

【緊急レポート】Microsoft社のデジタル署名ファイルが悪用する「SigLoader」による標的型攻撃を確認 | LAC WATCH

By 石川 芳浩

Published: 2020-12-01 · Archived: 2026-04-05 17:31:45 UTC

【更新情報】2021年1月13日

IOC (Indicator Of Compromised) を追記いたしました。

ラックの石川と松本です。

2020年7月ごろから、Microsoft社のデジタル署名（コードサイン証明書）されたDLLファイルが悪用するマルウェア「SigLoader」が使われた標的型攻撃を複数確認しています。攻撃者によって、正規のデジタル署名されたファイルを攻撃に悪用された場合、セキュリティ対策製品による検知をすり抜け、広範囲にわたる攻撃活動を容易に実行される恐れがあります。

- [デジタル署名DLLファイルが悪用するSigLoader](#)
- [デジタル署名されたファイルの改ざん手法](#)
- [SigLoaderを使用した攻撃手口の概要](#)
- [SigLoaderが読み込むファイル](#)
- [SigLoaderが利用する暗号化方法](#)
- [シェルコードによる2段目のSigloader \(2nd\) 作成](#)
- [SigLoader \(2nd\) が読み込むファイルと暗号化方法](#)
- [シェルコードによる最後のペイロード作成](#)
- [最終的に実行されるペイロード \(DelfsCake\)](#)
- [まとめ](#)
- [IOC \(Indicator Of Compromised\)](#)

デジタル署名DLLファイルが悪用するSigLoader

今回は、このMicrosoft社のデジタル署名されたファイルが悪用するSigLoaderの特徴を紹介します。なお、このSigLoaderを利用する攻撃者は、2020年12月1日に、APT10の可能性があると@Int2e_氏によってTwitter^{※1}で報告されています。

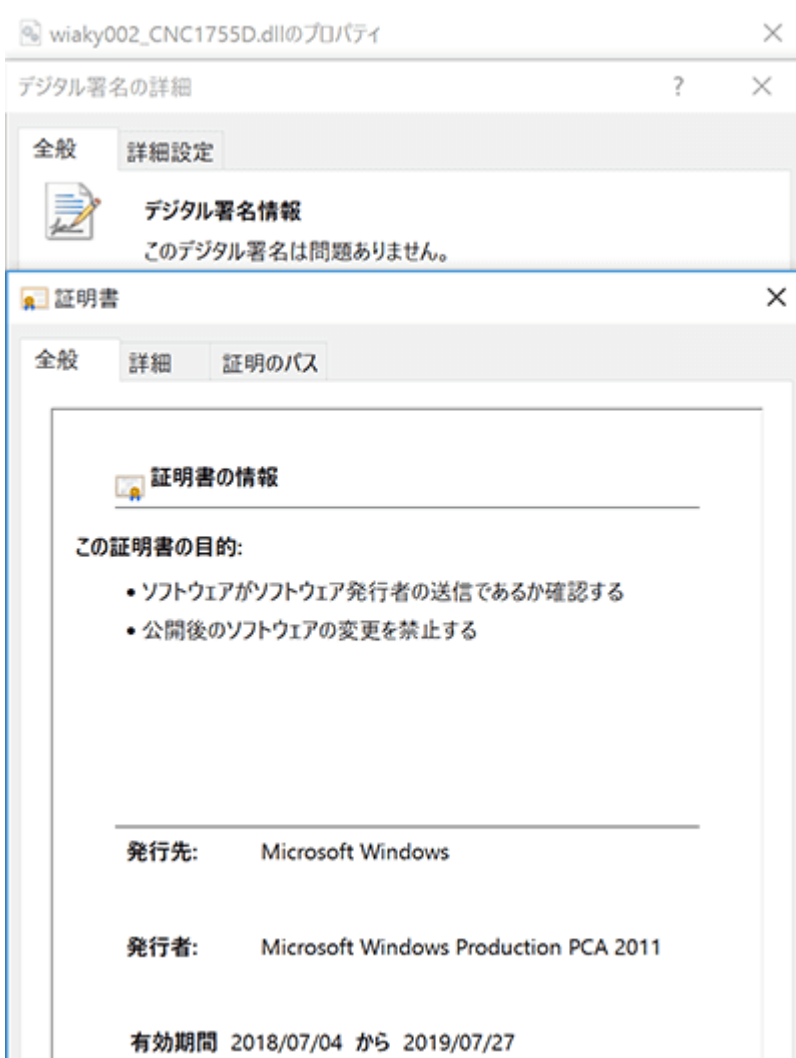
※1 https://twitter.com/Int2e_/status/1333501729359466502

図1は、SigLoaderで悪用されたDLLファイル (wiaky002_CNC1755D.dll) のデジタル署名をWindowsのファイルプロパティまたはSigcheck^{※2}で確認したものです。

※2 [Sigcheck - Windows Sysinternals | Microsoft Docs](#)

青線枠で示すように、署名の有効期間が2019年7月27日で失効しているため、エラーメッセージが出力されていますが、赤線枠のように「Signed」と検証されており、署名は正しいものであることがわかり

ます。



```
C:\>sigcheck64.exe -i wiaky002_CNC1755D.dll

Sigcheck v2.80 - File version and signature viewer
Copyright (C) 2004-2020 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\>wiaky002_CNC1755D.dll:
  Verified: Signed
  Signing date: 10:35 2018/09/15
  Signing date: 10:35 2018/09/15
  Catalog: C:\wiaky002_CNC1755D.dll
  Signers:
    Microsoft Windows
      Cert Status: This certificate or one of the certificates in the certificate chain is not time valid.
      Valid Usage: NT5 Crypto, Code Signing
      Cert Issuer: Microsoft Windows Production PCA 2011
      Serial Number: 33 00 00 01 C4 22 B2 F7 9B 79 3D AC B2 00 00 00 00 01 C4
      Thumbprint: AE9C1AE54763822EEC42474983D8B635116C8452
      Algorithm: sha256RSA
      Valid from: 5:45 2018/07/04
      Valid to: 5:45 2019/07/27
```

図1 Microsoft社のデジタル署名を持つDLLファイルの署名の有効性確認

また、図2に示すように、このDLLファイルは、Windows 10で利用される「pkeyhelper.dll」のファイルメタ情報やPDBファイル情報を持つことが確認できます。

Property	Value
CompanyName	Microsoft Corporation
FileDescription	Product Key Helper
FileVersion	10.0.17763.1 (WinBuild.160101.0800)
InternalName	Product Key Helper
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	pkeyhelper.dll
ProductName	Microsoft® Windows® Operating System

```

| 20 23 0B 00 00 00 00 00 | 00 00 00 00 52 53 44 53 | #.....RSDS |
| 3A F8 3D B8 1C A4 7B AC | A0 76 2E A2 BE 05 B5 A3 | :.=...{..v..... |
| 01 00 00 00 70 6B 65 79 | 68 65 6C 70 65 72 2E 70 | ....pkeyhelper.p |
| 64 62 00 00 47 43 54 4C | 00 10 00 00 70 01 00 00 | db..GCTL....p... |

```

図2 「wiaky002_CNC1755D.dll」のファイルメタ情報およびPDBファイル情報（一部抜粋）

デジタル署名されたファイルの改ざん手法

SigLoaderは、前述の通りMicrosoft社のデジタル署名されたDLLファイルを読み込み、攻撃に悪用します。一般的にデジタル署名が有効である場合は、署名した組織が実在することやデータが改ざんされていないことなどが証明されており、万一、デジタル署名のあるファイルを改ざんした場合、デジタル署名が無効であると検出されます。

しかし、SigLoaderを利用する攻撃者は、デジタル署名されたDLLファイル内に存在する証明書テーブル（Certificate Table）のサイズを拡張し、そのテーブル内のデータを変更することで、デジタル署名に影響を及ぼさずに、DLLファイルを改ざんしています。

Windowsでは、デジタル署名のハッシュ計算にあたって証明書テーブルを対象範囲外としているため、証明書テーブルを改ざんしたとしてもハッシュ値が一致し、デジタル署名が有効であると表示されます。この手法は、2009年にHugo氏のブログで報告^{※3}されており、その後、2016年にはBlack Hat USAでも発表^{※4}されています。

※3 [Changing a Signed Executable without Altering Windows Digital Signatures | Aymeric on Software](#)

※4 [Certificate Bypass: Hiding and Executing Malware from a Digitally Signed Executable | Deep Instinct](#)

図3は、Microsoft社のデジタル署名された正規の「pkeyhelper.dll」とSigLoaderが悪用するMicrosoft社のデジタル署名された「wiaky002_CNC1755D.dll (pkeyhelper.dll)」を比較したものです。SigLoaderが悪

用する改ざんされたDLLファイルには、正規のDLLファイルには存在しない、赤線枠の領域にデータが含まれていることが確認できます。

```

000BA320 | 24 06 03 55 04 03 13 1D | 4D 69 63 72 6F 73 6F 66 | $.U...Microsof
000BA330 | 74 20 54 69 6D 65 2D 53 | 74 61 6D 70 20 50 43 41 | t Time-Stamp PCA
000BA340 | 20 32 30 31 30 02 13 33 | 00 00 00 E4 DA 1F 36 DE | 2010..3.....6.
000BA350 | F7 70 2D D8 00 00 00 00 | 00 E4 30 16 04 14 55 25 | .p-.....0...U%
000BA360 | 53 21 CA 7D A3 2B 27 B7 | BA B4 E2 7B 24 17 35 9F | s!.)+'.....{$.5.
000BA370 | 95 89 30 0D 06 09 2A 86 | 48 86 F7 0D 01 01 0B 05 | ..0...*.H.....
000BA380 | 00 04 82 01 00 94 DA FA | BB 6A 71 6C 5B 9C 38 A9 | .....jqll[.8.
000BA390 | 89 27 74 EB BF 84 30 3E | 08 12 67 B2 C2 C4 93 90 | .'t...0>..g.....
000BA3A0 | EC 29 37 E5 C5 4B 15 D5 | 3F 19 95 72 7E 7F C7 94 | .)7..K..?.r~...
000BA3B0 | 4D 24 88 E7 55 DE 07 CB | 4A 45 E1 04 B1 41 B7 28 | M$.U...JE...A.(
000BA3C0 | EB EE 6D 7C B1 18 E0 84 | F1 E0 97 64 C7 D0 4E A4 | ..m|.....d..N.
000BA3D0 | A1 DE EB FD 52 12 D3 97 | 68 28 A0 60 BA 7C DF 37 | ....R...h(`.|.7
000BA3E0 | A3 B0 A0 FC F7 1F 9C 98 | 7C 0F B2 88 E4 EE 3D 7B | .....|.....={
000BA3F0 | 40 82 FB 64 20 AB F3 8A | D8 AB CFA1 5B 93 38 81 | @..d .....[.8.
000BA400 | F6 A1 26 3A BD 52 A8 E1 | 37 2A A7 BB 32 02 F2 BE | ..&::R..7*..2...
000BA410 | 6C CD DC AC 5A 72 9A 29 | 0B 10 D1 52 DF F1 52 A7 | l...Zr.)...R..R.
000BA420 | 4D F0 55 16 8B DF D5 40 | A5 DF 25 8D 9F DF DC D5 | M.U....@...%.....
000BA430 | 54 47 45 1B 52 AA 66 DB | 66 91 71 27 58 DD 50 39 | TGE.R.f.f.q'X.P9
000BA440 | 96 D3 F0 1C 1F A6 7D 06 | 91 D1 BB 85 53 5A EA 3D | .....}.....SZ.=
000BA450 | 79 15 84 57 9B 14 BD 99 | 0A 2D 88 CE F6 6A C1 B8 | y..W.....-...j..
000BA460 | 0E 30 0B C0 1E 38 BF 13 | 35 40 CF 82 62 65 8B BE | .0...8..5@..be..
000BA470 | 89 D2 65 4F 66 01 50 8A | 4E A6 87 77 59 46 E1 0E | ..eOf.P.N..wYF..
000BA480 | 92 FB DB 70 D1 00 00 00 | 7B FC 55 93 27 C3 11 4F | ...p....{.U.'..O

```

```

000BA320 | 24 06 03 55 04 03 13 1D | 4D 69 63 72 6F 73 6F 66 | $.U...Microsof
000BA330 | 74 20 54 69 6D 65 2D 53 | 74 61 6D 70 20 50 43 41 | t Time-Stamp PCA
000BA340 | 20 32 30 31 30 02 13 33 | 00 00 00 E4 DA 1F 36 DE | 2010..3.....6.
000BA350 | F7 70 2D D8 00 00 00 00 | 00 E4 30 16 04 14 55 25 | .p-.....0...U%
000BA360 | 53 21 CA 7D A3 2B 27 B7 | BA B4 E2 7B 24 17 35 9F | s!.)+'.....{$.5.
000BA370 | 95 89 30 0D 06 09 2A 86 | 48 86 F7 0D 01 01 0B 05 | ..0...*.H.....
000BA380 | 00 04 82 01 00 94 DA FA | BB 6A 71 6C 5B 9C 38 A9 | .....jqll[.8.
000BA390 | 89 27 74 EB BF 84 30 3E | 08 12 67 B2 C2 C4 93 90 | .'t...0>..g.....
000BA3A0 | EC 29 37 E5 C5 4B 15 D5 | 3F 19 95 72 7E 7F C7 94 | .)7..K..?.r~...
000BA3B0 | 4D 24 88 E7 55 DE 07 CB | 4A 45 E1 04 B1 41 B7 28 | M$.U...JE...A.(
000BA3C0 | EB EE 6D 7C B1 18 E0 84 | F1 E0 97 64 C7 D0 4E A4 | ..m|.....d..N.
000BA3D0 | A1 DE EB FD 52 12 D3 97 | 68 28 A0 60 BA 7C DF 37 | ....R...h(`.|.7
000BA3E0 | A3 B0 A0 FC F7 1F 9C 98 | 7C 0F B2 88 E4 EE 3D 7B | .....|.....={
000BA3F0 | 40 82 FB 64 20 AB F3 8A | D8 AB CFA1 5B 93 38 81 | @..d .....[.8.
000BA400 | F6 A1 26 3A BD 52 A8 E1 | 37 2A A7 BB 32 02 F2 BE | ..&::R..7*..2...
000BA410 | 6C CD DC AC 5A 72 9A 29 | 0B 10 D1 52 DF F1 52 A7 | l...Zr.)...R..R.
000BA420 | 4D F0 55 16 8B DF D5 40 | A5 DF 25 8D 9F DF DC D5 | M.U....@...%.....
000BA430 | 54 47 45 1B 52 AA 66 DB | 66 91 71 27 58 DD 50 39 | TGE.R.f.f.q'X.P9
000BA440 | 96 D3 F0 1C 1F A6 7D 06 | 91 D1 BB 85 53 5A EA 3D | .....}.....SZ.=
000BA450 | 79 15 84 57 9B 14 BD 99 | 0A 2D 88 CE F6 6A C1 B8 | y..W.....-...j..
000BA460 | 0E 30 0B C0 1E 38 BF 13 | 35 40 CF 82 62 65 8B BE | .0...8..5@..be..
000BA470 | 89 D2 65 4F 66 01 50 8A | 4E A6 87 77 59 46 E1 0E | ..eOf.P.N..wYF..
000BA480 | 92 FB DB 70 D1 00 00 00 | 7B FC 55 93 27 C3 11 4F | ...p....{.U.'..O
000BA490 | B8 0B 8D D7 B5 88 62 6C | 5C F0 10 95 D9 03 76 44 | .....bl\.....vD

```

図3 「pkeyhelper.dll」の比較（上：正規DLLファイル／下：改ざんされたDLLファイル）

SigLoaderを使用した攻撃手口の概要

SigLoaderは、DLL Side-Loadingを悪用して実行されます。正規の実行ファイルによって読み込まれたSigLoaderは、最初に実行ファイルと同じディレクトリ内にあるMicrosoft社のデジタル署名がされたDLLファイル、または暗号化されたペイロード（DATファイル）を読み込みデータを復号します。その後、復号したペイロードをメモリ領域に展開し実行します。最終的に実行されるペイロードは、2020年11月20日時点で3種類が確認できています。

図4は、SigLoaderの挙動を示したものであり、SigLoaderを2段階で使用するケース（青矢印）と1段階のみのケース（赤矢印）の2パターン存在することを確認しています。

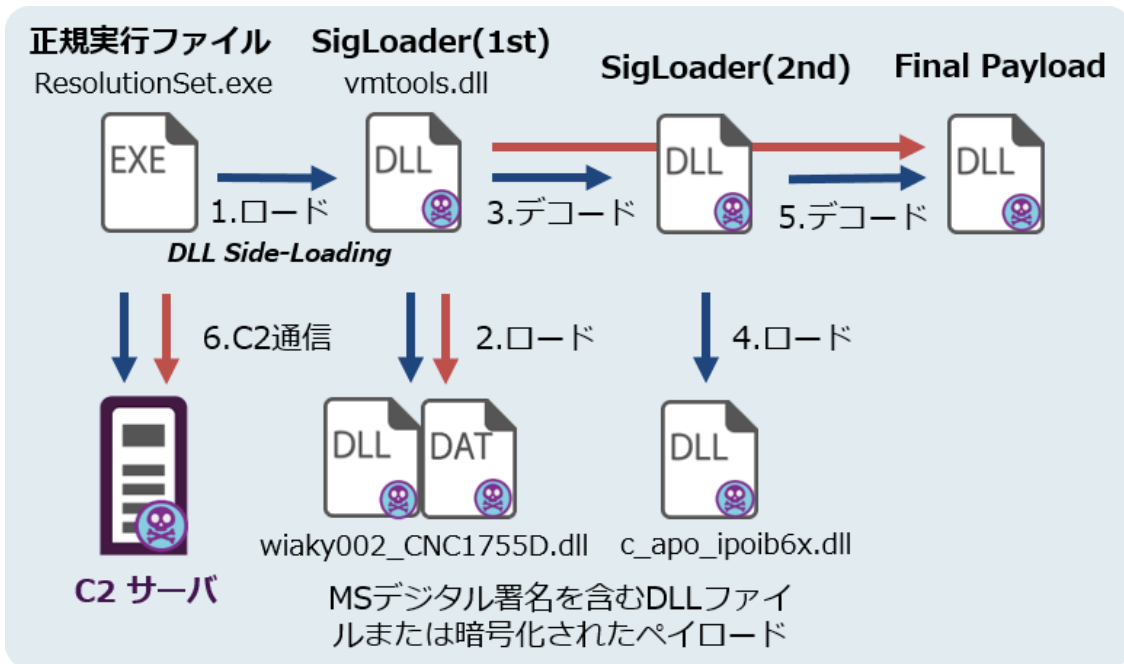


図4 SigLoaderの動作概要図（一例）

以降では、VMware製品の正規実行ファイル（ResolutionSet.exe）を悪用して、実行されるSigLoader（vmtools.dll）の攻撃手口を詳しくみていきます。

SigLoaderが読み込むファイル

SigLoaderには図5に示すように、Sigという文字列を起点に読み込むファイルや暗号化方法などがファイル内にハードコード^{※5}されており、このハードコードされた内容を利用してペイロードが展開されていきます。

※5 別のSigLoaderでは、Sigという文字列や暗号化方法などが未記載のものも確認しています。このようなSigLoaderでは、暗号化方法が記載された箇所に暗号化キーなどが含まれています。

```

0x180027868 5369 6700 2e00 0000 635f 6170 6f5f 6970 Sig.....c_apo_ip
0x180027878 6f69 6236 782e 646c 6c00 0000 4445 5300 oib6x.dll...DES.
0x180027888 4145 5300 0000 0000 7769 616b 7930 3032 AES....wiaky002
0x180027898 5f43 4e43 3137 3535 442e 646c 6c00 0000 _CNC1755D.dll...
0x1800278a8 4e6f 6e65 0000 0000 5253 4100 6368 6172 None...RSA.char
0x1800278b8 0000 0000 7368 6f72 7400 0000 696e 7400 ....short...int.
0x1800278c8 584f 5200 0000 0000 2205 9319 0b00 0000 XOR.....".....
    
```

図5 SigLoaderにハードコードされたファイル名や暗号化方法（一例）

まずSigLoaderはMicrosoft社のデジタル署名されたDLLファイル「wiaky002_CNC1755D.dll」を読み込み、ファイル内のオーバーレイ領域に含まれるデータを取得します。今回のケースでは、先ほど紹介した図3の赤線枠に示すオフセット0xBA488からデータ末尾（0xF4090）までを読み込み、そのコードをメモリ領域上に展開します。このオフセット値0xBA488は、データ末尾から図6に示すように、0x39C08（0x39C05から値を1増やし8の倍数で割り切れる値）バイト遡った値から算出されています。

```
● 15 | v15[12] = a1;  
● 16 | v5 = 0x39C05 * a3;  
● 17 | if ( 0x39C05 * a3 % 8 )  
    18 | {  
    19 |     do  
● 20 |         ++v5;  
● 21 |     while ( v5 % 8 );  
    22 | }  
    23 | else  
    24 | {  
● 25 |     v5 += 8;  
    26 | }
```

図6 読み込むデータのオフセットの計算式 (一部抜粋)

SigLoaderが利用する暗号化方法

読み込まれたオーバーレイ領域に含まれるデータは、XOR+カスタムDES+AESによって暗号化されています。暗号化キーは図7に示すように、SigLoader自体にハードコードされており、AESキーと初期化ベクトル (赤線枠)、カスタムDESで利用するキー (橙線枠) です。また、このSigLoaderには、2段目で利用されるSigLoader (2nd) のAESキーや初期化ベクトル (青線枠)、カスタムDESで利用するキー (緑線枠) もハードコードされています。

```

0x180030b40 e828 0100 059c 0300 5055 3838 4979 704a .(.....PU88IypJ
0x180030b50 7675 4269 6832 3455 7057 3247 6730 3148 vuBih24UpW2Gg01H
0x180030b60 3437 3830 4e77 6334 0000 0000 0000 0000 4780Nwc4.....
0x180030b70 3669 416e 3871 7a66 4637 4d30 5177 444b 6iAn8qzffF7M0QwDK
0x180030b80 4d33 6a44 4e6c 5250 435a 3930 3556 4433 M3jDNlRPCZ905VD3
0x180030b90 0000 0000 0000 0000 0000 0000 0000 0000 ..... AES Key .....
0x180030ba0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030bb0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030bc0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030bd0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030be0 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030bf0 3669 416e 3871 7a66 4637 4d30 5177 444b 6iAn8qzffF7M0QwDK
0x180030c00 0000 0000 0000 0000 0000 0000 0000 0000 ..... AES IV .....
0x180030c10 3631 4136 564f 4b77 4862 3163 7535 4a63 61A6V0KwHb1cu5JC
0x180030c20 374e 764c 676d 4339 3163 5243 7a32 496c 7NvLgmC91cRCz2I
0x180030c30 0000 0000 0000 0000 0000 0000 0000 0000 ..... AES Key .....
0x180030c40 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030c50 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030c60 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030c70 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030c80 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x180030c90 3631 4136 564f 4b77 4862 3163 7535 4a63 61A6V0KwHb1cu5JC
0x180030ca0 0000 0000 0000 0000 0000 0000 0000 0000 ..... AES IV .....
0x180030cb0 435a 3659 3330 3077 3962 4463 7544 3363 CZ6Y300w9bDcuD3C
0x180030cc0 5a39 7635 676d 3054 5463 3730 7a4f 366c Z9v5qm0TTc70z06L
    
```

図7 SigLoaderにハードコードされている暗号化キー（一部抜粋）

SigLoaderはこれらの暗号化キーを使用して、オーバーレイ領域に読み込まれた暗号化データを図8の順で復号^{※6}し、シェルコードをメモリ領域に展開します。復号は、AESでは鍵長256ビット、ブロックチェーンングとしてCBCモードを使用し、カスタムDESでは、鍵長64ビット、ECBモードを使用します。復号後は、Call命令でシェルコードが展開されたメモリ領域を呼び出します。

※6 別のSigLoaderでは暗号化アルゴリズムの適用順序や適用回数が異なる可能性があります。



図8 SigLoaderによるデータ復号（一例）

SigLoaderには、XOR、カスタムDES、AESの3種類の暗号化アルゴリズムが実装されています。これらのうち、SigLoaderで実装されているカスタムDESは、使用される定数に異常はないものの、サブ鍵生成アルゴリズムやラウンド関数などが標準のものと異っており、標準の暗号化ライブラリでは暗号化されたデータを復号することができません。

また、これらのデータをエンコードするための処理が多数みられ、サブ鍵生成アルゴリズムやラウンド関数などは、攻撃者が自ら実装している可能性があります。また、コード内にRSAの実装を見据えた箇所があることから、今後RSAの処理に関しても実装される可能性があります。

シェルコードによる2段目のSigloader (2nd) 作成

復号されたシェルコードは、SigLoader (2nd) を作成するために用いられます。図9は、シェルコードを一部デコンパイルしたものです。特徴的な文字列cepepac (Cakepice) をチェックするコードが含まれていることが確認できます。

```

● 75 v0 = "ecipekac";
● 76 v1 = 0i64;
● 77 v2 = 0i64;
● 78 while ( aEcipekac[v2] != 'e'
79         || aEcipekac[v2 + 1] != 'c'
80         || aEcipekac[v2 + 2] != 'i'
81         || aEcipekac[v2 + 3] != 'p'
82         || aEcipekac[v2 + 4] != 'e'
83         || aEcipekac[v2 + 5] != 'k'
84         || aEcipekac[v2 + 6] != 'a'
85         || aEcipekac[v2 + 7] != 'c' )
86     {
● 87     v1 = (unsigned int)(v1 + 1);
● 88     if ( ++v2 >= 10000 )

```

図9 シェルコード内の「ecipekac」を確認するコード（一部抜粋）

このシェルコードには、自身のコード内に以下の3つのコードが分割されて含まれており、これらを組み合わせて、次のDLLファイルを読み込む2段目のSigloader (2nd) を作成します。分割されたデータの組み合わせ方法は、シェルコードに含まれる文字列ecipekacを起点に行われます。

また、この文字列以降の16バイトには、PEファイルを作成する際のメモリ領域を確保する際のサイズ0x39000（図10：橙線枠）およびSection Headersおよびプログラムコードのサイズ0x038E18（図10：紫線枠）が指定されています。

- Section Headersおよびプログラムコード（図10：緑線枠）
- NT Header（PE Header）（図11：青線枠）
- MS-DOS Header/Stub（図11：赤線枠）

```

0x00000bf5 6563 6970 656b 6163 0090 0300 188e 0300 ecipekac .....
0x00000c05 2e74 6578 7400 0000 2a84 0200 0010 0000 .text...*.....
0x00000c15 0086 0200 0004 0000 0000 0000 0000 0000 .....
0x00000c25 0000 0000 2000 0060 2e72 6461 7461 0000 .....rdata.
0x00000c35 b4b2 0000 00a0 0200 00b4 0000 008a 0200 .....
0x00000c45 0000 0000 0000 0000 0000 0000 4000 0040 .....@..@
0x00000c55 2e64 6174 6100 0000 245d 0000 0060 0300 .data...$]...
0x00000c65 0026 0000 003e 0300 0000 0000 0000 0000 &...>.....
0x00000c75 0000 0000 4000 00c0 2e70 6461 7461 0000 .....pdata.
0x00000c85 a81e 0000 00c0 0300 0020 0000 0064 0300 .....d..
0x00000c95 0000 0000 0000 0000 0000 0000 4000 0040 .....@..@
0x00000ca5 2e72 7372 6300 0000 b401 0000 00e0 0300 .rsrc.....
0x00000cb5 0002 0000 0084 0300 0000 0000 0000 0000 .....
0x00000cc5 0000 0000 4000 0040 2e72 656c 6f63 0000 .....@..@.reloc.
0x00000cd5 7208 0000 00f0 0300 000a 0000 0086 0300 r.....
0x00000ce5 0000 0000 0000 0000 0000 0000 4000 0042 .....@..B
0x00000cf5 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x00000d05 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x00000d15 0000 0000 0000 0000 0000 0000 0000 0000 .....
    
```

図10 Section Headersおよびプログラムコード（一部抜粋）

```

0x00039a15 0000 0000 0000 0000 5045 0000 6486 0600 .....PE.d
0x00039a25 c42d 775f 0000 0000 0000 0000 f000 2200 .-w_....."
0x00039a35 0b02 0a00 0086 0200 0006 0100 0000 0000 .....@..
0x00039a45 a0f7 0000 0010 0000 0000 0040 0100 0000 .....
0x00039a55 0010 0000 0002 0000 0500 0200 0000 0000 .....
0x00039a65 0500 0200 0000 0000 0000 0400 0004 0000 .....
0x00039a75 367b 0400 0300 4081 0000 1000 0000 0000 6{...@.....
0x00039a85 0010 0000 0000 0000 0000 1000 0000 0000 .....
0x00039a95 0010 0000 0000 0000 0000 1000 0000 0000 .....
0x00039aa5 0000 0000 0000 0000 4849 0300 3c00 0000 .....HI.<
0x00039ab5 00e0 0300 b401 0000 00c0 0300 a81e 0000 .....
0x00039ac5 0000 0000 0000 0000 00f0 0300 c405 0000 .....
0x00039ad5 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x00039ae5 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x00039af5 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x00039b05 00a0 0200 d002 0000 0000 0000 0000 0000 .....
0x00039b15 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x00039b25 4d5a 9000 0300 0000 0400 0000 ffff 0000 MZ.....
0x00039b35 b800 0000 0000 0000 4000 0000 0000 0000 .....@..
0x00039b45 0000 0000 0000 0000 0000 0000 0000 0000 .....
0x00039b55 0000 0000 0000 0000 0000 0000 e000 0000 .....
0x00039b65 0e1f ba0e 00b4 09cd 21b8 014c cd21 5468 .....!..L.!Th
0x00039b75 6973 2070 726f 6772 616d 2063 616e 6e6f is program canno
0x00039b85 7420 6265 2072 756e 2069 6e20 444f 5320 t be run in DOS
0x00039b95 6d6f 6465 2e0d 0d0a 2400 0000 0000 0000 mode...$.
0x00039ba5 33ec 6198 778d 0fcb 778d 0fcb 778d 0fcb 3.a.w...w...w...
0x00039bb5 18fb 91cb 7b8d 0fcb 18fb a4cb 4e8d 0fcb ....{.....N...
0x00039bc5 18fb a5cb e38d 0fcb 7ef5 9ccb 728d 0fcb .....~...r...
0x00039bd5 778d 0ecb 298d 0fcb 18fb a0cb 758d 0fcb w...).~...u...
0x00039be5 18fb 92cb 768d 0fcb 5269 6368 778d 0fcb ...v...Richw...
    
```

図11 MS-DOS Header/StubおよびPE Header

これらのデータの組み換えに際して、最初に図11に示すように赤線枠のMZヘッダが含まれるMS-DOS Header/Stub (0xE0バイト) を事前に確保したメモリ領域にコピーし、次に青線枠のPE Header (0x108バイト)、最後にtextセクションなどが含まれるSection Headersおよびプログラムコード (0x038E18) をコピーし、PEファイル (SigLoader) を作成します。その後は、Call命令で2段目のSigLoaderが展開されたメモリ領域を呼び出します。

SigLoader (2nd) が読み込むファイルと暗号化方法

2段目のSigLoader (2nd) は、前述したSigLoader (1st) とほぼ同様な作りです。SigLoader (2nd) は、図12に示すように、正規実行ファイルと同じディレクトリにあるMicrosoft社のデジタル署名されたDLLファイル「c_apo_ipoib6x.dll」のオーバーレイ領域に含まれるデータを読み込み、データを復号します。その後、復号したシェルコードをメモリ領域上に展開し、Call命令で呼び出します。

このDLLファイルは、図13に示すように、Windows 10で利用される「wintrust.dll」を改ざんしたものです。展開されたシェルコードは、SigLoader (1st) によって作成された文字列Cakepiceを含むシェルコードとは異なります。

なお、データの暗号化方法については、SigLoader (1st) と同様のものが利用されており、データの復号順序は、SigLoader (1st) とは逆順のAES + XOR(Key=0x0) + カスタムDES + XOR(Key=0xBC)です。また、AESおよびカスタムDESの暗号化キーについては、図7に示した青線枠と緑線枠のものが利用されています。

```
0x14002f86c 5369 6700 2e00 0000 0000 0000 635f 6170 Sig.....c_ap
0x14002f87c 6f5f 6970 6f69 6236 782e 646c 6c00 0000 o_ipoib6x.dll...
0x14002f88c 4445 5300 4145 5300 5c00 0000 2f00 0000 DES.AES.\.../...
0x14002f89c 5253 4100 4c53 4200 6368 6172 0000 0000 RSA.LSB.char....
0x14002f8ac 7368 6f72 7400 0000 696e 7400 584f 5200 short...int.XOR.
```

図12 SigLoader (2nd) にハードコードされたファイル名や暗号化方法 (一例)

Property	Value
CompanyName	Microsoft Corporation
FileDescription	Microsoft Trust Verification APIs
FileVersion	10.0.17763.1 (WinBuild.160101.0800)
InternalName	WINTRUST.DLL
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	WINTRUST.DLL
ProductName	Microsoft® Windows® Operating System
ProductVersion	10.0.17763.1

```
| 00 00 00 00 52 53 44 53 | F9 3F AA BE 49 90 C9 FF | .....RSDS.?..I...
| 7D 03 E3 D0 70 2C 7C 02 | 01 00 00 00 77 69 6E 74 | }...p,|.....wint
| 72 75 73 74 2E 70 64 62 | 00 00 00 00 00 55 47 50 | rust.pdb.....UGP
```

図13 「wintrust.dll」のファイルメタ情報およびPDBファイル情報 (一部抜粋)

シェルコードによる最後のペイロード作成

SigLoader (2nd) によって復号されたシェルコードは、最後のペイロードを作成するために用いられます。最後のペイロード (図14: 青線枠) は、シェルコード内にRC4で暗号化されて含まれており、図14

の赤線枠で示すRC4キーで復号することで最終的なペイロードをメモリ領域に展開します。復号後は、Call命令でペイロードが展開されたメモリ領域を呼び出します。

```

0x00001249 79cf 4f26 6a57 775b b7ab d429 4e51 ca2f y.O&jWw[...]NQ./
0x00001259 3c04 83b5 5b7d f12c 5914 875d d987 5959 <...[},Y...YY
0x00001269 bef0 0e9c 7f50 2404 2f2a be54 fcce a927 .....P$./*..T...!
0x00001279 fd30 45ad d3eb 2500 a98a 3731 cd5d e069 .0E...%...71.]i
0x00001289 ca84 1d6c 9fa2 d71c 81f7 acfd aadf 5aa0 ...l.....Z.
0x00001299 a4de a761 252d f483 6294 5836 3104 8a09 ...a%-..b.X61...
0x000012a9 e526 1a0b 3d10 7102 042a 7e1c a279 60ec .&..=.q..*~..y'.
0x000012b9 0d1d 9839 93a8 a39a e5f0 3e04 78e2 06e4 ...9.....>.x...
0x000012c9 3598 0a97 b357 c78a aa0e 2ac7 949f abcb 5....W....*.....
0x000012d9 565c 61b9 0a80 f76e 5465 7f08 3e4b b4fe V\a....nTe..>K..
0x000012e9 06da 6e6f 81cd 684c 5824 285e 1d2a fd33 ..no..hLX$(^.*.3
0x000012f9 4c3d 5a17 8ad0 4c85 d202 3471 81ef 5aea L=Z...L...4q...Z.
0x00001309 d974 c139 e7bd 6147 26b0 e449 54c9 c8cf .t.9..aG&..IT...
0x00001319 15f9 9e2d b7ec ccfb f0d9 d352 2444 14bf ....-.....R$D..
0x00001329 dd8f a1be 6a97 8569 b6f2 9700 614c b4cc ....j..i....aL..
0x00001339 6ebb 4b7d 4d16 97e3 6330 639c 3aec 5e39 n.K}M...c0c.:^9
0x00001349 ea68 1117 a68f 31eb 04e2 445a 27bf afe9 .h....1...DZ'...
    
```

図14 RC4キーおよび暗号化されたペイロード（一例）

最終的に実行されるペイロード（DelfsCake）

最終的に実行されるペイロード（DelfsCake）は、DLL形式で、C2サーバからデータやペイロードなどを受信して実行する機能を有します。

DelfsCakeには、図15に示すようにRSA公開鍵（1024bit）がハードコードされており、この鍵を使用して送信するデータを暗号化します。初期通信時に送信する内容は、図16の通りです。ユーザ名やコンピュータ名、プロセスID、OSバージョン、ビルド番号、ソケット名、実行日時、ランダムに生成した文字列などがC2サーバに送られます。

```

0x18001241e 9be4 76ae a82f 8659 410b 9219 b16f a1e6 ..v.../..YA....o..
0x18001242e 881a 0af1 e302 bdac 2b30 8192 9eb1 4ad6 .....+0....J.
0x18001243e d379 4d49 474a 416f 4742 414c 6b58 6345 .yMIGJAoGBALkXcE
0x18001244e 5443 4e62 4b52 554d 6c7a 3042 6b6c 384d TCNbKRUMlZ0Bkl8M
0x18001245e 722f 4a6d 3141 3456 4b78 644c 426c 4458 r/Jm1A4VKxdLBldX
0x18001246e 4374 442f 3966 4372 6653 446c 327a 2f4a CtD/9fCrfSDlZz/J
0x18001247e 6879 6b46 4a69 6b37 3837 7054 3035 5175 hykFJik787pT05Qu
0x18001248e 4b49 734c 575a 4c76 322f 6c71 4d6c 4478 KIsLWZLv2/lqMLDx
0x18001249e 6e4b 4550 4551 5244 4264 6d39 3030 4966 nKEPEQRDBdm900If
0x1800124ae 3237 7853 6863 4b2f 7152 6f53 4f4f 3865 27xShcK/qRoS008e
0x1800124be 6455 4434 3450 7068 4635 634d 664b 3136 dUD44PphF5cmfK16
0x1800124ce 564d 6f4e 3965 3344 4556 6550 347a 6475 VMon9e3DEVeP4zdu
0x1800124de 4361 6e50 3476 6246 7048 3376 7761 5449 CanP4vbFpH3vwaTI
0x1800124ee 314f 7231 5152 4167 4d42 4141 453d 00cf 10r1QRAGMBAAE=..
0x1800124fe 4f26 6a57 775b b7ab d429 0000 0000 0000 O&jWw[...].....
0x18001250e 0000 0000 0000 0000 0000 0000 0000 0000 .....
    
```

図15 DelfsCakeにハードコードされたRSA公開鍵（一例）

```

000000000026EE20 03 08 54 00 65 00 73 00 74 00 07 12 50 00 43 00 ..T.e.s.t...P.C.
000000000026EE30 2D 00 54 00 45 00 53 00 54 00 30 00 31 00 04 E8 -.T.E.S.T.O.1..è
000000000026EE40 08 00 00 02 40 09 06 01 B1 1D 01 00 30 00 05 13 .....@...±...0...
000000000026EE50 32 30 32 30 2F 31 31 2F 32 30 20 31 30 3A 32 32 2020/11/20 10:22
000000000026EE60 3A 32 31 06 0A 57 46 76 35 4C 30 63 6C 37 69 08 :21..WFv5L0c17i.
000000000026EE70 C0 A8 C8 0A 00 00 00 00 00 00 00 00 00 00 00 00 A`E.....
    
```

図16 送信されるデータの例（暗号化前）

DelfsCakeには、表1に示すコマンドが実装されています。なお、コマンド「d」に関しては、コマンドの受信時のデータを使用してライブラリのロードと関数アドレスの取得を行い、その関数をCall命令で呼び出します。Call命令の際に引数として受信したデータを指定することから、単純なコマンド実行を行う命令ではないと考えます。

表1 DelfsCakeのコマンド一覧

また、このDelfsCakeを実際に動作させてみましたが、2020年10月30日に確認した時点では、図17に示すように、C2サーバに接続後、RSTパケットが戻ってきており、C2サーバからデータは取得できていないため、最終的にどのようなマルウェアが実行されるか不明です。

Time	Source	Destination	Protol	Len	Info
2020-10-30...	192.168.108.131	88.198.101.58	TCP	66	51040 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=14
2020-10-30...	192.168.108.131	88.198.101.58	TCP	66	[TCP Retransmission] 51040 → 443 [SYN] Seq=0
2020-10-30...	192.168.108.131	88.198.101.58	TCP	62	[TCP Retransmission] 51040 → 443 [SYN] Seq=0
2020-10-30...	88.198.101.58	192.168.108.131	TCP	60	443 → 51001 [RST, ACK] Seq=1 Ack=1 Win=64240

図17 DLLファイルのC2サーバとの通信例

SigLoaderによって展開される最終的なペイロードは、DelfsCakeの他にも2種類確認しています。1つは、ペネトレーションツールであるMetasploit Framework^{※7}やCobalt Strike^{※8}で作成されたシェルコードです（図18）。シェルコードは、C2サーバと通信を確立した後、Cobalt Strike Beaconなど次のステージのマルウェアをダウンロードし、実行します。このシェルコードには、特徴的な文字列（baidu.com）が複数ハードコードされていることが確認できます。

※7 [Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit](#)

※8 [Adversary Simulation and Red Team Operations Software - Cobalt Strike](#)

```

0x00000000 fce8 8900 0000 6089 e531 d264 8b52 308b .....`..1.d.R0.
0x00000010 520c 8b52 148b 7228 0fb7 4a26 31ff 31c0 R..R..r(..J&1.1.
0x00000020 ac3c 617c 022c 20c1 cf0d 01c7 e2f0 5257 .<a|., .....RW
0x00000030 8b52 108b 423c 01d0 8b40 7885 c074 4a01 .R..B<...@x..tJ.
0x00000040 d050 8b48 188b 5820 01d3 e33c 498b 348b .P.H..X ...<I.4.
0x00000050 01d6 31ff 31c0 acc1 cf0d 01c7 38e0 75f4 ..1.1.....8.u.
0x00000060 037d f83b 7d24 75e2 588b 5824 01d3 668b .}.;}$u.X.X$.f.
0x00000070 0c4b 8b58 1c01 d38b 048b 01d0 8944 2424 .K.X.....D$$
0x00000080 5b5b 6159 5a51 ffe0 585f 5a8b 12eb 865d [[aYZQ..X_Z....]
0x00000090 686e 6574 0068 7769 6e69 5468 4c77 2607 hnet.hwiniThLw&.
0x000000a0 ffd5 e800 0000 0031 ff57 5757 5757 683a .....1.WWWWWh:
0x000000b0 5679 a7ff d5e9 a400 0000 5b31 c951 516a Vy.....[1.QQj
0x000000c0 0351 5168 bb01 0000 5350 6857 899f c6ff .QqH...SPhW...
0x000000d0 d550 e98c 0000 005b 31d2 5268 0032 a084 .P.....[1.Rh.2..
0x000000e0 5252 5253 5250 68eb 552e 3bff d589 c683 RRRSRPh.U.;.....
0x000000f0 c350 6880 3300 0089 e06a 0450 6a1f 5668 .Ph.3....j.Pj.Vh
0x00000100 7546 9e86 ffd5 5f31 ff57 576a ff53 5668 uF...._1.WWj.SVh
0x00000110 2d06 187b ffd5 85c0 0f84 ca01 0000 31ff -..{.....1.
0x00000120 85f6 7404 89f9 eb09 68aa c5e2 5dff d589 ..t.....h...]...
0x00000130 c168 4521 5e31 ffd5 31ff 576a 0751 5650 .hE!^1..1.Wj.QVP
0x00000140 68b7 57e0 0bff d5bf 002f 0000 39c7 7507 h.W...../.9.u.
0x00000150 5850 e97b ffff ff31 ffe9 9101 0000 e9c9 XP.{...1.....
0x00000160 0100 00e8 6fff ffff 2f71 3555 6100 6261 .....o.../q5Ua.ba
0x00000170 6964 752e 636f 6d00 6261 6964 752e 636f idu.com.baidu.co
0x00000180 6d00 6261 6964 752e 636f 6d00 6261 6964 m.baidu.com.baid
0x00000190 752e 636f 6d00 6261 6964 752e 636f 6d00 u.com.baidu.com.
    
```

図18 ペネトレーションツールを利用して作成されたシェルコード（一部抜粋）

もう一つも、次のステージのマルウェアをダウンロードし、実行することが主な機能と考えられるマルウェア（GreetCake）です。図19に示すようにRC4キーがハードコードされており、この鍵を使用して送信するデータを暗号化します。また、表2のような、いくつかのコマンド機能が実装されています。

GreetCakeは、C2サーバから特定の命令コマンドを受信することによって、C2サーバからデータをダウンロードし、復号後、PEファイルを実行します。図20に示すように、実行する前にPEファイル内に「hello」という文字列が含まれているか確認するというコードが特徴的です。

表2 GreetCakeのコマンド一覧

命令コマンド	説明
0x12C	C2通信の終了
0x12D	スレッドを作成し、PEファイルの実行
0x12F	データ送信（詳細不明）
0x130	PEファイルの実行
0x131 - 0x134	C2通信制御の設定（スリープ時間、タイムアウトなど）

```

0x140029920  0000 0000 0000 0000 0000 0000 0000 0000  .....
0x140029930  0000 0000 bb01 0000 4c32 3575 7566 7234  .....L25uufr4
0x140029940  4852 6a37 6858 4c00 0000 0000 0000 0000  HRj7hXL.....
0x140029950  0000 0000 0000 0000 0000 0000 0000 0000  .....
    
```

図19 GreetCakeにハードコードされたRC4キー（一例）

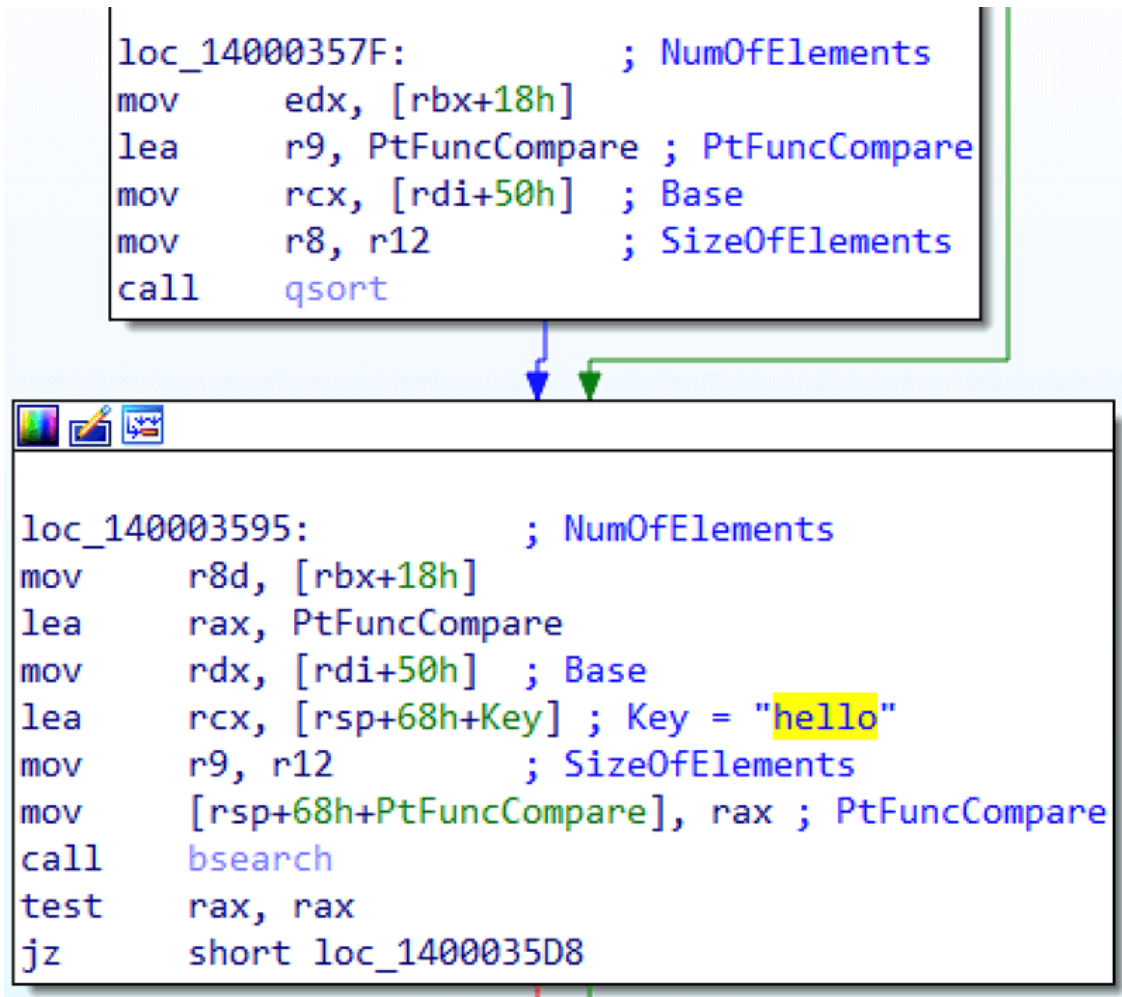


図20 文字列「hello」を検索するコード（一部抜粋）

これら2種類のペイロードについても、DelfsCakeと同様にC2サーバからダウンロードされるデータは取得できていないため、最終的にどのようなマルウェアが実行されるかは不明です。

まとめ

今回は、Microsoft社のデジタル署名されたDLLファイルを悪用するSigLoaderについて紹介しました。

SigLoaderは、デジタル署名されたDLLファイルの悪用、ペイロードの多段利用や攻撃者が独自に開発した暗号化方法の利用と高度な技術が組み込まれたローダです。暗号化方法では、RSAのコードが未実装であり、開発途中の可能性が窺えるため、今後も、さらに機能が向上したSigLoaderによる攻撃が続く可能性があります。

加えて、今回のデジタル署名されたファイルの改ざん手法を悪用した攻撃は、実際のサイバー攻撃の事例としてあまり報告されておらず、既存のセキュリティー対策製品で検出できない可能性があり、注意が必要です。

私たちがSigLoaderの攻撃を確認した事例の1つでは、攻撃者は標的組織への侵入経路として、SSL-VPN製品の脆弱性を悪用後、端末を侵害しSigLoaderを設置していました。SSL-VPN製品を悪用する攻撃は、標的型攻撃に限らず、金銭目的である攻撃者なども悪用しており、これらの製品の脆弱性を悪用

されないためにも、日々の脆弱性情報の管理と修正パッチの適用や緩和策の適用などの早急な対応が求められます。

また、2020年10月ごろからZeroLogonの脆弱性^{※9}が標的型攻撃で悪用されていることが、CISA^{※10}やSymantec社^{※11}から報告されていますので、この脆弱性についても十分に注意する必要があります。

※9 [CVE-2020-1472 - セキュリティ更新プログラム ガイド - Microsoft - Netlogon の特権の昇格の脆弱性](#)

※10 [APT Actors Chaining Vulnerabilities Against SLTT, Critical Infrastructure, and Elections Organizations | CISA](#)

※11 [Japan-Linked Organizations Targeted in Long-Running and Sophisticated Attack Campaign | Symantec Blogs](#)

ラックの脅威分析チームでは、今後もこのSigLoaderを利用する攻撃者について、継続的に調査し、広く情報を提供していきますので、ご活用いただければ幸いです。

IOC (Indicator Of Compromised)

2021年1月13日 更新

SigLoaderハッシュ値 (MD5)

bb45da230ee45e25df27a518dac0560a
cca46fc64425364774e5d5db782ddf54
42b6ab4bfa271019990960276e5c8176
6c68753503b79998ee29f52ce0f08e72
2e917dfcd0cf13e8c2a1bf4298d30794

ペイロード (MD5)

f5061a2b19a6cbe7a93c5c2b8d6fb20e
4638220ec2c6bc1406b5725c2d35edc3
03413d12861a9cd61352e364139ff212
d37964a9f7f56aad9433676a6df9bd19

DelfsCake (MD5)

fd722cfafc12774cefbfb2edd3d2a5eff

GreetCake (MD5)

78e1f309154fca4341a251bb9cb1e790
da2bd4a55a6291bdb3e8a81a60091611

通信先

85.204.74[.]113

88.198.101[.]58

81.7.7[.]159

Source: https://www.lac.co.jp/lacwatch/report/20201201_002363.html