



カオス化する A41APT キャンペーン に対して私達ができること

柳下 元、玉田 清貴、中津留 勇、石丸 傑

2022年1月27日

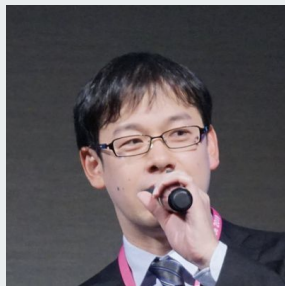


著者紹介



柳下 元

株式会社マクニカ
ネットワークス
カンパニー



玉田 清貴

セキュアワークス
株式会社



中津留 勇



石丸 傑

株式会社
カスペルスキー



目次

今の A41APT の情報を持ち寄り共有

- A41APT キャンペーンとは
- 2021年も継続する A41APT キャンペーン
 - 継続・アップデートされた TTPs
 - 2021年に初めて観測された TTPs
 - 攻撃グループの考察
- カオス化する A41APT キャンペーンに
対抗するために

A41APT キャンペーンとは

A41APT キャンペーン概要

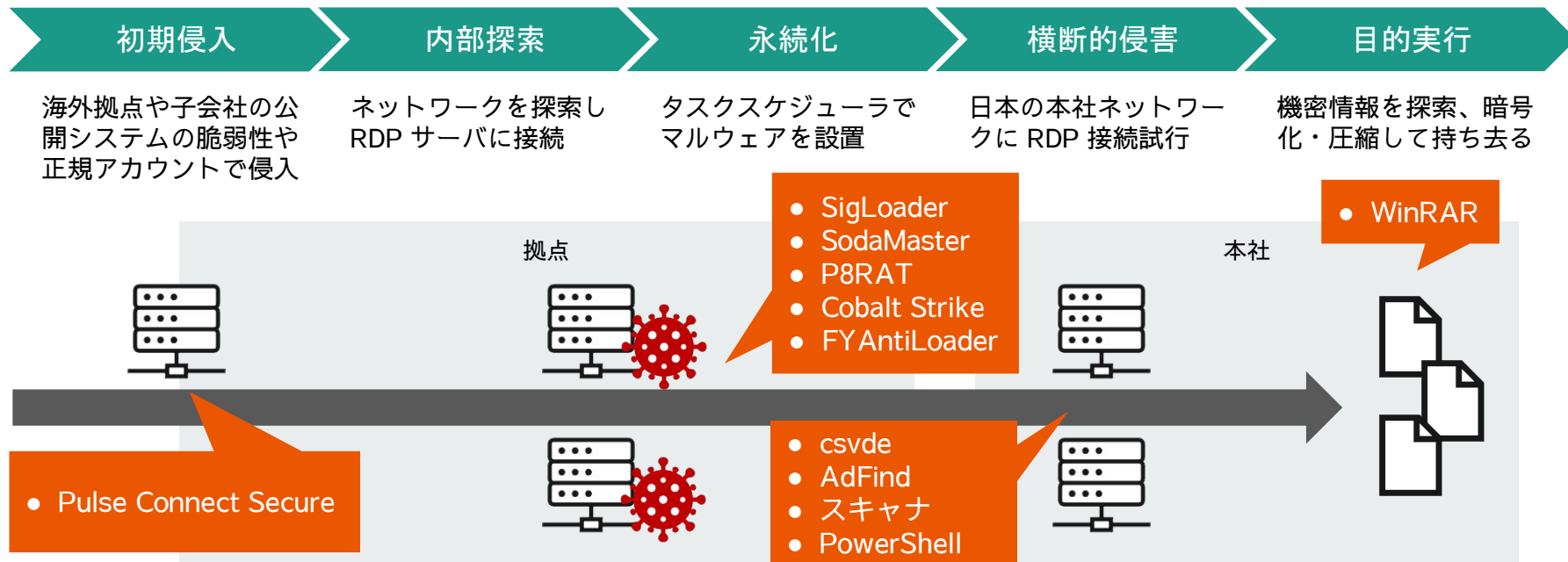
JSAC 2021 で公表された攻撃キャンペーン

- 活動時期
 - 2019年3月から2021年1月
- 被害組織
 - 日本 (日系企業 / 日系企業の海外拠点)
- 名称の由来
 - 接続元の端末名 DESKTOP-A41UVJV から A41APT と命名



公開システムから侵入し、SigLoader や SodaMaster といったマルウェアを展開する攻撃キャンペーンを「A41APT キャンペーン」と区別

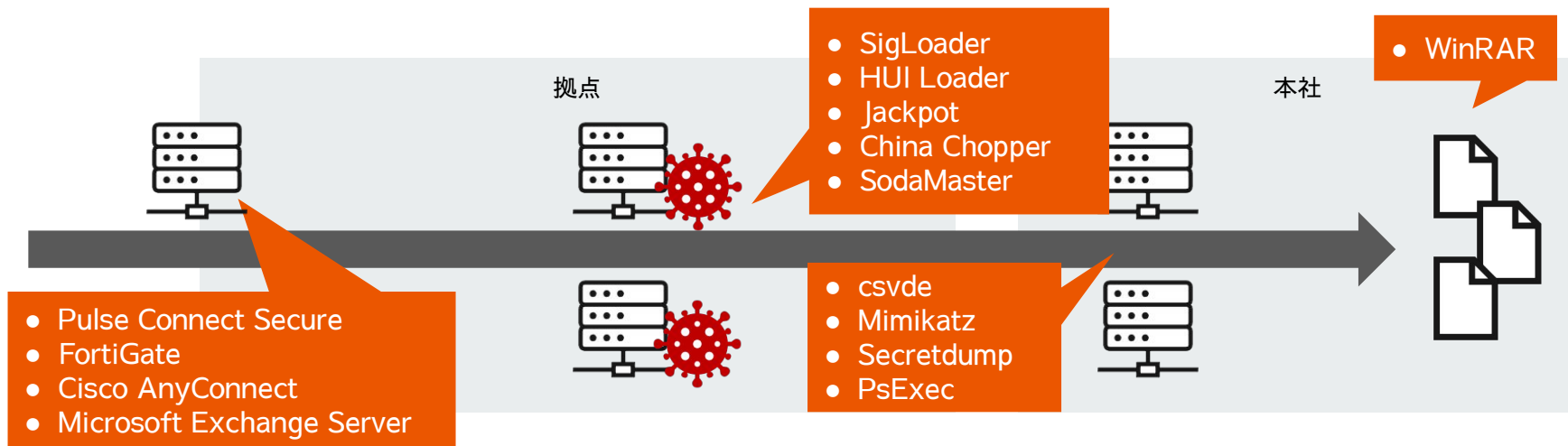
明らかにされた TTPs



2021年も継続する A41APT キャンペーン

2021年も継続していた A41APT キャンペーン

- 日本国内を含む複数の組織において攻撃が行われたことを確認
 - 2020年から継続・アップデートされた TTPs や、初めて確認された TTPs など、著者それぞれが異なる複数のインシデントを調査



トレンドマイクロ社により公開された情報

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

「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。

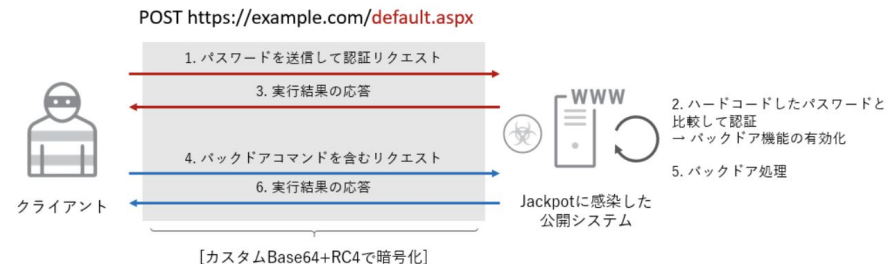


図8：Jackpotがバックドア機能を有効化するためのシーケンス例

継続、アップデートされた TTPs

2021年も継続する A41APT キャンペーン

VPN デバイス経由の侵入

2021年に発生したインシデントで観測

- 既知の脆弱性の悪用が目立つ
 - Pulse Connect Secure
 - FortiGate: CVE-2018-13379
 - Cisco AnyConnect: CVE-2020-3125
- 今はパッチが適用されていても、当時の認証情報は漏れているかもしれない

DESKTOP-A41UVJV は見られず、
別途観測されていた DESKTOP-O2KM1VL が
継続して使用されていることを確認している
(A41APT の由来が消えた)



侵入後に使用するツールセット

- 組織内部での認証情報窃取や横断的侵害などに使われたツールにいくつか変化が見られる
 - Mimikatz
 - secretdump.py
 - PsExec
 - csvde
 - WinRAR
- よく使われるツールを必要に応じて取り入れていると考えられる

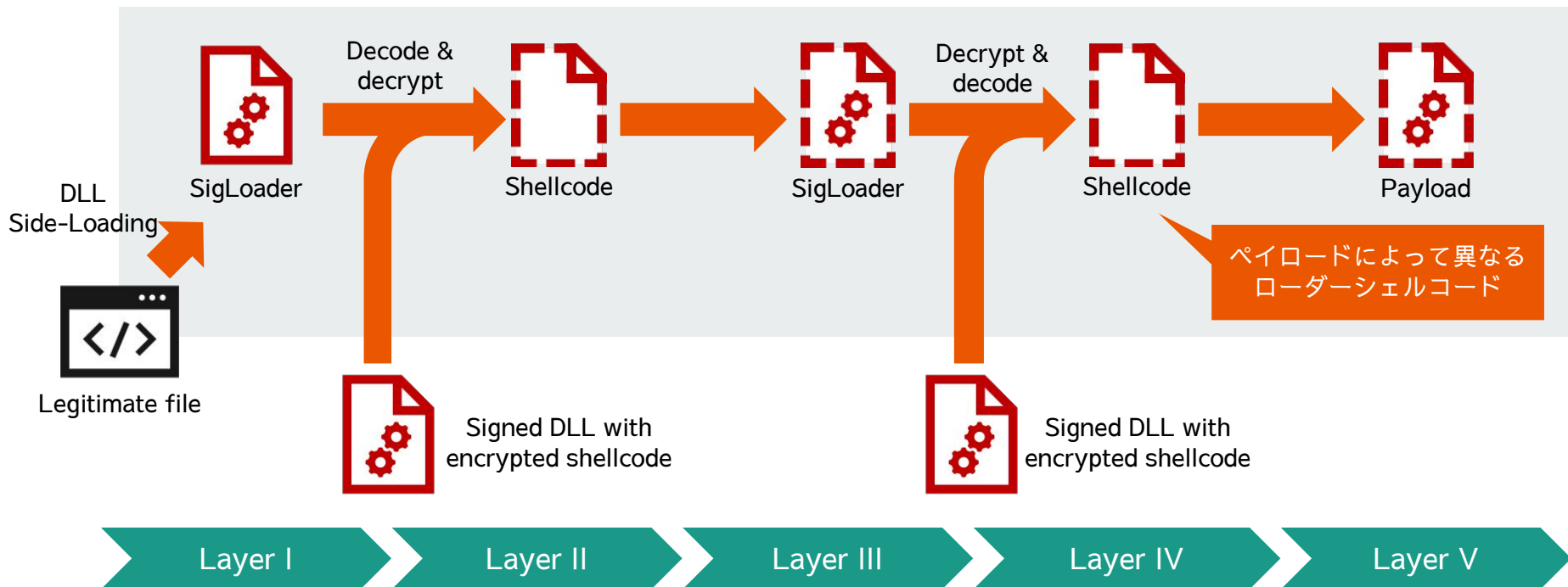


マルウェアのアップデート

- 継続して SigLoader と SodaMaster が使用されている
 - 一方で Cobalt Strike, P8RAT, FYAntiLoader は見られていない
- SigLoader と SodaMaster には変化が見られる
 - コンパイル時刻の改ざん
 - SigLoader: 2021/?? -> 2017/05 など
 - SodaMaster: 2021/04 -> 2012/10 など
 - 仕様変更や機能追加
 - SigLoader: 復号処理の変化
 - SodaMaster: コマンドやデータフォーマットの変化



SigLoader の変わらぬ流れ



SigLoader の復号処理の変化

- 復号アルゴリズムの識別子が文字列から数値へと変更
 - 0: AES
 - 1: DES
 - 3: XOR
- 復号アルゴリズムの順番は別途ハードコード

```
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

確認した Layer I と Layer III それぞれの
SigLoader はアルゴリズムが逆順に設定されていた

```
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__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
```

SigLoader の復号処理の変化

- AES モードは CBC から ECB へ変更
- 32文字の先頭16文字を暗号鍵として使用

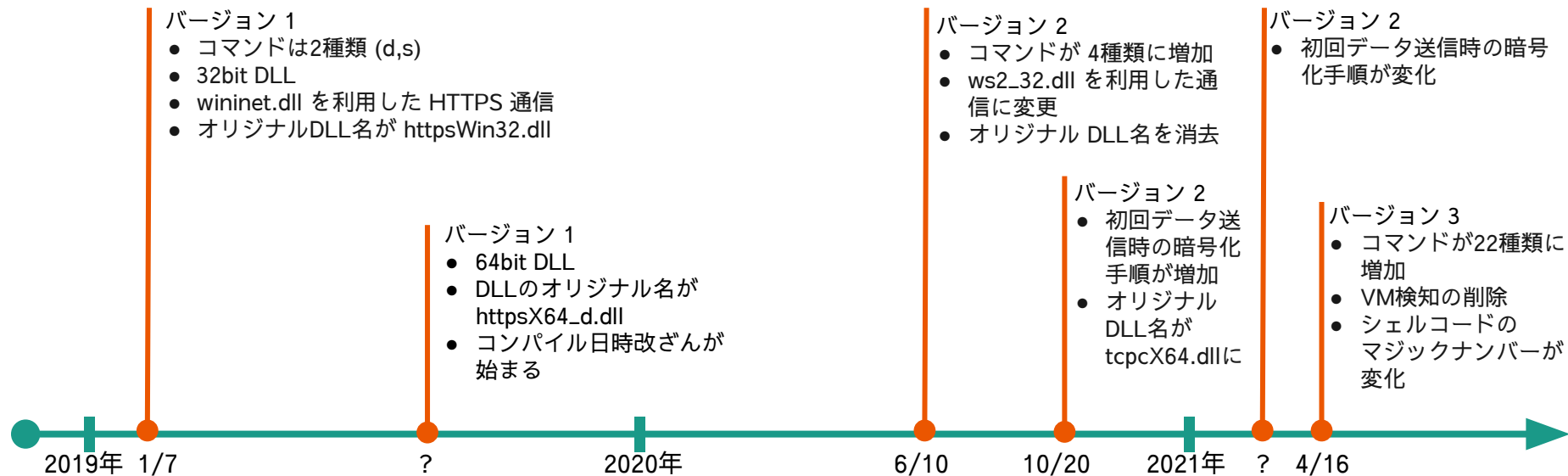
```
FC_Rijndael_sbox db 63h, 7Ch, 77h, 7Bh, 0F2h, 6Bh, 6Fh, 0C5h, 30h, 1, 67h
; DATA XREF: aes_init_180001EE0+9C+o
db 2Bh, 0FEh, 0D7h, 0ABh, 76h, 0CAh, 82h, 0C9h, 7Dh, 0FAh
db 59h, 47h, 0F0h, 0ADh, 0D4h, 0A2h, 0AFh, 9Ch, 0A4h, 72h
db 0C0h, 0B7h, 0FDh, 93h, 26h, 36h, 3Fh, 0F7h, 0CCh, 34h
db 0A5h, 0E5h, 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;
    ( 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 の変遷

観測した 20+ の検体をコンパイル日時の共通点から 3つのバージョン、6回の活動に分類



各 SodaMaster の比較

※ 改ざんが疑われるコンパイル日時は薄灰色

	バージョン 1		バージョン 2			バージョン 3
コンパイル日時	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
ファイル形式	x86 DLL	x64 DLL	x64 DLL			x64 DLL
オリジナル DLL 名	httpsWin32.dll	httpsX64_d.dll	-	tcpcX64.dll		tcpcX64.dll
エクスポート関数	DLLEntry		-	DLLEntry		DLLEntry
ネットワーク API	wininet		ws2_32			ws2_32
コマンド	d, s		d, f, l, s			c - x
VM 検知	✓		✓			-
末尾に追加する 不要データの サイズ計算	-		-	収集した端末情報 の格納アドレスを 使用	GetTickCount を 使用	GetTickCount を 使用

SodaMaster 用シェルコードの変化

過去報告されていたシェルコードと動作は変わらず

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

- バージョン 3 では、magic bytes が更新
- SodaMaster の更新に伴いデータのサイズも増加

offset	data	description
0x000	90 90 90 90 90 90 90 90	識別用の8バイトのmagic bytes、データ処理を行う前に比較し確認
0x008	0x11600	暗号化されたデータのサイズ 現状観測した検体はすべて同じ値
0x00C	A9 5B 7B 84 9C CB CF E8 B6 79 F1 9F 05 B6 2B FE	16 bytes RC4 key (検体毎に異なる)
0x01C	C7 36 7E 93 D3 07 1E 86 23 75 10 49 C8 AD 01 9F [skipped]	RC4で暗号化されたSodaMasterのパayload

SodaMaster の最新コマンド一覧

※ e, j, k, n-p, t-vは未実装

※ 赤字はトレンドマイクロの解析結果との差分

コマンド	動作
c	Microsoft Outlook の認証情報の窃取
d	DLL の実行（指定したエクスポート関数）
f	送信データへのサイズ情報追加の有効化/ 無効化
g	シェルコードの実行（関数テーブル無）
h	指定の接続先へ送信元を偽装したパケット 送信を繰り返す (DoS?)
i	指定の接続先へ 0x20000バイトの 0xCC で 埋められたデータ送信を繰り返す (DoS?)

コマンド	動作
l	C2 との通信間隔の設定
m	スクリーンショットの取得
q	キーロガーの有効化
r	キーロガーの無効化
s	シェルコードの実行（関数テーブル有）
w	メッセージボックスで文字列を表示
x	プロセス終了

コマンド増加に伴う SodaMaster コマンド処理の変化

数が多くなったことにより Case 文を廃止

バージョン 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;
}
```

バージョン 3

```
j = 0i64;
if ( command_list[0].function != 0 )
{
    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_list
command_list dw 'c' ; id
; DATA XREF
; sub_18000

db 6 dup(0)
dq offset get_outlook_creds; function
dw 'd' ; id
db 6 dup(0)
dq offset exec_dll ; function
dw 'e' ; id
db 6 dup(0)
dq offset nullsub_2 ; function
dw 'f' ; id
```

SodaMaster の初回送信データのフォーマット

- 送信データチャンクのフォーマット

1 Byte	1 Byte	データサイズ分
ID	データサイズ (可変長の場合のみ)	データ

- 実際の送信データ (暗号化前)

03	0A	6A 00 61 00 6D 00 65 00 73 00	07	1E	44 00	..j.a.m.e.s...D.
45 00 53 00 4B 00 54 00 4F 00 50 00 2D 00 30 00						E.S.K.T.O.P.-.0.
55 00 37 00 45 00 38 00 55 00 43 00	04	08 17 00				U.7.E.8.U.C.....
00 02 40 09 0A 00 62 4A 01 00 30 00	05	11 32 30				..@...bJ..0...20
32 31 2F 31 32 2F 38 20 31 36 3A 34 30 3A 31	06					21/12/8.16:40:1.
0B 47 4E 53 76 62 34 33 41 32 78 6F	08	C0 A8 64				.GNSvb43A2xo.7id
64 00						d.

SodaMaster の初回送信データの内容

7種類のデータを送信

ID	サイズ	データ
0x03	可変	ユーザ名
0x07	可変	コンピュータ名
0x04	5 バイト	PID, 権限フラグ
0x40	9 バイト	プロセッサのアーキテクチャ (1 バイト), OS メジャーバージョン (2 バイト), OS ビルド番号 (2 バイト), レガシー OS フラグ (例: Win2003 x64 = 0xFF10) (2 バイト), OS プロダクトタイプ (2 bytes)
0x05	可変	実行日時 (yyyy/mm/dd hh:mm:ss)
0x06	可変	C2 通信の暗号化で使用するRC4鍵
0x08	4 バイト	ソケット名

SodaMaster の初回送信データの暗号化

1. RSAで暗号化
 - Base64 エンコードされた公開鍵がハードコード
 - 検体ごとに公開鍵は異なる
2. 暗号化後のデータを反転
3. 末尾に不要なデータを追加（バージョン2以降）
 - 追加データのサイズ計算（検体によって方法が異なる）
 - i. (暗号化後データが格納されたアドレス + 感染端末から取得した情報が格納されたアドレス) % 0x50 + 5
 - ii. (暗号化後データが格納されたアドレス + GetTickCount の戻り値) % 0x50 + 5
 - 特定のアルゴリズムで暗号化後のデータから1バイトずつ抽出して末尾に追加

RSA 暗号化 + 反転

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								

末尾に追加される不要なデータ

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	00	00	00	00	00	00	00	00	00	00	00

これまで報告されていなかった TTPs

2021年も継続する A41APT キャンペーン

Jackpot Webshell

- トレンドマイクロ社が詳細解析結果を公表した Webshell
 - 2021年に新たに観測された SigLoader のペイロードの1つ
- スタンドアロンの HTTP サーバとしても動作
 - 特定 URL に接続された際に Webshell として動作
 - 被害組織のドメインがハードコードされている
 - 公開されているサーバでなければならいため、IIS サーバで発見されることが多い
 - IIS が稼動していても寄生する形で動作可能



Microsoft Exchange Server を起点とした侵入

ProxyShell の脆弱性を悪用

- 脆弱性の悪用得た PowerShell セッションで以下のコマンドを実行
 - copy
 - dir
 - ipconfig /all
 - net user /domain
 - net time /domain
 - 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 <ツール群の保存パス>
 - ping
 - query user
 - type
 - tasklist
 - whoami
- 最後に China Chopper Webshell を設置

HUI Loader の使用

- 2015年から存在していた名前の無いローダーが SodaMaster のローダーとして使用されていた
 - 文字列 "HUIHWASDIHWEIUDHDSFSFEFWFEWFDSEGEFERWGWEEFWFEWD" から命名

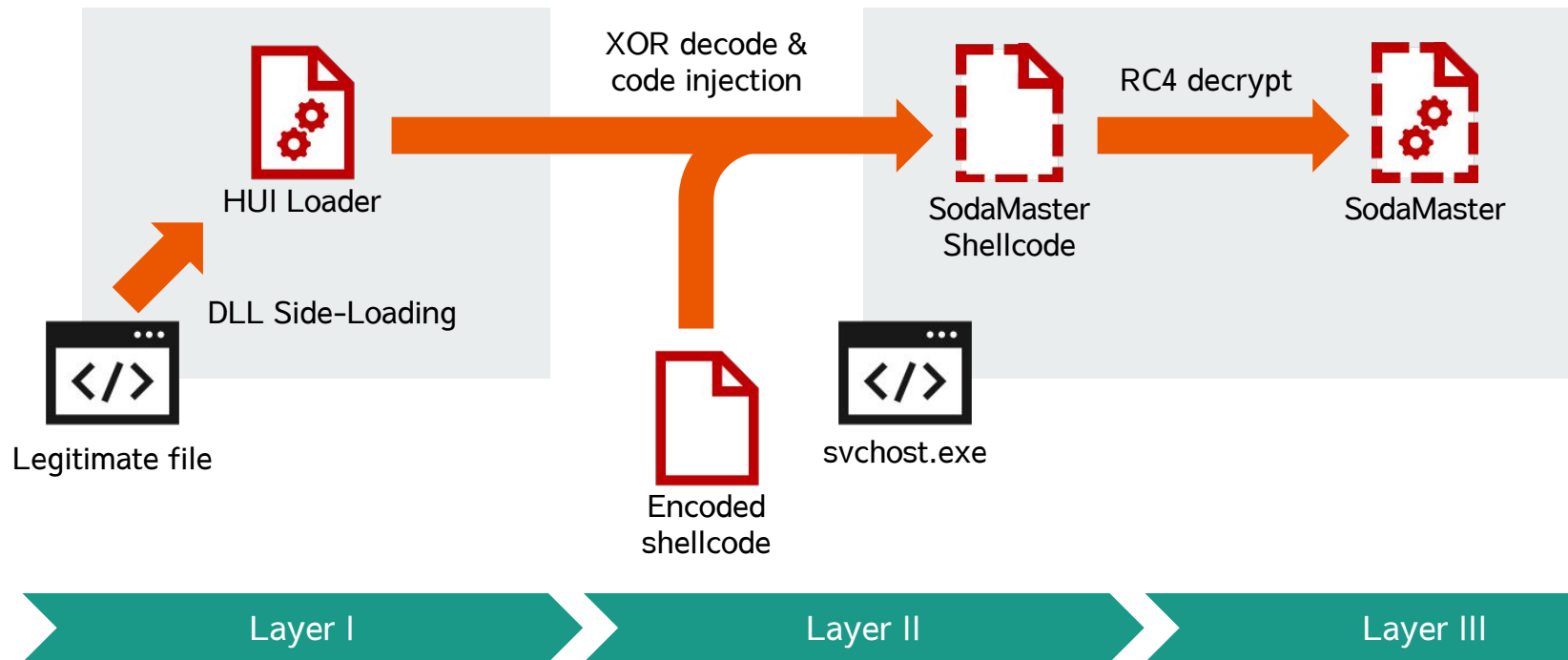
```
lea r8, aCWindowsInstall ; "c:\\windows\\install.log"
lea rcx, FileName ; lpFileName
db 66h, 66h, 66h, 66h
nop word ptr [rax+rax+00000000h]

; 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 ; 'E' ; 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
```

HUI Loader の流れ



HUI Loader のデコード処理

検体ごとに固有の鍵を使用して XOR を行う

```
fl0ldProtect = 0;
memset(Buffer, 0, sizeof(Buffer));
memcpy(xor_key, "Z:/9#4AUG. \\BN1T1\\KD0HM(<Y7![1", 30);
encrypted_payload = CreateFileW(&FileName, 0xC0000000, 3u, 0, 3u, 0x0, 0);
file_mapping = CreateFileMappingW(encrypted_payload, 0, 4u, 0, 0, 0);
if (!file_mapping)
    exit(0);
_swprintf(Buffer, L"HUIHWASDIHWEIUDHDSFSFEFEFEFEFDSGEFERWGWEEFEFEFEWWD
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, &fl0ldProtect);
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@;=GJ5PUS-2%S(J&\\\"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;' ],0;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&090<]!A8(&
.... ]
```

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

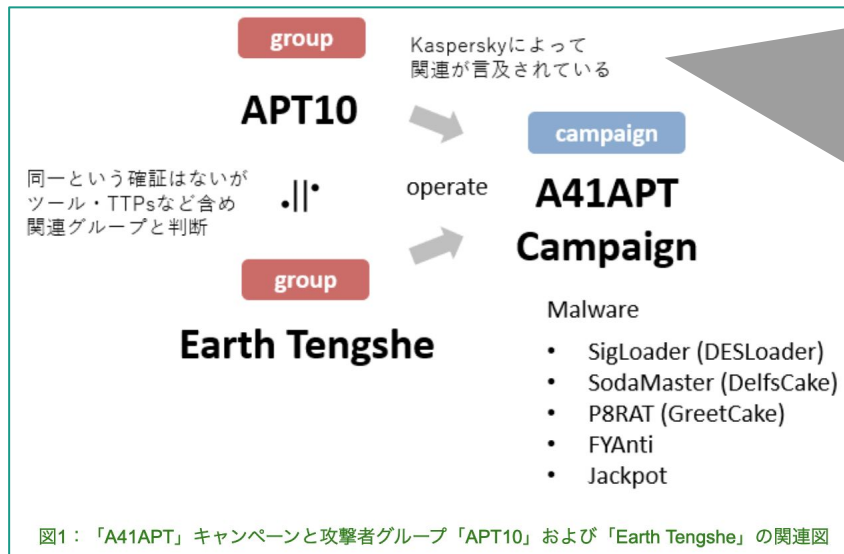
```
def hui_decode(enc, key):
    key = bytearray(key.encode())
    dec = bytearray()
    for i in range(len(enc)):
        payloadbyte = enc[i] ^ 0x20 ^ key[i%len(key)]
        dec.append(payloadbyte)
    return dec
```


攻撃グループの考察

2021年も継続する A41APT キャンペーン

これまでに言われている攻撃グループ像

BRONZE RIVERSIDE (APT10) に近い攻撃グループ？



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。

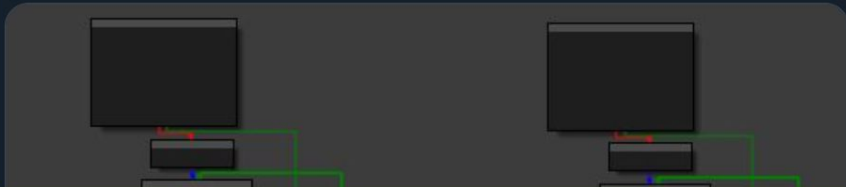
このほか、私たちが過去に作成したAPT10の活動に関するThreat Intelligence Portalレポートには、複数の類似性と共通のTTPが見られます。

A41APT と BRONZE RIVERSIDE と LockFile

2021年8月に HUI Loader が BRONZE RIVERSIDE と LockFile で使用されていると指摘

 **Emanuele De Lucia** 🇮🇹
@Manu_De_Lucia

#LockFile #Ransomware 0x100017c0 <> 0x10001330
3a4b6d3685ddbcc18d607cd7a4c2844e <>
957af740e1d88fabdaf73bd619cb3d31 #loader of
'desktop.ini' and 'kavs' || file name:
'active_desktop_render.dll' seems to share chunks with
#APT10 #MenuPass #StonePanda 🤔🤔



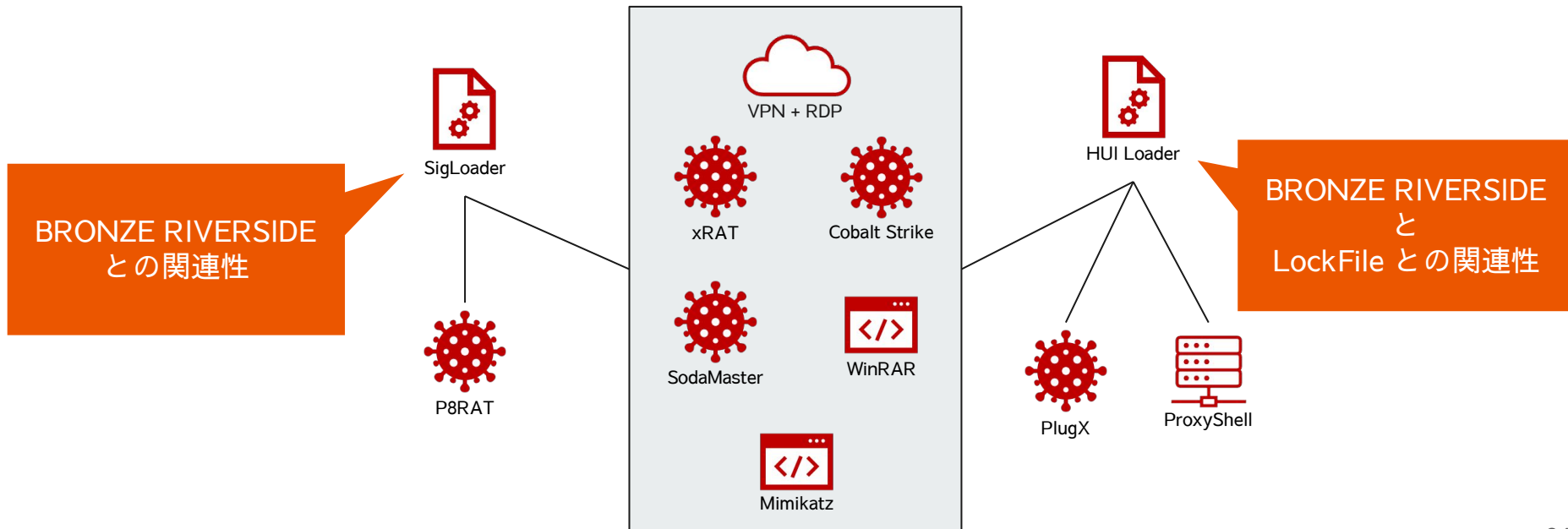
 **Thomas Roccia** 🙌
@frOgger_

Both files use DLL side-loading and contain quite a few similar functions as shown below using Diaphora. 🔍

Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Descrip
10001050	StartAddress	1.000	9	9	Perfect i
10001330	sub_10001330	1.000	11	11	Same d
1000499f	sub_1000499f	1.000	5	5	Same d
100049c5	sub_100049c5	1.000	5	5	Same d
100089dd	sub_100089dd	1.000	23	23	Same d
1000a21b	sub_1000a21b	1.000	4	4	Same d
1000432e	sub_1000432e	1.000	1	1	Same d
1000433d	sub_1000433d	1.000	1	1	Same d
10002d0b	sub_10002d0b	1.000	1	1	Mnemo
1000434c	sub_1000434c	1.000	1	1	Mnemo
10009ea3	sub_10009ea3	1.000	3	3	Mnemo
10001480	SetDesktopMonitorHook	1.000	2	2	Mnemo
1000417e	sub_1000417e	1.000	1	1	Nodes,
10003b6d	sub_10003b6d	1.000	1	1	Nodes,
10009bad	sub_10009bad	1.000	1	1	Nodes,

カオス化する A41APT キャンペーンの再定義

使用する独自マルウェアでは一括りにできないが、攻撃グループは中国に関連している可能性は高い



カオス化する A41APT に 対抗するために

自組織を「知る」ことへの挑戦

Opportunistic Compromise, Targeted Deployment 時代に生き残るために

- 攻撃者は、目的に関係する広範囲に攻撃を仕掛け、侵害できてからどうするか考える
 - 本社がダメなら子会社、子会社がダメなら海外拠点など
 - セキュリティ対策の甘い所から優先的にインシデントが発生する傾向
- 自組織を本当の意味で知っているかが肝
 - 関連組織 (海外拠点、子会社、グループ会社) の状況
 - 本社側で管理しきれない運用体制やシステム
 - 利用できる予算が少ない
 - 本社とNWやシステムが共有されている
 - 運用構築ベンダ丸投げの体制
 - ネットワーク管理
 - アカウント管理
 - エンドポイント管理

2020年以降、これらの傾向は変化していない

まとめ

- A41APT の曖昧さ
 - 継続したマルウェア開発、関連 TTPs の増加
 - 背後に見える攻撃グループ像は複数存在
 - ただし、起点はインターネット境界
- A41APT であっても侵入型ランサムウェアであっても同じ考えが必要
 - 海外拠点や関連会社含めた、インターネット境界保護が重要
 - 構築・運用ベンダも考慮した体制づくり
 - C2 確立後のツールや AD 関連操作を検知する仕組み
 - EDR やログイン監査、自動起動エントリチェックによる積極的な脅威ハンティング
- 情報共有をしていく形
 - 各社それぞれの事例だけでは全容を捉えきれない
 - 被害組織につながらない、各社の事例を持ち寄る形



A41APT に関する参考 URL

1. A41APT case ~ Analysis of the Stealth APT Campaign Threatening
 - https://jsac.jpcert.or.jp/archive/2021/pdf/JSAC2021_202-niwa-yanagishita_jp.pdf
 - https://jsac.jpcert.or.jp/archive/2021/pdf/JSAC2021_202-niwa-yanagishita_en.pdf
2. APT10 : A41APTの活動内で発見された高度なマルチレイヤー型ローダー「Ecipekac」 | カスペルスキー公式ブログ
 - <https://blog.kaspersky.co.jp/apt10-sophisticated-multi-layered-loader-ecipekac-discovered-in-a41apt-campaign/30393/>
 - <https://securelist.com/apt10-sophisticated-multi-layered-loader-ecipekac-discovered-in-a41apt-campaign/101519/>
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 Tenshe」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>

その他参考 URL

1. Uncovering New Activity By APT10 | FortiGuard Labs
 - <https://www.fortinet.com/blog/threat-research/uncovering-new-activity-by-apt-10>
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>



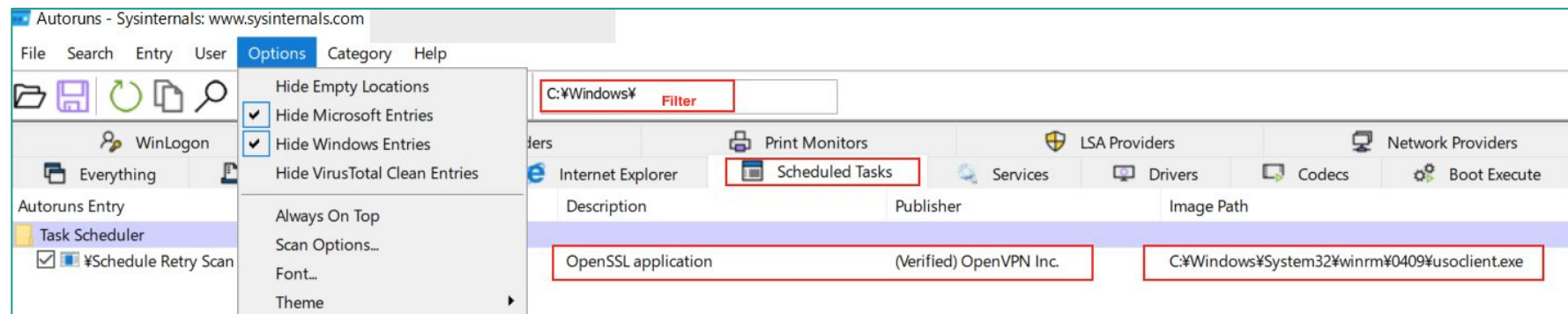
IoCs

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	SigLoader (Layer I)
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

参考: 不審な自動起動エントリの確認

EDR やフォレンジック調査がすぐに行えない場合

- Autoruns などを用いて自動起動エントリを調査することが有効
- A41APT キャンペーンでは、タスクスケジューラが使用されるため Scheduled Tasks から以下の条件にあてはまるタスクを調査
 - C:¥Windows¥ や C:¥Intel¥ 配下に存在するサードパーティ製の実行ファイル等



IOCs - 使用されたスケジュールタスク

プログラム	Description	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.

その他にVMware Tools、SandboxieなどProgram Files配下の3rd Party 製の実行ファイルがC:\Windows や C:\Intel\ に設置される可能性がある



ご清聴ありがとうございました

