JSAC2021

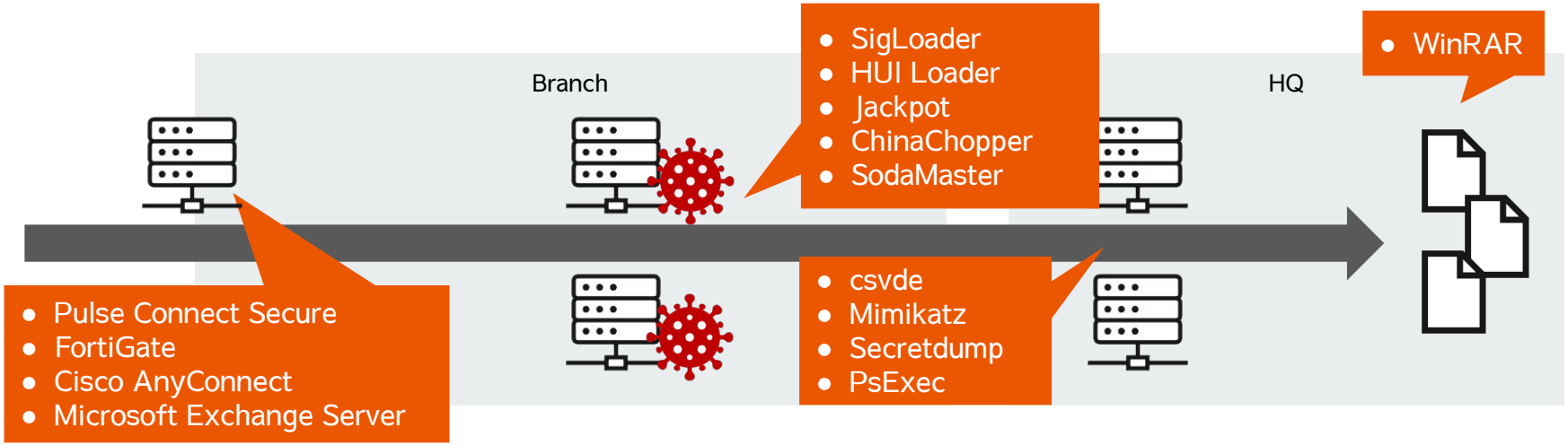
Initial Intrusion	Internal Reconnaissance	Persistence	Lateral Movement	Actions on Objectives
Penetrate via internet -facing system by using vulnerabilities or stolen credentials	Perform network scan, connect to RDP server	Install malware by using Taskscheduler	Connect to RDP server at HQ office	Search for secret information, exfiltrate after encryption and compression



Continuous A41APT Campaign in 2021

Continuous A41APT Campaign in 2021

- Observed attacks against multiple organizations in Japan and branch offices
  - We investigated each different incident to disclose updated TTPs and discovered new TTPs



Public Information by Trend Micro

https://blog.trendmicro.co.jp/archives/29842

しかし最近観測されたSodaMasterのバックドアコマンドは、一部は未実装ですが、アルファベットの「c」から「x」までサポートされていました。またコマンドの分岐処理は、switch文を使わずにループ処理でコマンドハンドラを検索する変則的な手法が用いられていました。

```
if ( g_handlers[0].func )
{
  id = 0i64;
  func = &g_handlers[0].func;
  do
  {
    if ( *&g_handlers[id].cmd_id == a1->cmd_id )
      (*func)(a1->arg, (size - 1));
    id = ++next_id;
    func = &g_handlers[next_id].func;
  }
  while ( *func );
}
```

```
handler         struc ; (sizeof=0x10, mappedto_80)
cmd_id          db ?
aligned         db 7 dup(?)
func            dq ?                    ; offset
handler         ends
```

```
; handler g_handlers[22]
g_handlers dq 'c'
off_180019428 dq offset send_outlook_info
                                        ;
                                        ;
dq 'd'
dq offset run_dll
dq 'e'
dq offset nullsub_2
dq 'f'
dq offset set_flag
dq 'g'
dq offset run_shellcode
dq 'h'
dq offset send_raw_packet
dq 'i'
dq offset send_0xCC_packet
```

「Jackpot」のCommunication Protocol

トレンドマイクロで観測したJackpotは、ハードコードされたURLへのPOSTリクエストのみを処理します。それ以外のメソッドによるリクエストに対しては、正規の応答に見せかけたレスポンスを返します。攻撃者クライアントは、後述する独自のメッセージパケットをカスタムBase64とRC4で暗号化して送信します。この際、次のようなパスワードによる認証プロセスを経ることで、バックドア機能が有効化され、以降のバックドア処理が可能になる仕組みになっていました。これは、不特定多数がアクセス可能な公開システムにJackpotを感染させることを想定し、意図しないリクエストを処理しないための予防策と考えられます。

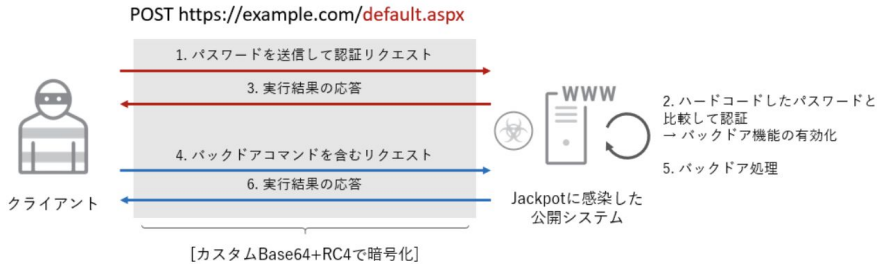


図8：Jackpotがバックドア機能を有効化するためのシーケンス例

Continued and Updated TTPs

Continuous A41APT Campaign in 2021

Intrusion via VPN Devices

Observations in 2021

- Using known vulnerabilities
  - Pulse Connect Secure
  - FortiGate: CVE-2018-13379
  - Cisco AnyConnect: CVE-2020-3125
- Even if it's patched now, the credentials from back then might have been leaked

Note that we've only seen the host name "DESKTOP-O2KM1VL" already reported in 2021

(Never seen "DESKTOP-A41UVJV" - origin of the campaign name)

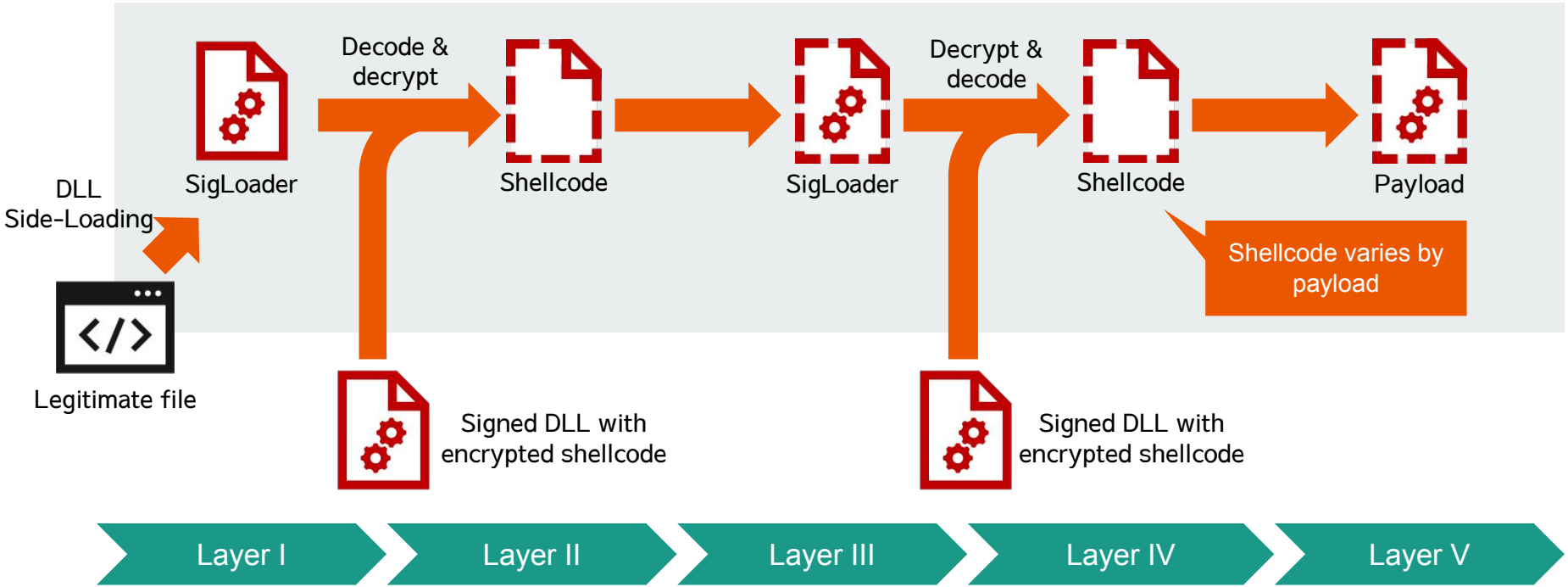
Tool Sets Used After Intrusion

- Following tools were found in the lateral movement stage
  - Mimikatz
  - secretdump.py
  - PsExec
  - csvde
  - WinRAR
- The threat actor seems to use various tools as needed

Malware Updates

- SigLoader and SodaMaster are still used in 2021
  - Cobalt Strike, P8RAT and FYAntiLoader were not observed in 2021
- With some changes:
  - Tampering compile time
    - SigLoader: e.g. 2021/?? -> 2017/05
    - SodaMaster: e.g. 2021/04 -> 2012/10
  - Updates on major functions
    - SigLoader: decryption process
    - SodaMaster: commands and data format

SigLoader Execution Flow



Decryption Process of SigLoader

- Algorithm identifiers were changed from string to number
  - 0: AES
  - 1: DES
  - 3: XOR
- The order of decrytpion algorithm is hardcoded

```
cipher_id_18002DC70 dd 3                ; DATA XREF
                                        ; 3 = XOR
                    dd 0                ; 0 = AES
                    dd 35E1Dh
```

Layer I

```
cipher_id_1CBA0643F30 dd 0              ; DATA XREF:
                                        ; sub_1CBA06
                                        ; 0 = AES
                      dd 3              ; 3 = XOR
```

Layer III

The order is reversed between Layer I and Layer III in all analyzed samples

```
if ( !*cipher_id )                                  // 0 = AES
{
  v190 = 4;
  v191 = 4;
  v192 = 10;
  v193 = 16;
  if ( v30 )
    v47 = v30;
  v121 = operator new(0x35E15ui64);
  *(_QWORD *)MultiByteStr = v121;
  v122 = (char *)operator new(0xB0ui64);
  aes_init__180001EE0(&v190, v122);
  v123 = (__int64)v47;
  v124 = v121 - v47;
  v125 = 13794i64;
  do
  {
    aes_dec__1800019B0(&v190, v123, v124 + v123, v122);
    v123 += 16i64;
    --v125;
  }
  while ( v125 );
  j_j_free(v122);
}
v30 = *(_BYTE **)MultiByteStr;
if ( *cipher_id == 1 )                              // 1 = DES
{
  if ( *(_QWORD *)MultiByteStr )
    v47 = *(_BYTE **)MultiByteStr;
  v30 = (_BYTE *)des_1800034E0(v47);
  *(_QWORD *)MultiByteStr = v30;
}
if ( *cipher_id == 3 )                              // 3 = XOR
```

Decryption Process of SigLoader

- The AES mode was changed from CBC to ECB
- The AES key is the first 16 bytes from the hardcoded 32 bytes string

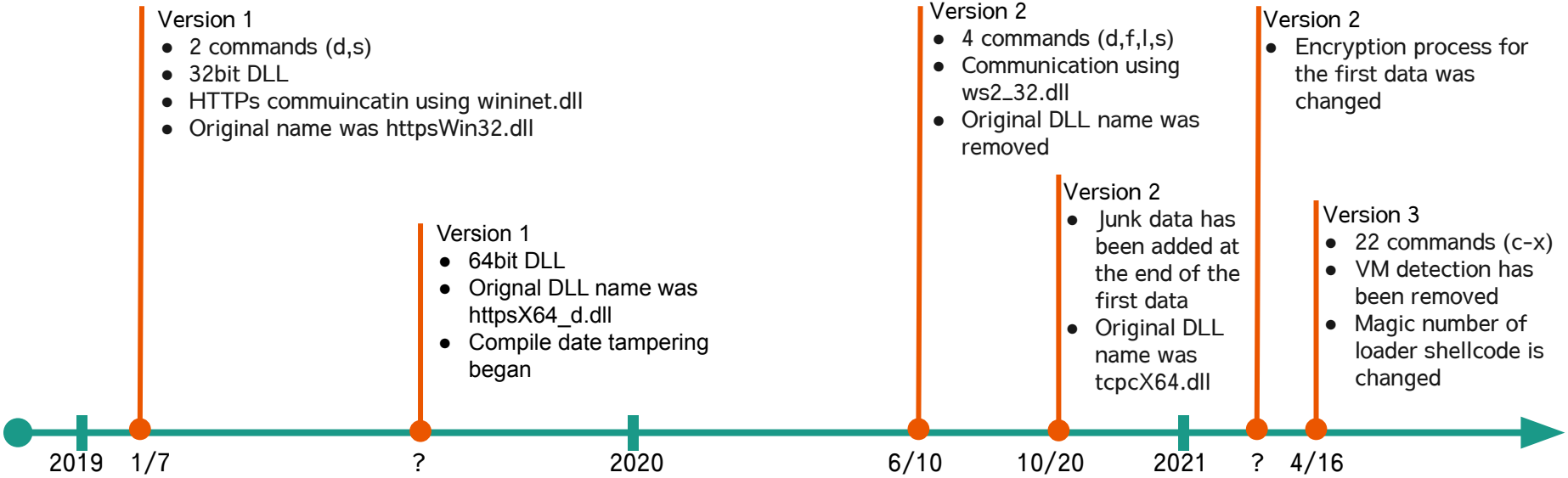
```
FC_Rijndael_sbox db 63h, 7Ch, 77h, 7Bh, 0F2h, 6Bh, 6Fh, 0C5h, 30h, 1, 67h
                                        ; DATA XREF: aes_init_180001EE0+9C↑o
                 db 2Bh, 0FEh, 0D7h, 0ABh, 76h, 0CAh, 82h, 0C9h, 7Dh, 0FAh
                 db 59h, 47h, 0F0h, 0ADh, 0D4h, 0A2h, 0AFh, 9Ch, 0A4h, 72h
                 db 0C0h, 0B7h, 0FDh, 93h, 26h, 36h, 3Fh, 0F7h, 0CCh, 34h
                 db 0A5h, 0E5h, 0F1h, 71h, 0D8h, 31h, 15h, 4, 0C7h, 23h
                 db 0C3h, 18h, 96h, 5, 9Ah, 7, 12h, 80h, 0E2h, 0EBh, 27h
                 db 0B2h, 75h, 9, 83h, 2Ch, 1Ah, 1Bh, 6Eh, 5Ah, 0A0h, 52h
                 db 3Bh, 0D6h, 0B3h, 29h, 0E3h, 2Fh, 84h, 53h, 0D1h, 0
                 db 0EDh, 20h, 0FCh, 0B1h, 5Bh, 6Ah, 0CBh, 0BEh, 39h, 4Ah
                 db 4Ch, 58h, 0CFh, 0D0h, 0EFh, 0AAh, 0FBh, 43h, 4Dh, 33h
                 db 85h, 45h, 0F9h, 2, 7Fh, 50h, 3Ch, 9Fh, 0A8h, 51h, 0A3h
                 db 40h, 8Fh, 92h, 9Dh, 38h, 0F5h, 0BCh, 0B6h, 0DAh, 21h
                 db 10h, 0FFh, 0F3h, 0D2h, 0CDh, 0Ch, 13h, 0ECh, 5Fh, 97h
                 db 44h, 17h, 0C4h, 0A7h, 7Eh, 3Dh, 64h, 5Dh, 19h, 73h
                 db 60h, 81h, 4Fh, 0DCh, 22h, 2Ah, 90h, 88h, 46h, 0EEh
                 db 0B8h, 14h, 0DEh, 5Eh, 0Bh, 0DBh, 0E0h, 32h, 3Ah, 0Ah
                 db 49h, 6, 24h, 5Ch, 0C2h, 0D3h, 0ACh, 62h, 91h, 95h, 0E4h
                 db 79h, 0E7h, 0C8h, 37h, 6Dh, 8Dh, 0D5h, 4Eh, 0A9h, 6Ch
                 db 56h, 0F4h, 0EAh, 65h, 7Ah, 0AEh, 8, 0BAh, 78h, 25h
                 db 2Eh, 1Ch, 0A6h, 0B4h, 0C6h, 0E8h, 0DDh, 74h, 1Fh, 4Bh
                 db 0BDh, 8Bh, 8Ah, 70h, 3Eh, 0B5h, 66h, 48h, 3, 0F6h, 0Eh
                 db 61h, 35h, 57h, 0B9h, 86h, 0C1h, 1Dh, 9Eh, 0E1h, 0F8h
                 db 98h, 11h, 69h, 0D9h, 8Eh, 94h, 9Bh, 1Eh, 87h, 0E9h
                 db 0CEh, 55h, 28h, 0DFh, 8Ch, 0A1h, 89h, 0Dh, 0BFh, 0E6h
                 db 42h, 68h, 41h, 99h, 2Dh, 0Fh, 0B0h, 54h, 0BBh, 16h
```

```
if ( 4 * a1[1] > 0 )
{
  v8 = a2;
  do
  {
    v9 = v8[aeskey - a2];
    ++v6;
    *v8++ = v9;
  }
  while ( v6 < 4 * a1[1] );
}
v10 = 4 * a1[1];
for ( i = v10; v10 < 4 * *a1 * (a1[2] + 1); a2[i - 1] = v4[3] ^ a2[v22 -
{
  v12 = a2[i - 4];
  *v4 = v12;
  v13 = a2[i - 3];
  v4[1] = v13;
  v14 = a2[i - 2];
  v4[2] = v14;
  v15 = a2[i - 1];
  v4[3] = v15;
  v16 = a1[1];
  v17 = v10 / 4 % v16;
  if ( v17 )
    v16 > 6 && v17 == 4 )
    *v4 = FC_Rijndael_sbox[*v4 % 16 + (unsigned __int64)(*v4 & 0xF0)];
    v4[1] = FC_Rijndael_sbox[v4[1] % 16 + (unsigned __int64)(v4[1] & 0
```

nuWU1hZsNRptORhw8iY4sYm0WQjbfjB

SodaMaster Evolution

We classified 20+ samples into 3 versions, and confirmed 6 activities from compile time and common features as follows



Comparison for Each Version of SodaMaster

※ Light gray: Compile date might be tampered

	Version 1		Version 2			Version 3
Compile Date (Export Date)	2019/01/07 10:33:18	2016/07/28 23:34:54	2019/06/10 16:58:10	2020/10/20 10:46:49	2016/07/28 19:07:26	2012/10/13 12:00:07
File Type	x86 DLL	x64 DLL	x64 DLL			x64 DLL
Original DLL Name	httpsWin32.dll	httpsX64_d.dll	-	tcpcX64.dll		tcpcX64.dll
Export Function	DLLEntry		-	DLLEntry		DLLEntry
Network API	**wininet**		**ws2_32**			**ws2_32**
Command	**d, s**		**d, f, l, s**			**c - x**
Anti-VM	✓		✓			-
Addition of junk data	-		-	Using string length of collected PC info	Using GetTickCount	Using GetTickCount

Changes of Loader Shellcode for SodaMaster

The basic implementation was not changed

```
$MagicNumber    dq 0B14195F0B14195Fh
$EncDataSize    dd 19A00h
$RC4_Key        dq 26DBC48A6FB401ECh
                dq 875D06BEF01B754h
$EncSodaMaster  db 0B7h
                db 0E4h
                db  78h ; x
                db 0A3h
                db  88h
                db 0CDh
                db  8Eh
                db  97h
```

- The magic bytes have been changed from version 3
- The size of data was increased depending the payload which was the updated SodaMaster

offset	data	description
0x000	90 90 90 90 90 90 90 90	magic bytes for Identification, this is used for comparision before data processing
0x008	0x11600	Size of encrypted data, only this value (size) is observed
0x00C	A9 5B 7B 84 9C CB CF E8 B6 79 F1 9F 05 B6 2B FE	16 bytes RC4 key (each sample has different key)
0x01C	C7 36 7E 93 D3 07 1E 86 23 75 10 49 C8 AD 01 9F [skipped]	Encrypted SodaMaster payload with RC4

Command List of SodaMaster Version 3

※ e, j, k, n-p, t-v are not Implemented
※ Red: Different analysis results from Trend Micro

Command	Description
c	Steals credentials of Outlook
d	Executes DLL（Specified export function）
f	Enables/Disables adding size info into sending data
g	Executes shellcode（No function table）
h	Repeats sending source spoofed packet to specified destination (DoS?)
i	Repeats sending 0x20000 bytes data padded with 0xCC (DoS?)

Command	Description
l	Configures interval of C2 communications
m	Collects screenshot
q	Enables key logging
r	Disables key logging
s	Executes shellcode (With function table)
w	Shows string with MessageBox
x	Exits process

Changes on C2 Command Execution of SodaMaster

Loop statement for processing a large number of commands

Version 2

```
command = recv_data[4];
switch ( command )
{
  case 'd':
    exec_dll(recv_data + 5, (size - 5));
    break;
  case 'f':
    rc4_key = *(recv_data + 5);
    break;
  case 'l':
    sleep_time = *(recv_data + 5);
    break;
  case 's':
    exec_shellcode(recv_data + 5);
    break;
}
```

Version 3

```
j = 0i64;
if ( command_list[0].func
{
  k = 0i64;
  command_function = &command_list[0].function;
  do
  {
    if ( command_list[k].id == *(recv_data + 4) )
      (*command_function)(recv_data + 5, (data_size - 1));
    k = ++j;
    command_function = &command_list[j].function;
  }
  while ( *command_function );
}
return 1i64;
```

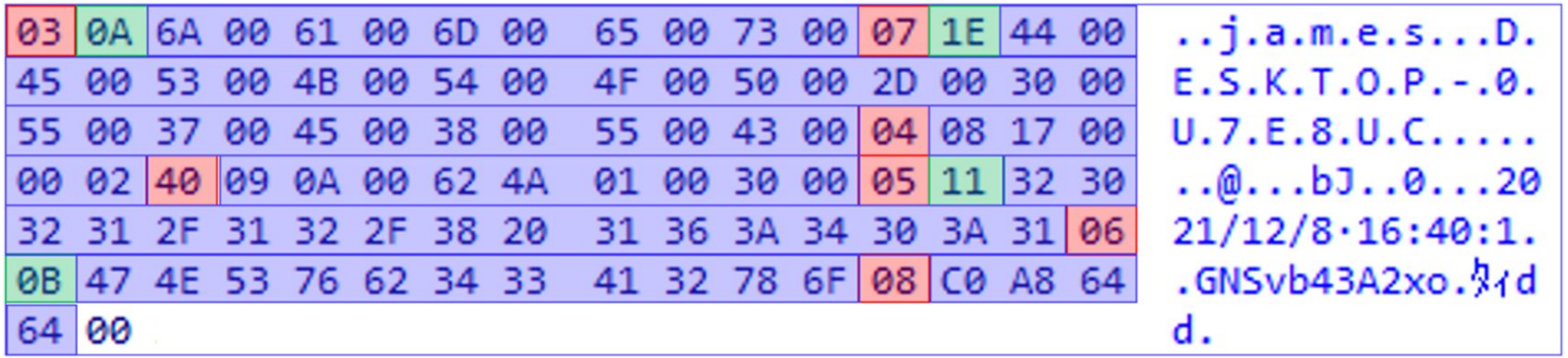
```
; command command_list[]
command_list    dw 'c'                  ; id
                                        ; DATA XREF
                                        ; sub_18000
                db 6 dup(0)
                dq offset get_outlook_creds; functi
                dw 'd'                  ; id
                db 6 dup(0)
                dq offset exec_dll      ; function
                dw 'e'                  ; id
                db 6 dup(0)
                dq offset nullsub_2     ; function
```

The First Data Format Sent to SodaMaster C2

- Data chunk format

1 Byte	1 Byte	Length of data
ID	Size of data (If data length is variable)	Data

- The raw data before encryption



The First Data Sent to SodaMaster C2

Contains 7 types of data

ID	Length	Data
0x03	Variable	Username
0x07	Variable	Computer name
0x04	5 Bytes	PID, Privilege flag
0x40	9 Bytes	Processor architecture (1 byte), OS major version (2 bytes), OS build number (2 bytes), Legacy OS flag (e.g., Win2003 x64 = 0xFF10) (2 bytes), OS product type (2 bytes)
0x05	Variable	Date of execution (yyyy/mm/dd hh:mm:ss)
0x06	Variable	RC4 encryption key for C2 communication
0x08	4 Bytes	Socket name

Encryption Process for The First Data

1. RSA encryption
   - Base64 encoded public key is hardcoded
   - The public key is different in each sample
2. Inverting encrypted data
3. Adding junk data at the end (ver.2 or later)
   - Two types of calculation methods for the size of the junk data were observed:
     i. (address of encrypted data + address of collected data from victim) % 0x50 + 5
     ii. (address of encrypted data + returned value of GetTickCount) % 0x50 + 5
   - Add data extracted from encrypted data by unique algorithm to the end

Encrypt by RSA + Invert

```
64 03 F2 FF 81 F1 DA 30  CD 82 44 74 ED 94 33 51
E0 43 E0 F9 F8 B2 81 7B  6F 3B 50 F2 8C 66 EF DD
F5 F1 D0 74 AC F4 FC 4F  1F 47 01 AA 89 91 0F 96
2A D9 D6 74 73 DD 50 1D  5D 74 3C F3 4A D2 9B F0
66 C6 38 A0 30 28 9C D9  C8 89 22 6C 42 EB 82 EA
D2 91 3B 0A F3 48 9A 42  FE 92 8C B4 08 F1 98 B7
2D D6 EB 2A 30 32 C4 91  E8 87 DA A1 E1 CF 01 D5
5A 82 90 24 8E ED 1A 73  BB 2D E0 A0 9E D8 C3 1C
```

Unnecessary data added to the end of data

```
   9A 82 ED EB 98 8C 51  8C 6F 3B 91 28 8C 9C DD
8C FF 6C F1 94 28 C8 C4  3C 0F 42 33 30 C4 82 28
FC 30 1D 42 51 74 4F 8C  74 9B 89 FC E0 28 30 F3
EA A0 51 94 73 EF 89 D6  B7 0F 4A DD 4F FC C8 66
7B 74 47 91 3C E0 1D 43  00 00 00 00 00 00 00 00
```

New TTPs Observed in 2021

Continuous A41APT Campaign in 2021

Jackpot Webshell

- Webshell malware firstly reported by Trend Micro with their deep analysis
  - It was used as a payload of SigLoader in 2021
- Works as a standalone HTTP sever
  - Jackpot receives commands via a POST request for the specific URL
    - A domain/IP address of victim organization is hardcoded
  - Jackpot tends to be found at the IIS servers, because the infected host must be the internet-facing server
    - Even if the IIS service is running, Jackpot works on the same port

Microsoft Exchange Server

Exploiting ProxyShell vulnerability

- The following commands were observed on a PowerShell session obtained by the exploit
  - copy
  - dir
  - ipconfig /all
  - net user /domain
  - net time /domain
  - ping
  - query user
  - type
  - tasklist
  - whoami
  - wmic /node:<ip address> process call create cmd /c Dnscmd /EnumZones ><output file>
  - [System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String('<string>')) | Out-File -FilePath <File path>
- China Chopper webshell was installed after above activities

HUI Loader

- We discovered another loader used for loading SodaMaster in 2021
  - Unnamed loader that has been observed since 2015 for various payloads
  - Named after string "HUIHWASDIHWEIUDHDSFSFEFWEFEWFDSGEFERWGWEEFWFWEWD"

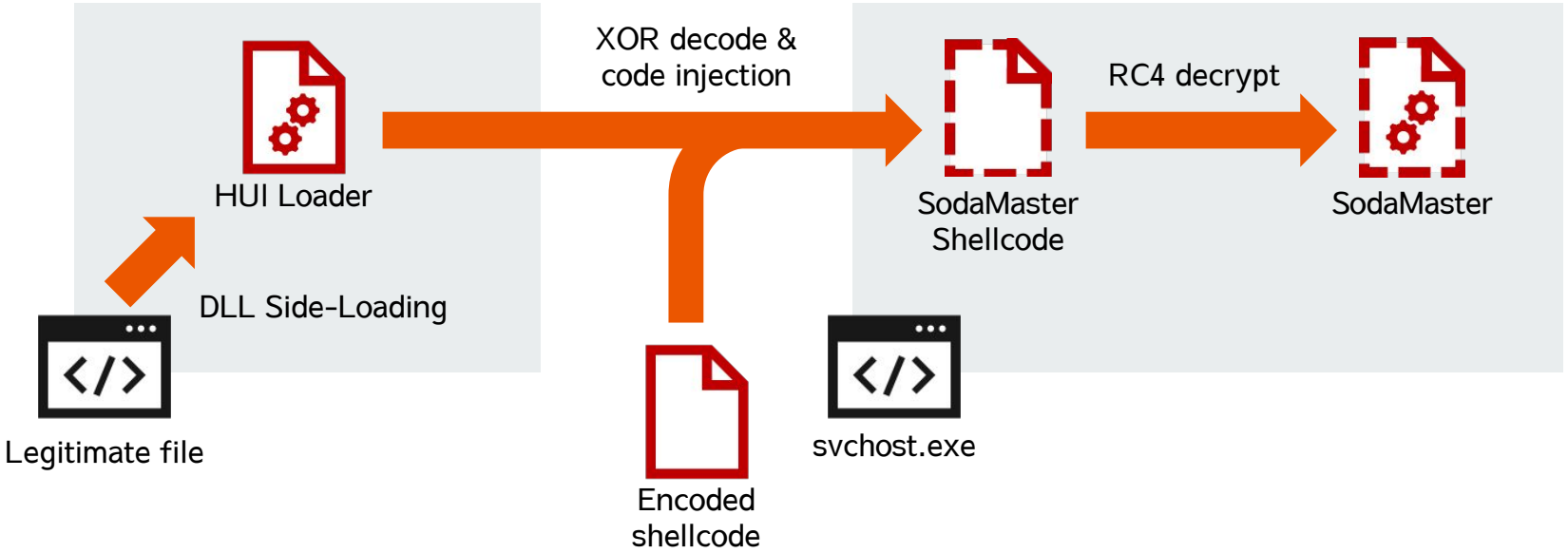
```
lea     r8, aCWindowsInstal ; "c:\\windows\\install.log"
lea     rcx, FileName   ; lpFileName
db      66h, 66h, 66h, 66h
nop     word ptr [rax+rax+00000000h]


                        ; CODE XREF: aa_open_install_log
movzx   eax, word ptr [rdx+r8]
add     rdx, 2
mov     [rdx+rcx-2], ax
test    ax, ax
jnz     short loc_180001480
xor     r9d, r9d        ; lpSecurityAttributes
mov     [rsp+48h+hTemplateFile], 0 ; hTemplateFile

mov     edx, 80000000h  ; dwDesiredAccess
lea     r8d, [r9+2]     ; dwShareMode
mov     [rsp+48h+dwFlagsAndAttributes], 80h ; '€' ; dwFl
mov     [rsp+48h+dwCreationDisposition], 3 ; dwCreationD
call    cs:CreateFileW
```

```
mov     eax, 'v'
lea     rcx, [rbp+300h+Buffer] ; lpBuffer
mov     [rsp+400h+filename+4], ax
mov     eax, 'c'
mov     edx, 104h       ; uSize
mov     [rsp+400h+filename+6], ax
mov     eax, 'h'
mov     dword ptr [rsp+400h+filename], 73005Ch ; "\s"
mov     [rsp+400h+filename+8], ax
mov     eax, 't'
mov     dword ptr [rsp+400h+filename+0Ah], 73006Fh ; "os"
mov     dword ptr [rsp+400h+filename+10h], 65002Eh ; ".e"
mov     dword ptr [rsp+400h+filename+14h], 650078h ; "xe"
mov     [rsp+400h+filename+18h], si
mov     [rsp+400h+filename+0Eh], ax
call    cs:GetSystemDirectoryW
xor     eax, eax
```

Execution Flow of HUI Loader



Decoding Process of HUI Loader

XOR decode by the hardcoded unique key

```
flOldProtect = 0;
memset(Buffer, 0, sizeof(Buffer));
qmemcpy(xor_key, "Z:/9#4AUG. \"BN1T1\\KD0HM(<Y7![1", 30);
encrypted_payload = CreateFileW(&FileName, 0xC0000000, 3u, 0, 3u, 0x
file_mapping = CreateFileMappingW(encrypted_payload, 0, 4u, 0, 0, 0)
if ( !file_mapping )
  exit(0);
_swprintf(Buffer, L"HUIHWASDIHWEIUDHDSFSFEFWEFEWFDSGEFERWGWEEFWFWEWD
payload_size = GetFileSize(encrypted_payload, 0);
CloseHandle(encrypted_payload);
map_view = MapViewOfFile(file_mapping, 4u, 0, 0, 0);
heap_size = payload_size + 50;
ProcessHeap = GetProcessHeap();
payload_data = HeapAlloc(ProcessHeap, 8u, heap_size);
VirtualProtect(payload_data, payload_size + 50, 0x40u, &flOldProtect
memcpy(payload_data, map_view, payload_size + 1);
UnmapViewOfFile(map_view);
size = payload_size;
data = payload_data;
for ( i = 0; i < size; ++i )
  data[i] ^= 0x20u;
for ( j = 0; j < size; ++j )
  data[j] ^= xor_key[j % 0x1Eu];
Sleep(0x1F4u);
aa_inject();
```

```
...: "5\"+9> 1VQ:6\\BG)F@;=GJ5POS-2%S(]&\"AY! 3Z@D5-KL
...: ")J<K(D%L6<BU/CJ0;2 (NH=3DAWQQG [>@1I+PK*S7,GDB\\
...: "I'B]ETP?1HBU50,'+OCZA/M\"C2YO-9J5UYAD5Z]*(W\"9EB
...: "];AHOMME<IY=+0 T*+J.-1=-)L9B=1\\I;'],O;1U)3*2TDB
...: "9!@CA?M[)[S5A4L2]T*FUHC9I&C+0]UALICQ7[E; Z(0?.:A
...: "sh7wh6gz36i2g692gwuk265qa",
...: "[F*[,.27PAIYCV0H#ND,DHNU.5TN$P?0,J8J0&O9O<]!A8(&
...: ]

In [2]: len(KNOWN_KEYS)
Out[2]: 20
```

```
def hui_decode(enc, key):
    key = bytearray(key.encode())
    dec = bytearray()
    for i in range(len(enc)):
        payloadbyte = enc[i] ^ 0x20 ^ key[i%len(key)]
        dec.append(payloadbyte)
    return dec
```

Attribution of The Threat Actor

Continuous A41APT Campaign in 2021

View of Trend Micro and Kaspersky

Linking to BRONZE RIVERSIDE (APT10) ?

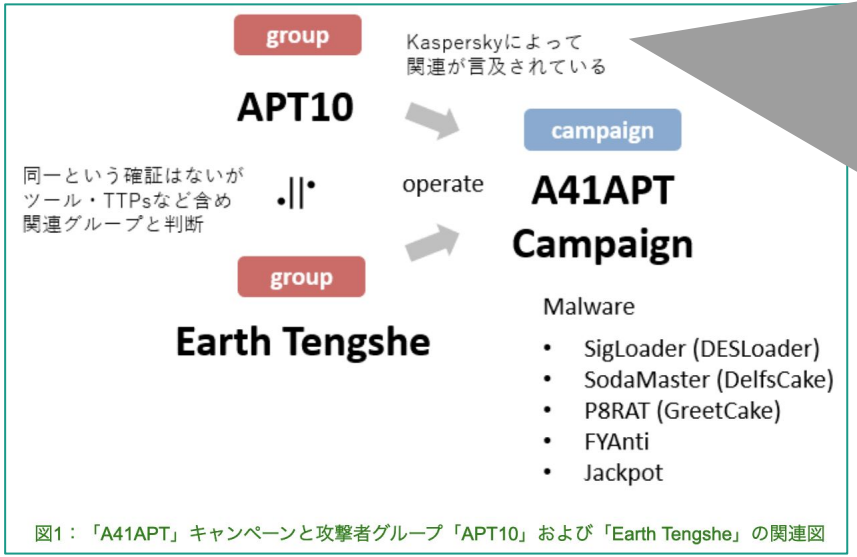


図1：「A41APT」キャンペーンと攻撃者グループ「APT10」および「Earth Tengshe」の関連図

調査結果に基づき、私たちはA41APTの活動の背後にAPT10が存在することにはかなりの確度があると考えています。裏付けとなるのは以下のポイントです。

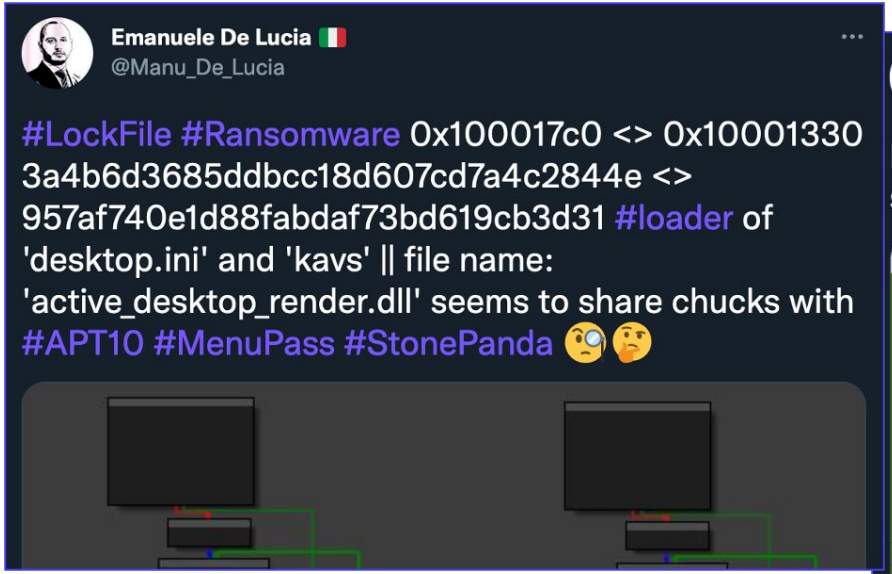
第1に、x86 SodaMaster検体にハードコードされた「www.rare-coisns[.]com」というURLが、ADEO IT Consulting Servicesによるレポート（英語）の中で言及されています。同レポートは、トルコの金融および電気通信セクターを標的とするAPT10の活動に関するもので、VirusTotalへの提出があった地理位置情報とも一致します。

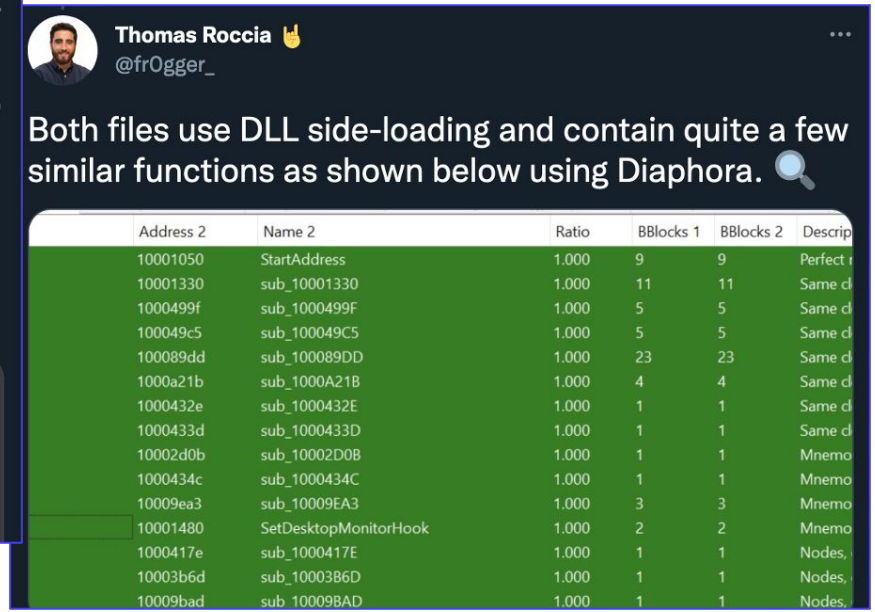
第2に、A41APTの攻撃活動とAPT10の活動との類似性は、Cylanceのブログ記事（英語）で説明されています。記事中ではEcipekac、FYAntiのユニークなエクスポート名である「F**kY**Anti」、CppHostCLRの使用、FYAntiの最終ペイロードとしてのQuasarRATについて触れられています。それだけでなく、Symantecのブログ記事（英語）にて言及されているFYAnti、「F**kY**Anti」というエクスポート名、.NETローダーの注入に使用されるCppHostCLR、QuasarRATも、BlackBerry Cylance Threat Research Teamによって発見されたAPT10グループの活動と類似しています。

このほか、私たちが過去に作成したAPT10 の活動に関するThreat Intelligence Portalレポートには、複数の類似性と共通のTTPが見られます。

A41APT, BRONZE RIVERSIDE and LockFile

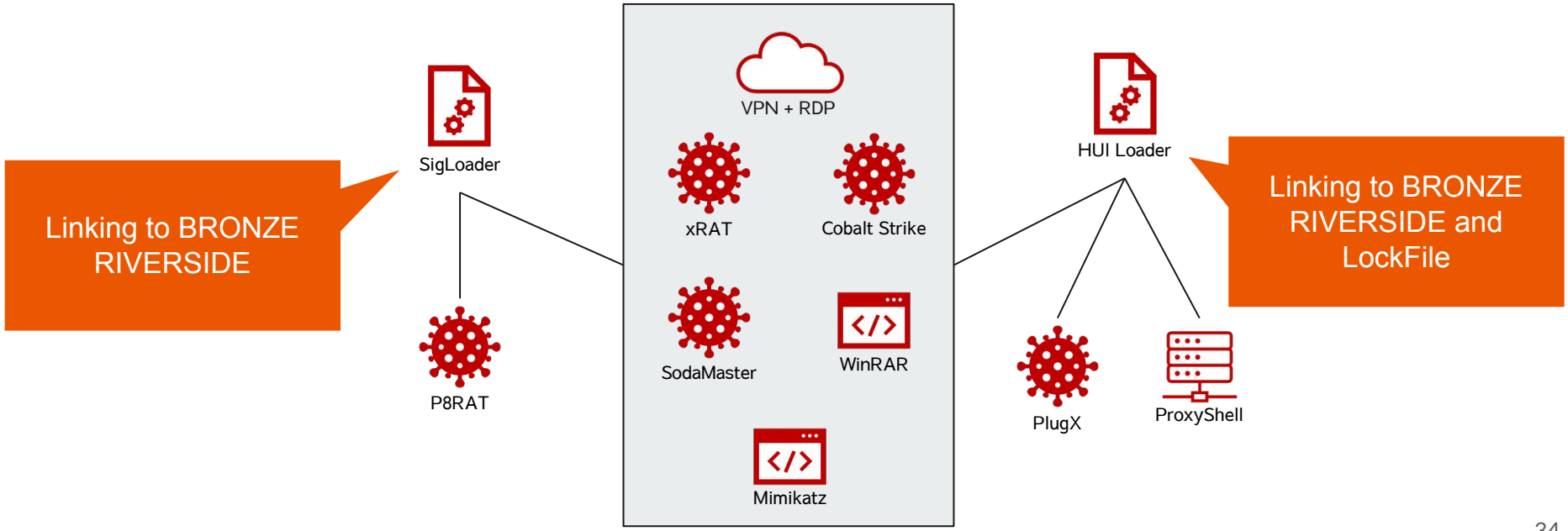
In August 2021, The HUI Loader was pointed out to be used with BRONZE RIVERSIDE and LockFile





Redefinition of Chaotic A41APT Campaign

Attribution should not be done only by the malware/tools used - but it's likely that the actor is based in China



What We Can Do against the Chaotic A41APT Campaign

Challenge to Know Your Own Organization

Fighting against Opportunistic Compromise, Targeted Deployment

- The actor attempts to compromise every organizations who seem to be related to their goals, then the actor will choose (an) organization(s) from among the victims as a start point
  - Not only HQ, subsidiaries and overseas branches will be affected
  - Incidents will happen at organizations who don't have enough security controls for internet-facing systems
- Do you have a true understanding of your organization from security perspective?
  - Infrastructure/Security controls of overseas branches, subsidiaries
    - Different systems from HQ
    - Low-budget security controls
    - Network/System sharing between HQ and subsidiaries
  - Leave maintenance and operation of system SI vendors
  - Network management, Account management, Endpoint management, etc.

Nothing changed from 2020

Conclusion

- Chaotic A41APT campaign
  - The campaign is still ongoing and expanding its TTPs
  - Multiple threat groups seem to be involved
  - The actor always intruded via Internet-facing systems
- Countermeasure should be the same with post-intrusion ransomware attacks
  - Protect internet-facing systems of whole your company including branch offices and subsidiaries
    - Cooperate with SI vendors
  - Detect usage of hacking tools or AD related tools for lateral movement after establishing C2
  - Hunt the threats by using EDR, auditing various logs, checking ASEP
- Information sharing like this would be helpful for everyone?
  - Difficult to reveal a whole picture of a campaign by a single vendor
  - Organizing information can preserve the anonymity of victims

Reference Regarding A41APT

1. A41APT case ~ Analysis of the Stealth APT Campaign Threatening
   - https://jsac.jpcert.or.jp/archive/2021/pdf/JSAC2021_202_niwa-yanagishita_en.pdf
   - https://jsac.jpcert.or.jp/archive/2021/pdf/JSAC2021_202_niwa-yanagishita_jp.pdf
2. APT10: sophisticated multi-layered loader Ecipekac discovered in A41APT campaign
   - https://securelist.com/apt10-sophisticated-multi-layered-loader-ecipekac-discovered-in-a41apt-campaign/101519/
   - https://blog.kaspersky.co.jp/apt10-sophisticated-multi-layered-loader-ecipekac-discovered-in-a41apt-campaign/30393/
3. APT10: Tracking down the stealth activity of the A41APT campaign
   - https://media.kasperskydaily.com/wp-content/uploads/sites/86/2021/02/25140359/greatidea_A41_v1.0.pdf
4. 標的型攻撃の実態と対策アプローチ 第5版 日本を狙うサイバーエスピオナージの動向2020年度 - Macnica Networks, TeamT5
   - https://www.macnica.co.jp/business/security/manufacturers/files/mpressioncss_ta_report_2020_5.pdf
5. 「Earth Tengshe」によるマルウェア「SigLoader」を用いた攻撃キャンペーンで観測された新たなペイロード
   - https://blog.trendmicro.co.jp/archives/29842

Other References

1. Uncovering New Activity By APT10 | FortiGuard Labs
   - https://www.fortinet.com/blog/threat-research/uncovering-new-activity-by-apt-
2. Insights into Ransomware Spread Using Exchange 1-Day Vulnerabilities 1-2 - NSFOCUS, Inc.,
   - https://nsfocusglobal.com/insights-into-ransomware-spread-using-exchange-1-day-vulnerabilities-1-2/
3. Twitter
   - https://twitter.com/Manu_De_Lucia/status/1430115616862638080
   - https://twitter.com/fr0gger_/status/1430213808434339842
4. Guidance for preventing, detecting, and hunting for exploitation of the Log4j 2 vulnerability
   - https://www.microsoft.com/security/blog/2021/12/11/guidance-for-preventing-detecting-and-hunting-for-cve-2021-44228-log4j-2-exploitation/
5. AutoRuns
   - https://docs.microsoft.com/ja-jp/sysinternals/downloads/autoruns

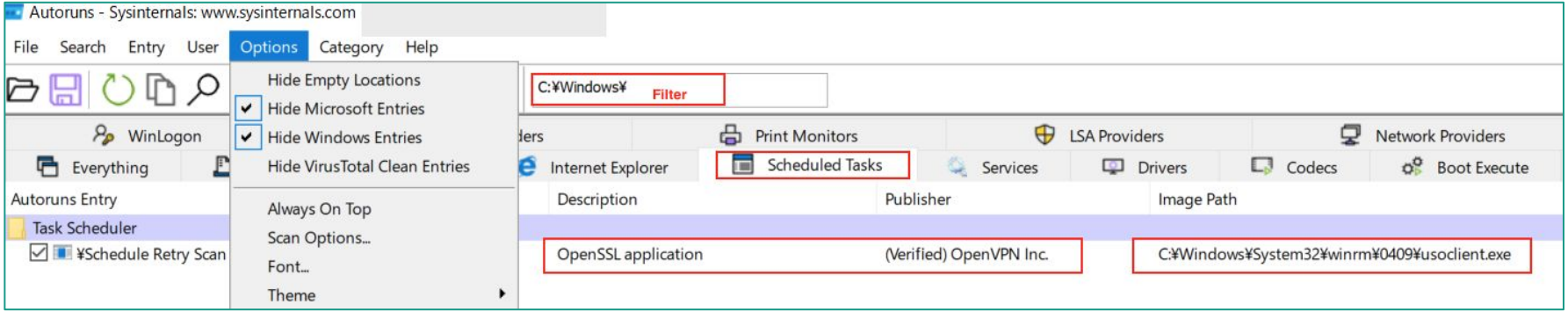
IoCs

Value	Type	Description
cf5ec3b803563d8ef68138f5303ebc362b72da36da29b9cba3062ae996db9234	SHA256	HUILoader
c13f93b7bb1f8f5f9bd6dd4d25f7af873119c8b8248490de6bd9b29d0c68783e	SHA256	Encoded SodaMaster shellcode
168.100.8.20	IP	SodaMaster C2
9bec85e6a3d811826580540b541723c6b5236377a3a980b1ffa5bf5f749a99d4	SHA256	HUILoader
7db327cc7bd622038f69b4df4178ca3145659a73cbcb10d0228e48f2ece60896	SHA256	Encoded SodaMaster shellcode
www[.]monferriina[.]com	Domain	Sodamaster C2
c0ed7939945726b61100009b926917723fdc5f9b2df0be070f2a500b6edf161c	SHA256	SigLoader (Layer I)
0a570b32d14799f6351ee211093567450d41705ca79e236a38ca15f135d78bfd	SHA256	SigLoader (Layer I)
2da5e37ec4c7059a7935165039ea31b0c9cc8f1bb0d0c620759776979158cf30	SHA256	SigLoader (Layer I)
e8797b4334fbaa067d5f91d1481bd8f55bf2e45483a92a8ea7030c2c604dd273	SHA256	SigLoader (Layer I)
68dd499bca62e004c97ccc17f68e3d6dde2885446924dabe8cc525763caa08a3	SHA256	Encrypted SodaMaster shellcode
192.248.183.113	IP	SodaMaster C2
1f1bcb03b008c4fdd462e7d2b5db5ca321ff6d56bbb22cddd39c82df1f6a038f	SHA256	DESLoader (1st Loader)
7337071599eb49c75c63dff210aa516ea8dbbe99a8a66237f66f3f3c7f5aed31	SHA256	Encrypted SigLoader shellcode
59986e20e03774c7d0f5adb4eca394f5f51b01a8c2ba9cb6c1ce30f9312b9566	SHA256	Encrypted SodaMaster shellcode
185.10.16.115	IP	SodaMaster C2
8efcecc00763ce9269a01d2b5918873144746c4b203be28c92459f5301927961	SHA256	HUILoader in 2015
20fc3cf1afcad9e6f19e9abebfc9daf374909801d874c3d276b913f12d6230ec	SHA256	Mimikatz

FYI: Hunting Suspicious ASEP

When EDR and Forensic Tools are not ready

- Audit ASEP by using tools such as Autoruns is effective
- In A41APT campaign, scheduled tasks are favor to be used
  - Investigating scheduled tasks with the following condition could be useful
    - 3rd party legitimate executables under C:\Windows\, C:\Intel\



IOCs - Examples of Scheduled Task

Program	Descritpion	Publisher
C:\Windows\RoutineMaintenance.exe		D3L
C:\Windows\ceiprole.exe	Malwarebytes Anti-Exploit 64bit tasks	Malwarebytes Corporation
C:\Windows\Vss\Writers\System\FamilySafety.exe	Java(TM) Platform SE binary	International Business Machines Corporation
C:\Windows\System32\winrm\0409\usoclient.exe	OpenSSL application	OpenVPN Inc.
C:\Windows\System32\da-DK\DataProviders.exe	OpenSSL application	OpenVPN Inc.

In other cases, 3rd party legitimate executables such as VMware Tools, Sandboxie that should be under \Program Files\ folder are installed under C:\Windows, C:\Intel\

Thank you