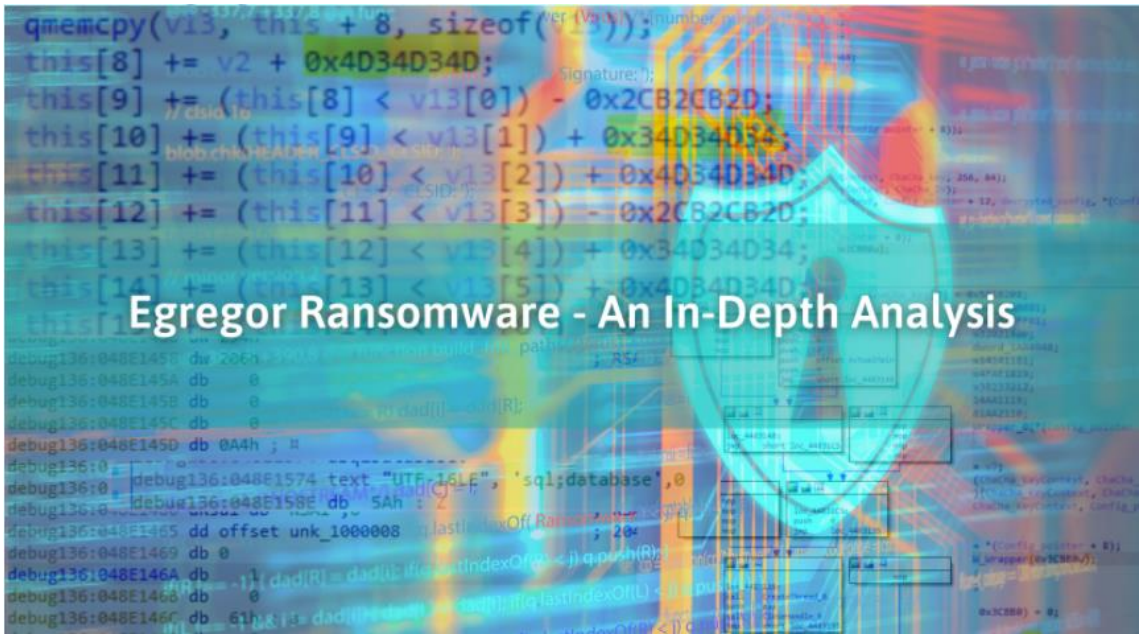


Egregor ランサムウェアについて分析
2020 年 12 月 7 日 Minerva のブログから

調査チームは Egregor ランサムウェアについて感染手口のプロセス及び今後さらに進化していくランサムウェア脅威を見据え、攻撃手法や検知回避など全てに及び徹底的に調査を実施しました。

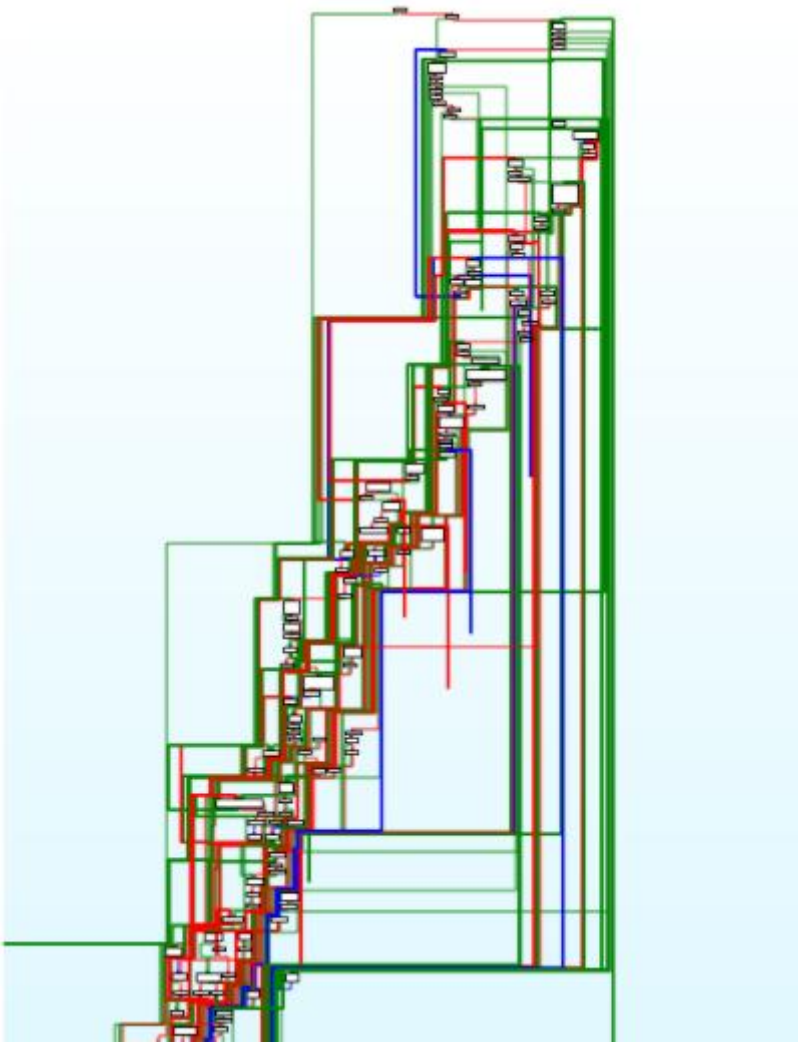
Egregor ランサムウェアの感染拡大とエクスプロイトコードが Sekhmet 或いは Maze ランサムウェアに類似している点を確認しました。さら Maze や Egregor ランサムなどは悪意コードが難読化された手法で組み込まれており、分析するのに多くの時間を費やす結果となりました。

今回のブログで Egregor ランサムウェアについてご紹介します。今回の調査で難読化されたランサムウェアを Minerva レスポンスチームが難読コードを解いたプロセスなどもご紹介します。

ローダー
今回のランサムウェアは b.dll という DLL ファイル名で攻撃を実行しようとしました。以下のスクリーンショットは実際のイベントです。

この悪意のエクスプロイトコードを分解するとコンパイラーベース手法で難読化されており、分析に多大な時間を費やすことになりました。

例えば、以下の DllRegisterServer 関数を参照ください。IDA グラフ上でランサムウェアが実行されるまでのプロセスを表示しています。



Egregor の構図は Maze ランサムウェアと同類タイプである事が判明し、Egregor の難読コードと照らし合わせ Blueliv というウエブサイトに掲載されている Maze ランサムコードの解

除スクリプトを参考にして分解することに成功しました。

ローダーは“--nop”というコマンドラインをクエリーして来ます。存在すると分かると攻撃がシャットダウンします。

さらに自己解凍するためにバイナリ・ラージ・オブジェクトが以下のステップに従い復号化されます。

- バイナリ・ラージ・オブジェクトはハードコーディングされたキー（今回のケースは 0x4）で排他的論理和（xor）がデコードされる。

- 排他的論理和のデータは Windows の API 関数である CryptStringToBinaryA を使用してベース 64 でデコードされる。

- ハードコーディングされた鍵と初期化ベクトルでペイロードの最終復号化で使用されるストリーム暗号 ChaCha20 アルゴリズムが初期化される。マルウェア作成者は鍵の変更回数をデフォルトの 20 から 4 のみに決定した。

2 回目のペイロードで DLL ファイルが復号化された後、ページアクセス権（パーミッション）RWX と VirtualAlloc を使用して新規アロケーションへコピー作成されます。

初期ローダーの最終ステージでは、メモリー上にペイロードの準備を行います。マルウェアは復号化されたペイロードがロードされ、次のステージを実行するために CreateThread 関数を使用します。

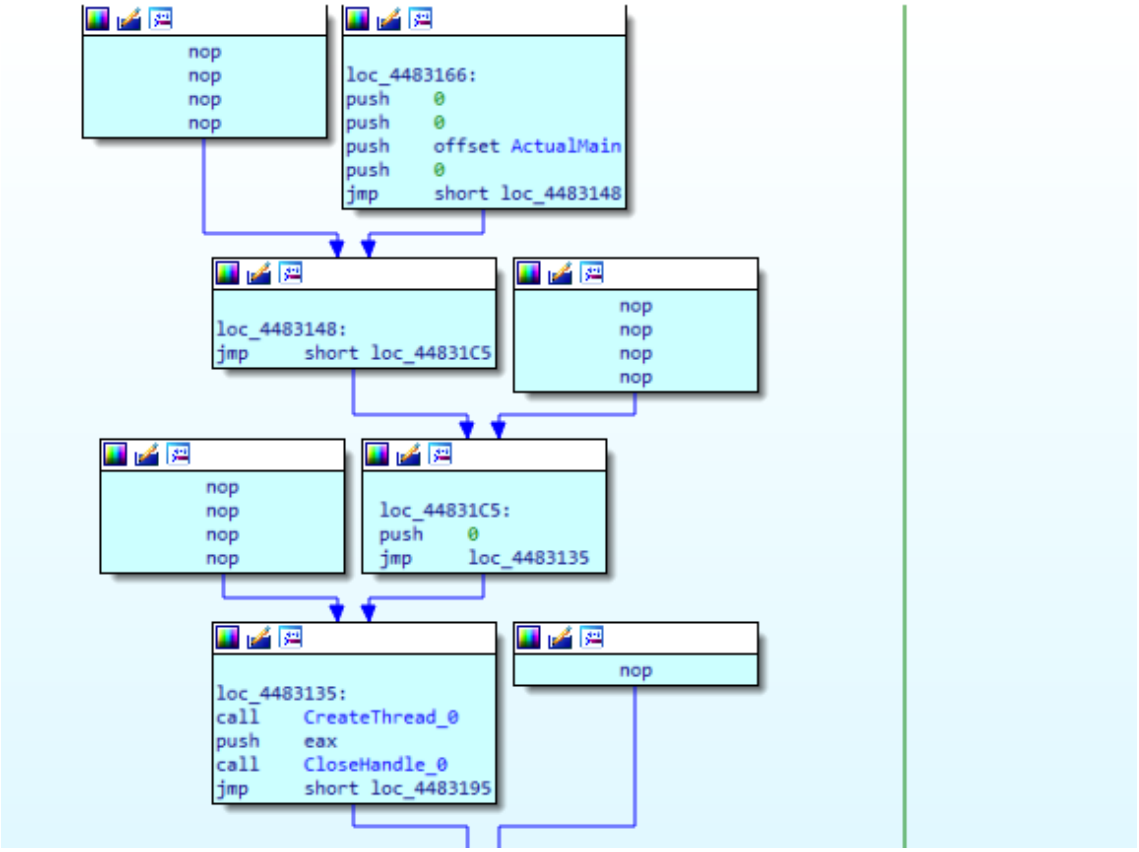
次のステージではコマンドラインの構文解析を行い、ランサムウェアバイナリーを復号するためにパスワードを含むパラミター”-p”を検索します。このランサムウェアはストリーム暗号を使っていくつかの定数を Rabbit 暗号と共有して復号します。

```
qmemcpy(v13, this + 8, sizeof(v13));
this[8] += v2 + 0x4D34D34D;
this[9] += (this[8] < v13[0]) - 0x2CB2CB2D;
this[10] += (this[9] < v13[1]) + 0x34D34D34;
this[11] += (this[10] < v13[2]) + 0x4D34D34D;
this[12] += (this[11] < v13[3]) - 0x2CB2CB2D;
this[13] += (this[12] < v13[4]) + 0x34D34D34;
this[14] += (this[13] < v13[5]) + 0x4D34D34D;
this[15] += (this[14] < v13[6]) - 0x2CB2CB2D;
this[16] = this[15] < v13[7];
```

ランサムウェアコード

ランサムウェアは、「DllEntryPoint」という名前のエクスポートが 1 つだけある DLL ファイルとしてコンパイルされます。

この関数は、ランサムウェアのメインサブルーチンを実行するスレッドを作成します。



ランサムウェアの悪意のあるプログラムが起動する前に関数が呼ばれ、各国の地域コードが決められます。ランサムウェアは3つの異なる API 関数を使用しますが、ロシア又は CIS 諸国にあるコンピューターへ攻撃をしかけないように設定しています。

```
LangId_1 = GetUserDefaultUILanguage();
LangId_2 = GetSystemDefaultLangID();
LangId_3 = GetUserDefaultLangID();
if ( LangId_3 == 0x419
  || LangId_1 == 0x419
  || LangId_2 == 0x419
  || LangId_3 == 0x422
  || LangId_1 == 0x422
  || LangId_2 == 0x422
  || LangId_3 == 0x423
  || LangId_1 == 0x423
  || LangId_2 == 0x423
  || LangId_3 == 0x428
  || LangId_1 == 0x428
  || LangId_2 == 0x428
  || LangId_3 == 0x42B
  || LangId_1 == 0x42B
  || LangId_2 == 0x42B
  || (unsigned __int16)LangId_3 == 0x42C
  || LangId_1 == 0x42C
  || LangId_2 == 0x42C
  || (unsigned __int16)LangId_3 == 0x437
  || LangId_1 == 0x437
  || LangId_2 == 0x437
  || (unsigned __int16)LangId_3 == 0x43F
  || LangId_1 == 0x43F
  || LangId_2 == 0x43F
```

以下のローカルコードを認識した場合、Egregor ランサムウェアは活動を停止します。

Locale Code	Country
0x843	Uzbek - Cyrillic
0x819	Russian - Moldova
0x440	Kyrgyz - Cyrillic
0x442	Turkmen
0x82C	Azerbaijani
0x423	Belarusian
0x42B	Armenian
0x443	Uzbek - Latin
0x43F	Kazakh
0x437	Georgian

0x42C	Azerbaijani
0x818	Romanian - Moldova
0x444	Tatar
0x428	Tajik

ローカルコードを確認した後、ランサム設定は実行可能なデータセクションにあるバッファーから復号化されます。暗号化設定の最初の 8 バイトは復号前の構文解析によってスキップされる PNG ヘッダーと起動されます。8 バイトに続く“d”から始まるワードは復号するためのデータ設定のサイズが包含されています。オフセット 12 から始まり、データ設定は改良型のストリーム暗号 ChaCha20 とハードコーディングされた鍵と初期化ベクトルを使用して復号されます。

暗号化されたインメモリのデータ設定

```
debug121:04932044 unk_4932044 db  89h ; ‰                ; DATA XREF: sub_4912510+1AAto
debug121:04932044                                        ; redundant PNG header
debug121:04932045 db  50h ; P
debug121:04932046 db  4Eh ; N
debug121:04932047 db  47h ; G
debug121:04932048 db  0Dh
debug121:04932049 db  0Ah
debug121:0493204A db  1Ah
debug121:0493204B db  0Ah
debug121:0493204C dd 1A8Ch                               ; size of the configuration
debug121:04932050 db 0A7h ; §                            ; Start of the actual encrypted config
debug121:04932051 db 0E3h ; ã
debug121:04932052 db 0BDh ; ½
debug121:04932053 db  19h
debug121:04932054 db  63h ; c
debug121:04932055 db  64h ; d
```

Configuration クラスの初期化関数の逆コンパイル

```
ChaCha_key[7] = 0x5010208;
ChaCha_key[6] = &unk_3AA0881;
ChaCha_key[5] = &unk_433FF01;
ChaCha_key[4] = 0x32021900;
ChaCha_key[3] = &dword_1AA4948;
ChaCha_key[2] = 0x14141181;
ChaCha_key[1] = 0x4FAE1819;
ChaCha_key[0] = 0x38233212;
ChaCha_IV[1] = 0x14AA1119;
ChaCha_IV[0] = 0x81AA2110;
v7 = Allocate_RW_Wrapper_0(*(Config_pointer + 8));
if ( !v7 )
  break;
decrypted_config = v7;
(ChaCha_Init_Key)(ChaCha_KeyContext, ChaCha_key, 256, 64);
(ChaCha20_init_IV)(ChaCha_KeyContext, ChaCha_IV);
(ChaCha_Encrypt)(ChaCha_KeyContext, Config_pointer + 12, decrypted_config, *(Config_pointer + 8));
if ( v9 || !v9 )
{
  size_of_config = *(Config_pointer + 8);
  v6 = Allocate_RW_Wrapper(0x3CBB0u);
  *(v14 + 4) = v6;
  v8 = 0xFFFC3450;
  do
    *(v6 + v8++ + 0x3CBB0) = 0;
  while ( v8 );
  initilize_config_Class(v17, decrypted_config, size_of_config);
```

構造データはいくつかの興味深い設定がありましたので記載します。

- ランサムノート
- プロセス停止リスト
- アルゴリズムを停止するためのブラックリスト化されたキーワード
- ファイルを暗号化するハードコード RSA2048 ビッド公開鍵
- リモートアドレスの存在を指標するフラッグ

暗号化されたワークステーションのフィンガープリントを作成するために、Egregor ランサムはいくつかの API 関数を使ってマシン情報を抽出します。

```
debug168:04C4001C dd offset dword_4C90000
debug168:04C40020 dd offset aWorkgroup_0              ; "WORKGROUP"
debug168:04C40024 db    0
debug168:04C40025 db    0
debug168:04C40026 db    0
debug168:04C40027 db    0
debug168:04C40028 dd offset aCF164540204218           ; "|C:F_164540/204218|D:C_0/820|"
debug168:04C4002C dd offset aTimi                     ; "timi"
debug168:04C40030 dd offset aWindowsDefende           ; "Windows Defender;"
debug168:04C40034 db    0
debug168:04C40035 db    0
debug168:04C40036 db    0
debug168:04C40037 db    0
debug168:04C40038 db    0
debug168:04C40039 db    0
debug168:04C4003A db    0
debug168:04C4003B db    0
debug168:04C4003C dd offset aWindows10Enter           ; "Windows 10 Enterprise N"
debug168:04C40040 db    0
```

ランサムウェアはデバイスと接続しているロジカルディスクのファイル名とタイプを認識するためや、更にはそれぞれどのくらいの空き容量があるのか特定するために API 関数である”GetLogicalDriveStrings”と“GetDiskFreeSpace”使用します。

ランサムウェアの RSA 暗号の公開鍵は暗号化されたデータ設定に保管されています。

```
debug136:048E1458 dw 206h                             ; RSA PRIVATE KEY
debug136:048E145A db    0
debug136:048E145B db    0
debug136:048E145C db    0
debug136:048E145D db 0A4h ; ¤
debug136:048E145E db    0
debug136:048E145F db    0
debug136:048E1460 aRsa1 db 'RSA1',0                   ; RSA Magic
debug136:048E1465 dd offset unk_1000008               ; 2048 bit
debug136:048E1469 db 0
debug136:048E146A db    1
debug136:048E146B db    0
debug136:048E146C db  61h ; a
debug136:048E146D db    0
```

各実行ファイルにはプライベート鍵と公開鍵のペアーが生成されています。公開鍵は対称鍵を

暗号化するのに利用します。固有の対称鍵は各ファイルを暗号化する際に生成されます。

Egregor の鍵が生成される構図は以下になります。

- 2048 ビット RSA 鍵ペアーは”CryptGenKey”を使用して生成される。これがセッション鍵である。

- API “CryptExportKey”を使用して鍵はエクスポートされる。

- エクスポートされたプライベート鍵は鍵と初期化ベクトルを無作為に利用し ChaCha で暗号化する。

- ChaCha 鍵は”Crypt Encrypt”関数と埋め込まれた RSA 公開鍵データを使って暗号化する。

- 暗号化された ChaCha 鍵と暗号化されたセッション鍵はハードコーディングされたパス上にあるディスクへ保存される（今回のケースは%ProgramData%¥dtb.dat.）。

このランサムウェアは同じプロトコールでセッション鍵を暗号化し、ランサムペイロード（Rabbit）を復号します。これは注目に値します。
このランサムウェアはマシンを暗号化する前に特定のプロセスとサービスを停止させます。ハードコードプロセス名のリストは暗号化された設定ファイルに保管され、マルウェアは稼働しているプロセスを列挙するために”NtQuerySystemInformation”を使用し、停止する時は”NtTerminateProcess”関数を使います。

今回のケースではプロセスリストは停止されました。

```
debug136:035F1592 db    0
debug136:035F1593 aSftesqlExeSqla:
debug136:035F1593 text "UTF-16LE", 'sftesql.exe;sqlagent.exe;sqlbrowser.exe;sqlwriter.e'
debug136:035F1593 text "UTF-16LE", 'xe;oracle.exe;ocssd.exe;dbsnmp.exe;synctime.exe;agn'
debug136:035F1593 text "UTF-16LE", 'tsvc.exe;isqlplussvc.exe;xfssvccon.exe;sqlservr.exe'
debug136:035F1593 text "UTF-16LE", ';mydesktopservice.exe;ocautoupds.exe;encsvc.exe;fir'
debug136:035F1593 text "UTF-16LE", 'efoxconfig.exe;tbirdconfig.exe;mydesktopqos.exe;oco'
debug136:035F1593 text "UTF-16LE", 'mm.exe;mysqld.exe;mysqld-nt.exe;mysqld-opt.exe;dben'
debug136:035F1593 text "UTF-16LE", 'g50.exe;sqbcoreservice.exe;excel.exe;infopath.exe;m'
debug136:035F1593 text "UTF-16LE", 'saccess.exe;mspub.exe;onenote.exe;outlook.exe;power'
debug136:035F1593 text "UTF-16LE", 'pnt.exe;sqlservr.exe;thebat.exe;steam.exe;thebat64.'
debug136:035F1593 text "UTF-16LE", 'exe;thunderbird.exe;visio.exe;winword.exe;wordpad.e'
debug136:035F1593 text "UTF-16LE", 'xe;Q8W32.exe;Q8W64.exe;ipython.exe;wpython.exe;pyth'
debug136:035F1593 text "UTF-16LE", 'on.exe;dumpcap.exe;procmon.exe;procmon64.exe;procex'
debug136:035F1593 text "UTF-16LE", 'p.exe;procexp64.exe',0
debug136:035F1A83 db 60h
```

アルゴリズムの停止に関していえば、ランサムウェア設定においてどのサービスが停止されるのか決定するのに用いられます。サービス名を列挙するために API 関数”EnumServicesStatus”を使用します。ブラックリストに紐づけされたサービスは Windows Service Control Manager API を使用して止めます。以下はサービスリストのキーワードです。

```
debug136:048E1574 text "UTF-16LE", 'sql;database',0
debug136:048E158E db  5Ah ; Z
```

Egregor ランサムウェアは、ハードコードされた HTTP URL へ接続できる特性を備えています。設定にオフセット 0x3a31e と 0x32fb が 0 を含まなければ、このランサムウェアは IP アドレス/DNS 名(設定上に埋め込まれている)へ接続し、改良型ストリーム暗号 ChaCha20 と Base64 を使ってコンテンツをデコードします。

IDAPython の難読化解除スクリプトはこちら

Minerva 製品(Minerva Armor)についてのお問い合わせは Pico Technologies (info@pico-t.co.jp)までお願い致します。