2025 BERLIN

24 - 26 September, 2025 / Berlin, Germany

GOOGLE CALENDAR AS C2 INFRASTRUCTURE: A CHINA-NEXUS CAMPAIGN WITH STEALTHY TACTICS

Tim Chen & Still Hsu

TeamT5, Taiwan

timc@teamt5.org
still@teamt5.org

ABSTRACT

In recent years, China-nexus threat groups have increasingly adopted tactics to obscure their malware footprint, particularly through the use of LOTS (living off trusted sites) and LOLBins (living-off-the-land binaries and scripts). Our latest research has uncovered a new malware variant named Calendarwalk (recently named TOUGHPROGRESS by *Google*). Calendarwalk employs tactics not previously observed within the APT landscape, such as abusing LOTS through *Google Calendar* events and exploiting LOLBins via *Windows Workflow Foundation*. In this paper, we will examine Calendarwalk and the unique techniques it employs, then analyse its connection to Amoeba (widely known as APT41 or Earth Baku), based on our findings.

In December 2024, our team identified two fully undetected (FUD) samples exploiting XOML (Extensible Object Markup Language) in *Windows Workflow Foundation* (*WF*) to execute their payloads. Based on our observations, we believe this is the first documented instance of an APT employing this technique in a real-world scenario. Our analysis of these samples uncovered two shellcode payloads compressed and encoded using a consistent multi-stage compression/encoding chain. One of these payloads was an AES variant of Chatloader (also known as DodgeBox or StealthVector) that was previously associated with Amoeba, and the other was a never-before-seen malware that we have dubbed Calendarwalk.

Our analysis of Calendarwalk encountered significant hurdles posed by its obfuscation techniques, rendering static analysis ineffective on unmodified binaries. After circumventing these defences through targeted assembly patching, we confirmed Calendarwalk's capabilities – a novel C2 mechanism that retrieves and executes commands via *Google Calendar* events. During our research, we also discovered overlaps with Google Calendar RAT (GCR), an open-source proof-of-concept RAT that was published on *GitHub* in 2023, suggesting the malware developer may have taken heavy inspiration from the project.

We believe Calendarwalk is also closely connected to Tabbywalk (also referred to as CurveBack or MoonWalk), a malware family attributed to Amoeba last year. While Calendarwalk leverages *Google Calendar* for its C2 mechanism, Tabbywalk uses *Google Drive* for similar purposes. Both cases also involved the same version of Chatloader. We will explore the relationship between Calendarwalk and Tabbywalk to establish a potential attribution link.

Our research will highlight the evolving tactics and techniques used by the Chinese APT group, emphasizing their increasing reliance on LOTS and LOLBins to achieve their objectives.

INTRODUCTION

Amoeba is one of the most notorious and active Chinese state-sponsored APT groups, whose operations have garnered significant attention from the cybersecurity community. In recent years, we have observed that Amoeba has leveraged public cloud services as command-and-control (C2) servers or as intermediaries for malware distribution. Since 2021, when the Natwalk backdoor (also known as SideWalk or ScrambleCross) used *Google Docs* as dead drop resolvers containing C2 server configurations, Amoeba has demonstrated a clear preference for abusing legitimate cloud services to disguise malicious traffic. The group has also leveraged cloud platforms such as *Google Drive* and *Google Spreadsheets* as C2 servers for malware communication, further exemplifying this approach.

In late 2024, we observed attack activity linked to Amoeba, from which we obtained two malicious samples: Chatloader and a newly identified backdoor named Calendarwalk. Calendarwalk incorporates several innovative techniques, including abuse of the *Windows Workflow Foundation* and a novel method for cross-session process execution from a project named `IhxExec`. Additionally, the backdoor is protected by compiler-level obfuscation. Based on these advanced stealth techniques and obfuscation methods, we believe that Calendarwalk represents a significant advancement in Amoeba's toolkit and is likely to become a powerful weapon in the group's cyber espionage activities.

TECHNICAL ANALYSIS

XOML loading mechanism

The Calendarwalk sample that we observed was named `regedit.exe.xoml`, and the file itself is in the *Windows Workflow Foundation* XOML format. Under normal circumstances, users are expected to utilize *WF* through Visual Studio and define workflows using the XOML format. We discovered that Calendarwalk employs an XOML-based workflow as its initial execution method.

In August 2024, an article entitled 'Sharp4XOMLLoader: 通过执行XOML文件代码绕过安全防护' ('Sharp4XOMLLoader: Executing XOML Files to Bypass Security') [1] was published in the Chinese cybersecurity community and circulated within some public forums including *Weixin*. The article described how the inherent features of *Windows Workflow Foundation* XOML files can be abused to embed malicious code within a workflow, which can then be executed using Sharp4XOMLLoader as a method to evade anti-virus software and EDR detection.

Our analysis suggests that the Sharp4XOMLLoader tool referenced in the article is likely a wrapper or technique that leverages `wfc.exe`, a legitimate and *Microsoft*-signed component included with the *Windows* SDK as the Workflow

Command-line Compiler Tool since 2005. We also observed a high degree of code overlap between the XOML content used in the Calendarwalk sample and the example XOML provided in the article. Based on this evidence, we assess that the threat actor likely referenced or adapted the techniques described in the article to utilize *Workflow Foundation* XOML as a malware execution method.

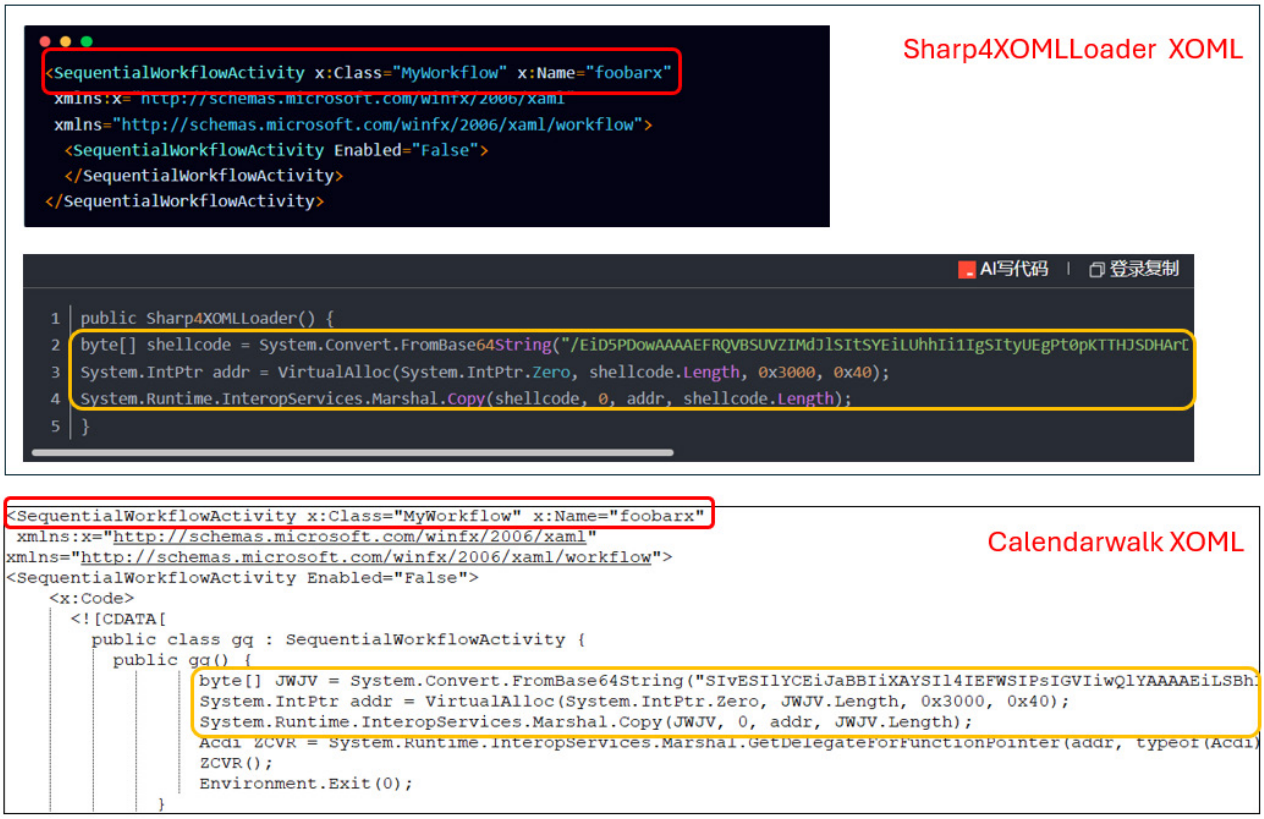


Figure 1: The Sharp4XOMLoader example payload overlaps with Calendarwalk's version.

The Calendarwalk sample leverages the `<x:Code>` tag in *Windows Workflow Foundation* to execute an embedded payload. The complete execution flow of Calendarwalk is illustrated in Figure 2.

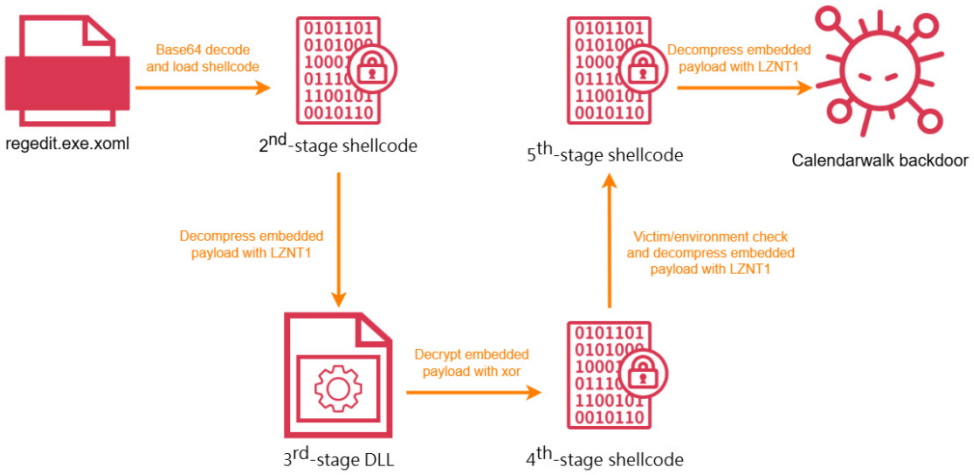


Figure 2: Execution flow of Calendarwalk.

Once executed, the Calendarwalk XOML file performs a Base64 decode on an embedded payload as its second-stage shellcode and executes it in memory. The second-stage shellcode uses the `Mul0x83_add` algorithm to compute its WinAPI

hashes and decompresses the embedded payload with LZNT1 to extract the third-stage DLL, which is then loaded into memory.

```
def compute_hash_mul83_add(function_name):

    hash_value = 0

    for byte in function_name:
        hash_value = (hash_value * 0x83 + byte) & 0xFFFFFFFF

    hash_value &= 0x7FFFFFFF

    return hash_value
```

Figure 3: WinAPI hash function in second-stage shellcode.

The third-stage DLL searches for the specific segment to locate and decrypt the final payload. It first uses `Mul0x21_add` hashing to resolve the necessary *Windows* API functions, then iterates through its segment names, computing a `Mul0x21_add` hash for each segment name using the given `hash_seed`. When the computed hash matches a predefined value, the malware identifies that segment as containing the encrypted payload. Through our analysis, we determined that the `.reloc` segment contains the encrypted payload. This payload is decrypted using an XOR algorithm and subsequently executed as the fourth-stage shellcode.

```
def compute_hash_mul21_add(function_name):

    hash_value = 5903

    for byte in function_name:
        hash_value = (hash_value * 0x21 + byte) & 0xFFFFFFFF

    return hash_value
```

Figure 4: WinAPI hash function in third-stage loader.

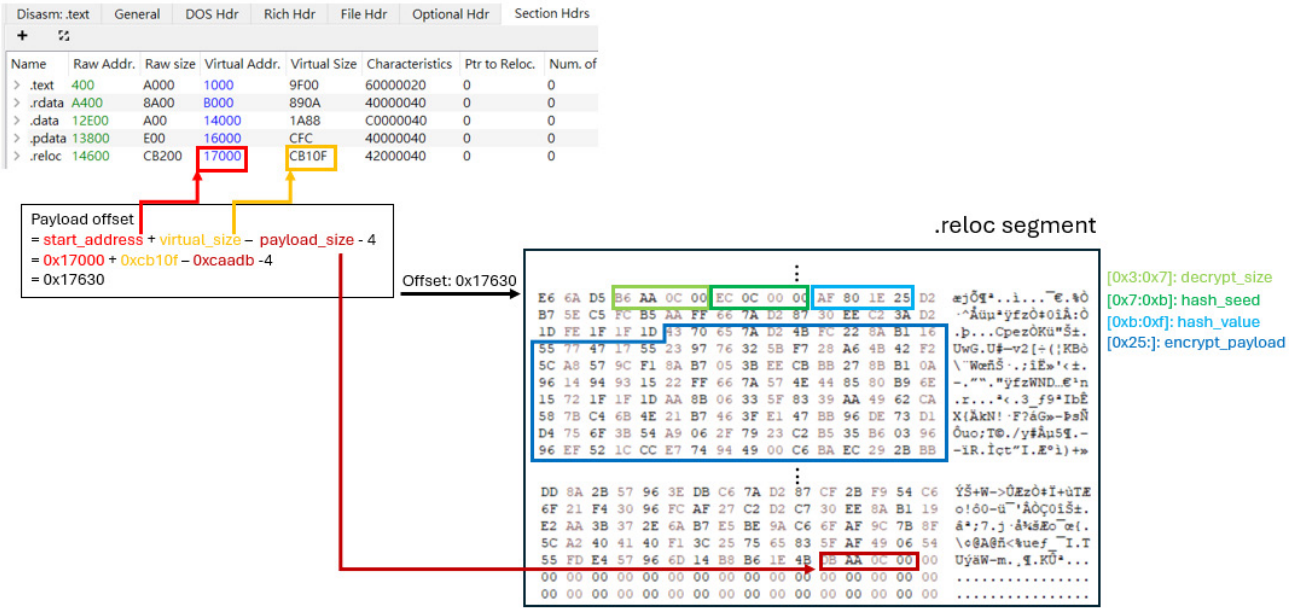


Figure 5: Payload structure in third-stage DLL.

The fourth-stage shellcode functions as an advanced, feature-rich loader. While it employs the same `Mul0x83_add` hashing algorithm as the second-stage shellcode to resolve *Windows* API functions, it extends functionality with additional capabilities, including:

- Anti-debug checks
  - Retrieves COM object with a non-existent CLSID
  - `IsDebuggerPresent()` API call
  - Checks if memory size exceeds 3GB
  - Checks process names for `ida.exe`, `ida64.exe`, `x32dbg.exe`, `x64dbg.exe`, `x96dbg.exe`
- Victim identity checks
  - Compares hostname against a hard-coded name
  - Compares MAC address against a hard-coded value
- Creation of a mutex named `ZLcaU2MeTQJ52Ec`

The structure of configuration in the fourth-stage shellcode is shown in Figure 6.

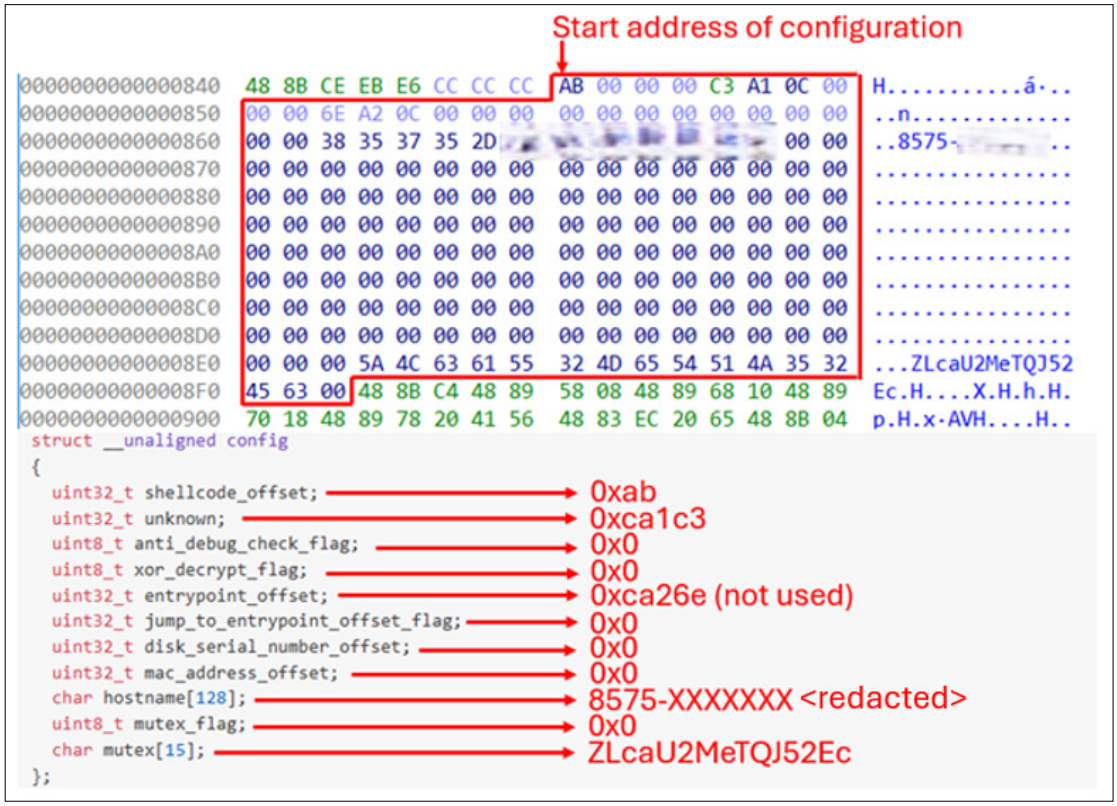


Figure 6: Configuration format of fourth-stage shellcode.

After performing environment and anti-debugging checks, the fourth-stage shellcode decompresses and loads the embedded fifth-stage shellcode using the LZNT1 algorithm. The fifth-stage shellcode is functionally identical to the second-stage shellcode and serves as another shellcode loader, which uses LZNT1 decompression to extract the final Calendarwalk backdoor and execute it in memory.

Calendarwalk

The Calendarwalk backdoor uses *Google Calendar* as its C2 server. Strings in the backdoor are protected using XOR encryption and are decrypted during runtime. However, a small portion of plaintext strings remain within the binary, from which we were able to extract the necessary information for communicating with the *Google Calendar* service – specifically, `client_id`, `client_secret`, `refresh_token` and `calendar_id`. We assess that the presence of these values in plaintext is intentional; the lack of protection on these tokens allows the malware builder to easily replace the token information with actor-specific credentials without having to recompile the entire binary.

Additionally, during analysis of Calendarwalk, we observed the use of an obfuscation technique that employs indirect function calls with arithmetic calculations. This technique prevents most decompilers from producing a readable pseudocode and disrupts the disassembler's ability to reconstruct control flow. The logic of this obfuscation technique is illustrated in Figure 7.

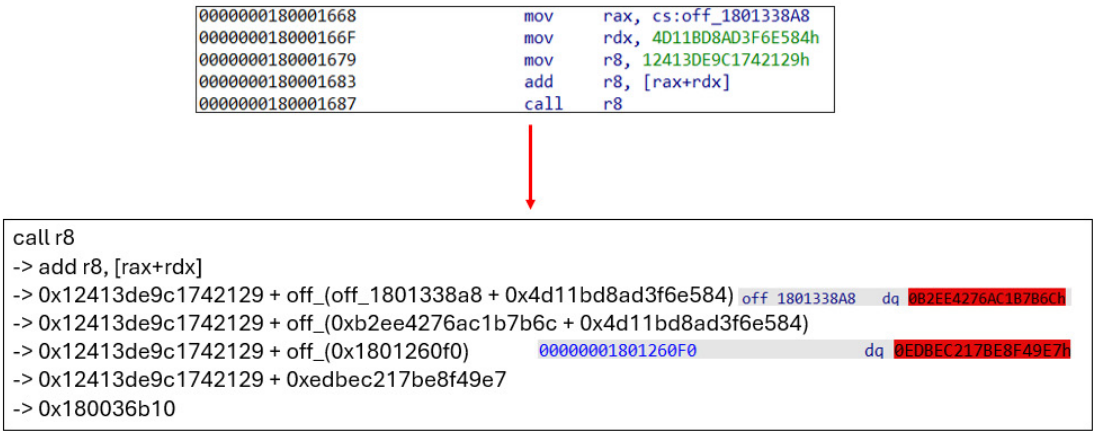


Figure 7: Obfuscation with arithmetic calculation for indirect function call in Calendarwalk.

Based on the logic of obfuscation, we were able to perform equivalent computations to resolve the target address of indirect function calls, allowing us to patch the assembly code accordingly.

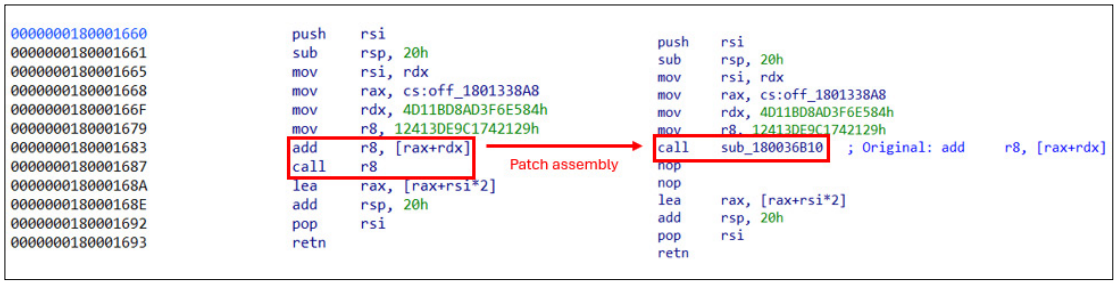


Figure 8: Original assembly vs. patched assembly.

However, due to compiler optimizations, we encountered issues such as register reuse in functions and conditional jumps with varying paths during assembly code patching. These complications hindered full automation of the deobfuscation process. To address this challenge, we analysed the obfuscation logic and manually patched the assembly code using flare-emu [2], thereby significantly improving code readability.

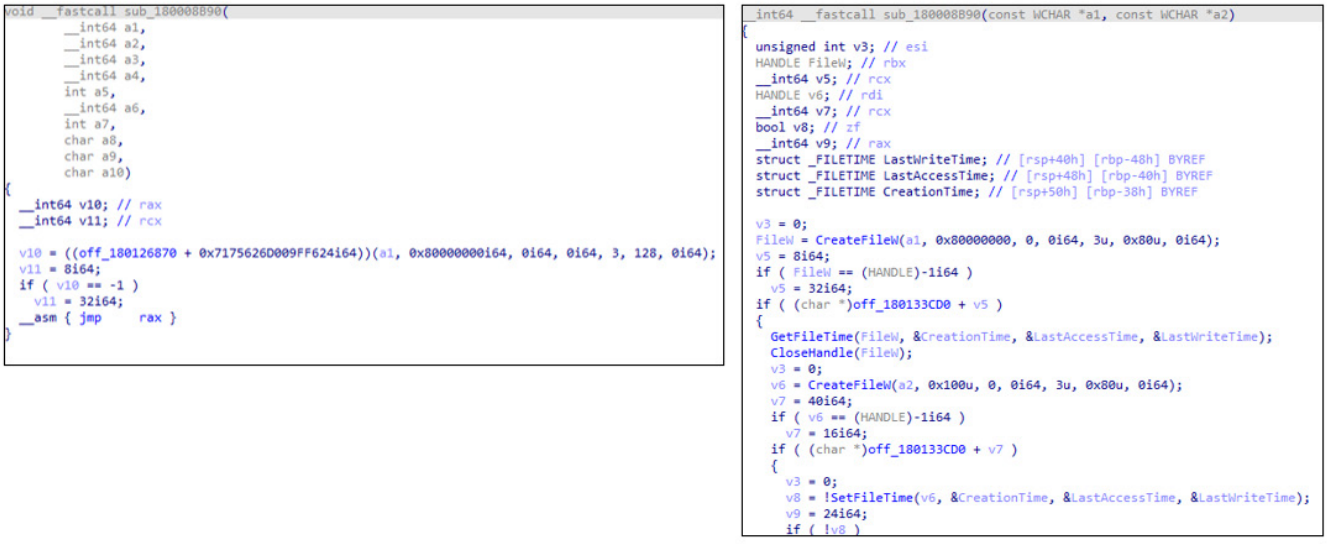


Figure 9: Comparison of original function (left) vs. patched function (right) in Calendarwalk.

After some levels of deobfuscation, we were able to analyse the main behaviour of Calendarwalk. Based on our observations, the operational flow of Calendarwalk can be summarized as follows:

1. Obtains an access token using the provided token information.
2. Collects victim information and transmits it to *Google Calendar* in encrypted format.

3. Spawns multiple threads, including:
    a. A thread responsible for executing commands received from *Google Calendar*.
    b. Multiple threads used for executing command-line instructions through the command pipe mechanism.
4. Receives and executes commands, either directly to the command pipe or based on specific `command_id`.

We also analysed the victim information collected by Calendarwalk. The data fields are separated by the pipe (`|`) character and include the following:

- WAN IP (retrieved via `api.ipify.org`)
- LAN IP
- Username
- Computer name
- Process ID
- Processor architecture
- Boot time
- Execution path
- OS version
- Domain group

After transmitting victim information, Calendarwalk continuously monitors *Google Calendar* for newly created events. When a new event is detected, Calendarwalk attempts to retrieve and decrypt the event content to extract commands issued by the attacker. These commands can appear in one of two forms:

- A command to be executed via `cmd.exe`
- A structured command containing a specific `command_id`, used to invoke a corresponding function:

Command ID (decimal)	**Description**
0	Exit process
1	Set sleep interval
4	List files
5	Set current directory
6	Move file
7	Copy file
8	Delete file/directory
9	Create directory
10	Unknown
11	Set file time or copy file time
12	Update victim information
13	Move file (Same as command ID 6)
17	List disk drive information
18	List process information
19	Kill process by PID
20	Impersonate the specified process
21	Registry management sub-command 1: List registry sub-command 2: Write registry value sub-command 3: Write registry key sub-command 4: Delete registry
23	Stop impersonation
24	Impersonate with logon domain/user/password
25	`IHxExec` for cross-session process execution

Table 1: Calendarwalk command table.

The main functionalities supported by Calendarwalk are related to file manipulation, process management, and registry operations. Notably, the command with `command_id 25` leverages `IhxExec` for cross-session process execution. This technique was introduced by the CICADA8 team in mid-2024 [3]. By using `IhxExec`, Calendarwalk is able to execute arbitrary programs under different user contexts without the use of traditional process injection methods, thereby enhancing stealth and evasion.

Upon further examination of the events on the *Google Calendar*, we identified two distinct types:

- Events with the timestamp `2023-07-30T00:00:00`, used to deliver commands to the victim.
- Events with the timestamp `2023-05-30T00:00:00`, used for victim information.

The encrypted payload is embedded in the event description field and is stored in hex-encoded format. The structure of the encrypted data is illustrated in Figure 10.

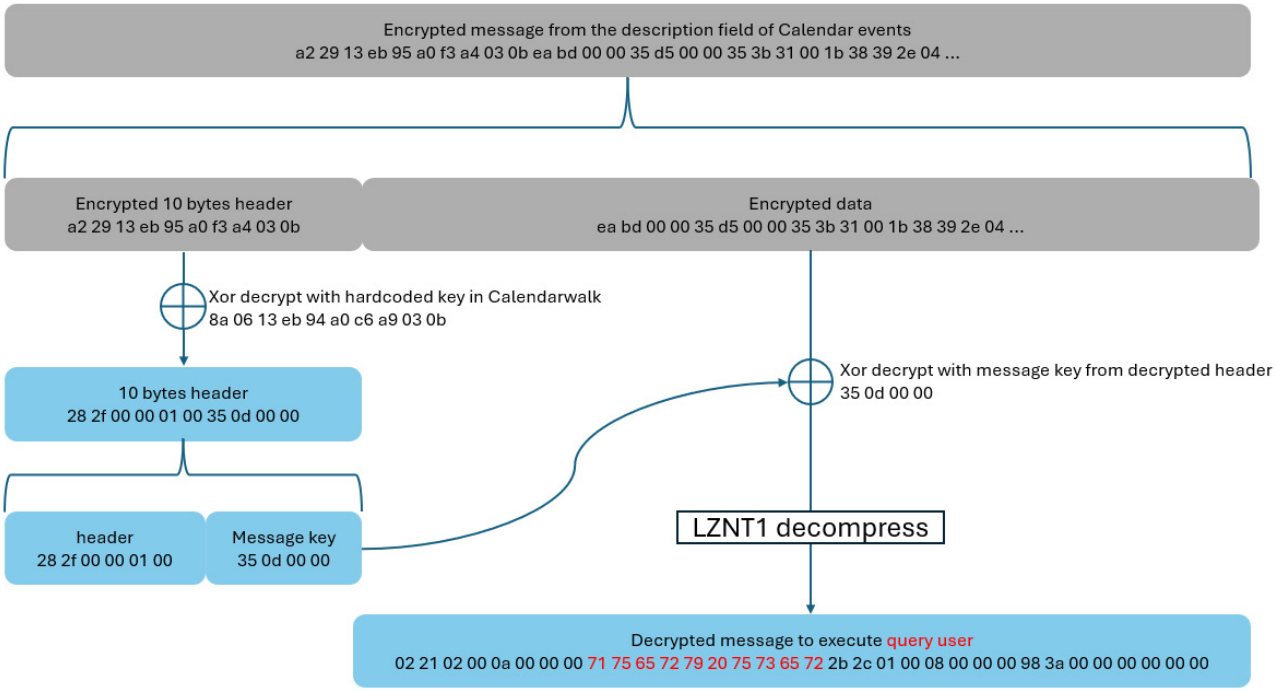


Figure 10: Structure of the victim info and command.

Using the same decryption method, we analysed the encrypted event descriptions on *Google Calendar* and observed that the attacker attempted to execute the `query user` command. Additionally, we identified a response message from a host belonging to the HR system (差勤電子表單) of a Taiwanese IT company, indicating that it was actively reporting the victim information.

RELATED FINDINGS

Following our analysis of Calendarwalk, we would like to share key findings from both the malware technical analysis and incident investigation. These findings include:

- The potential relationship between Calendarwalk and public open-source project Google-Calendar-RAT (GCR).
- The connection between Calendarwalk and another backdoor used by Amoeba, referred to as Tabbywalk.
- Insights into how the actor leveraged the Calendarwalk backdoor within the compromised environment during the intrusion.

Calendarwalk vs. Google-Calendar-RAT (GCR)

Given our interest in Amoeba's use of *Google Calendar* as a C2 server, we conducted a preliminary survey of publicly available research related to malicious abuse of *Google Calendar* services. During this process, we discovered an open-source project named Google-Calendar-RAT (GCR), developed by MrSaighnal [4].

GCR was released in 2023 as a proof-of-concept (PoC) demonstrating the use of *Google Calendar* as a command-and-control channel. Within the codebase, we observed that GCR utilized the timestamp `2023-05-30T00:00:00`, which matches the event time used in Calendarwalk. Furthermore, both GCR and Calendarwalk use the pipe character (`|`) as a field delimiter.

Although GCR is a basic proof-of-concept that lacks encryption or any sophisticated command-handling mechanisms, the similarities in event timestamp and data formatting lead us to assess that Calendarwalk's implementation was likely inspired by or derived from GCR.

```python
def first_connection(summary,service):
    event = {
        'summary': summary,
        'start': {
            'dateTime': '2023-05-30T00:00:00Z',
            'timeZone': 'Europe/Rome',
        },
        'end': {
            'dateTime': '2023-05-30T00:00:00Z',
            'timeZone': 'Europe/Rome',
        },
        'description': 'whoami|'
    }

    created_event = service.events().insert(calendarId=c2Calendar, body=event).execute()
    print(f"[+] New connection initillialized: {created_event['summary']}")
```

```python
                try:
                    # Split the command following the protocol rules
                    command, encoded_result = old_description.split('|')
                except:
                    break
```

Figure 11: Graph event dateTime and Command delimiter in GCR.

Calendarwalk vs. Tabbywalk

When we first spotted Calendarwalk, we noticed that one of the samples spotted within the same victim profile was using the same sets of shellcode as used by Calendarwalk – except that instead of dropping Calendarwalk, it dropped the AES-CFB variant of Chatloader. This loader is configured to retrieve and load its next-stage payload from `C:\ProgramData\USOPrivate.dat`. Based on our internal observations, Chatloader has been actively deployed since at least early 2020 and appears to be used exclusively by Amoeba.

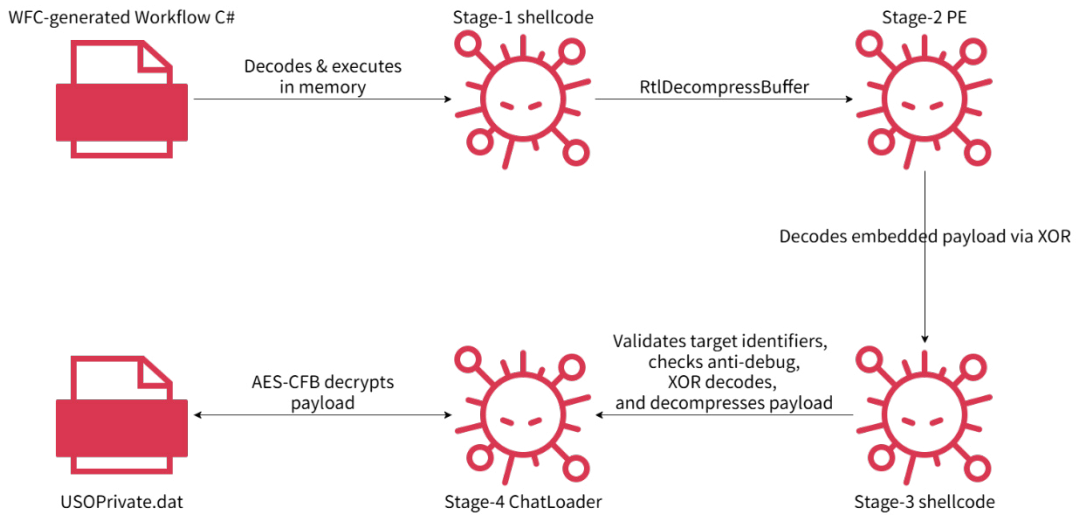


Figure 12: Chatloader targeting a separate payload using a similar loading mechanism.

The presence of Chatloader is particularly interesting given our previous findings, as in late 2023, we discovered another malware family targeting Taiwanese governmental entities that also utilized the same Chatloader variant as its initial loader. This malware, which we have dubbed Tabbywalk (also named Moonwalk by *Zscaler*), shares a critical characteristic with Calendarwalk: both families leverage *Google* services as their primary command-and-control

mechanism. Specifically, Tabbywalk is engineered to exfiltrate data and maintain communications either through a dedicated C2 channel or through *Google Drive* using hard-coded *Google* API credentials. Additionally, a similar module stomping logic to that found in some versions of Chatloader was also spotted in Tabbywalk and in *Trend Micro*'s report on StealthVector published in 2021 [5].

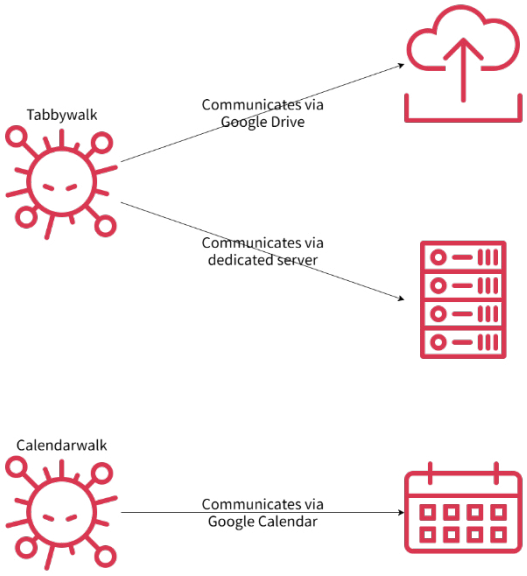
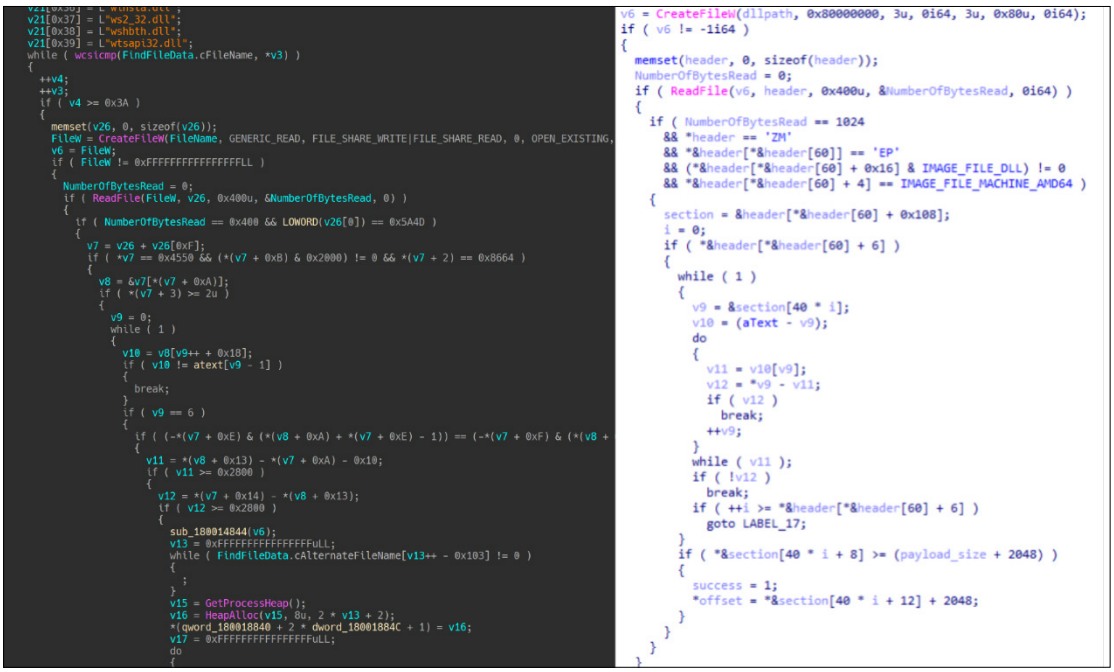


Figure 13: Tabbywalk and Calendarwalk both abuse Google services.



```
v21[0x37] = L"ws2_32.dll";
v21[0x38] = L"wshbth.dll";
v21[0x39] = L"wtsapi32.dll";
while ( wcsicmp(FindFileData.cFileName, *v3) )
{
  ++v4;
  ++v3;
  if ( v4 >= 0x3A )
  {
    memset(v26, 0, sizeof(v26));
    FileW = CreateFileW(FileName, GENERIC_READ, FILE_SHARE_WRITE|FILE_SHARE_READ, 0, OPEN_EXISTING,
    v6 = FileW;
    if ( FileW != 0xFFFFFFFFFFFFFFFFLL )
    {
      NumberOfBytesRead = 0;
      if ( ReadFile(FileW, v26, 0x400u, &NumberOfBytesRead, 0) )
      {
        if ( NumberOfBytesRead == 0x400 && LOWORD(v26[0]) == 0x5A4D )
        {
          v7 = v26 + v26[0xF];
          if ( *v7 == 0x4550 && (*(v7 + 0xB) & 0x2000) != 0 && *(v7 + 2) == 0x8664 )
          {
            v8 = &v7[*(v7 + 0xA)];
            if ( *(v7 + 3) >= 2u )
            {
              v9 = 0;
              while ( 1 )
              {
                v10 = v8[v9++ + 0x10];
                if ( v10 != atext[v9 - 1] )
                {
                  break;
                }
                if ( v9 == 6 )
                {
                  if ( (-*(v7 + 0xE) & (*(v8 + 0xA) + *(v7 + 0xE) - 1)) == (-*(v7 + 0xF) & (*(v8 +
                  {
                    v11 = *(v8 + 0x13) - *(v7 + 0xA) - 0x10;
                    if ( v11 >= 0x2800 )
                    {
                      v12 = *(v7 + 0x14) - *(v8 + 0x13);
                      if ( v12 >= 0x2800 )
                      {
                        sub_180014844(v6);
                        v13 = 0xFFFFFFFFFFFFFFFFuLL;
                        while ( FindFileData.cAlternateFileName[v13++ - 0x103] != 0 )
                        {
                          ;
                        }
                        v15 = GetProcessHeap();
                        v16 = HeapAlloc(v15, 8u, 2 * v13 + 2);
                        *(qword_180018840 + 2 * dword_18001884C + 1) = v16;
                        v17 = 0xFFFFFFFFFFFFFFFFuLL;
                        do
                        {
```

```
v6 = CreateFileW(dllpath, 0x80000000, 3u, 0i64, 3u, 0x80u, 0i64);
if ( v6 != -1i64 )
{
  memset(header, 0, sizeof(header));
  NumberOfBytesRead = 0;
  if ( ReadFile(v6, header, 0x400u, &NumberOfBytesRead, 0i64) )
  {
    if ( NumberOfBytesRead == 1024
      && *header == 'ZM'
      && *&header[*&header[60]] == 'EP'
      && (*&header[*&header[60] + 0x16] & IMAGE_FILE_DLL) != 0
      && *&header[*&header[60] + 4] == IMAGE_FILE_MACHINE_AMD64 )
    {
      section = &header[*&header[60] + 0x108];
      i = 0;
      if ( *&header[*&header[60] + 6] )
      {
        while ( 1 )
        {
          v9 = &section[40 * i];
          v10 = (aText - v9);
          do
          {
            v11 = v10[v9];
            v12 = *v9 - v11;
            if ( v12 )
              break;
            ++v9;
          }
          while ( v11 );
          if ( !v12 )
            break;
          if ( ++i >= *&header[*&header[60] + 6] )
            goto LABEL_17;
        }
        if ( *&section[40 * i + 8] >= (payload_size + 2048) )
        {
          success = 1;
          *offset = *&section[40 * i + 12] + 2048;
        }
      }
    }
  }
}
```

Figure 14: Module stomping similarities in Tabbywalk (left) and Chatloader (right [5]).

Real-world cases

Throughout our analysis of Calendarwalk activities in 2024 and 2025, we identified several Taiwanese victims.

In October/November 2024, we received an incident report involving a Taiwanese victim operating in the IT industry. During the investigation, we found that the threat actor had initially compromised a web server through an unknown attack vector (referred to as Endpoint #1). The actor subsequently performed a lateral movement to gain access to another *Windows* machine elsewhere in the network (referred to as Endpoint #2). An overview of the attack flow is shown in Figure 15.

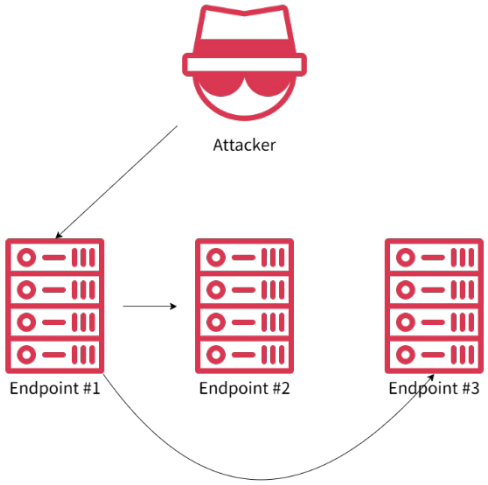


Figure 15: Overview of the attack flow.

After establishing control over Endpoint #2, the attacker dropped `Microsoft.DRM.ApplicationServices.dll` on the endpoint and executed

```
schtasks /create /tn winupdate /sc minute /mo 5 /tr "C:\Windows\Microsoft.NET\Framework64\
v4.0.30319\InstallUtil.exe /U C:\ProgramData\Microsoft.DRM.ApplicationServices.dll" /ru
system /f
```

which creates a scheduled task named `winupdate` that runs every five minutes with system privileges, executing `InstallUtil.exe` to 'uninstall' the ConfuserEx-obfuscated .NET assembly – when, in reality, the `Uninstall` implementation triggers the loader mechanism.

The loader attempts to load an external `dxdiag.dat` file located either under `C:\ProgramData\` or next to the executing assembly, verifies its MD5 hash, and decodes the payload. While the next stage payload `dxdiag.dat` was not recovered during the response efforts, analysis of the execution flow and loading process reveals strong similarities to the StealthMutant loader described by *Trend Micro* in the aforementioned report [5], suggesting that the TTPs documented in the 2021 report remain actively employed by the threat group.

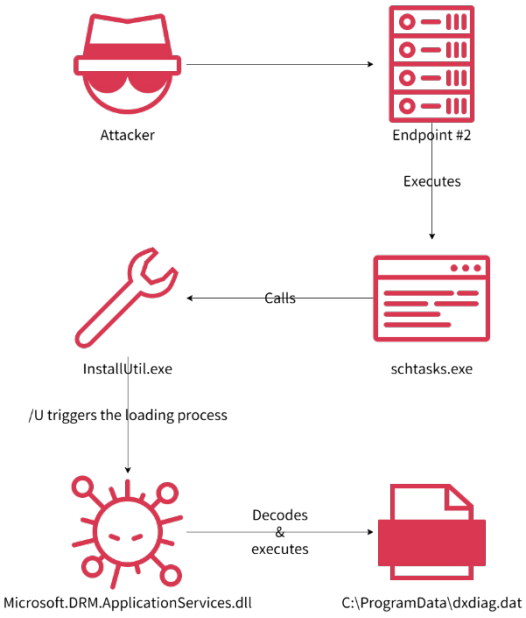


Figure 16: Execution and deployment flow of Microsoft.DRM.ApplicationServices.dll.

Once the above scheduled task had been set up, the attacker retrieved several files from a third compromised system via a UNC network share: `Diagnosis.exe`, `DeElevator64.dll`, and `win.log`. Our analysis later confirmed these files to be components of the Calendarwalk family, with the DLL acting as the loader, and `win.log` as the multi-stage shellcode that

ultimately unpacks Calendarwalk in memory. This incident demonstrated that Calendarwalk was actively being deployed as a post-exploitation toolkit.

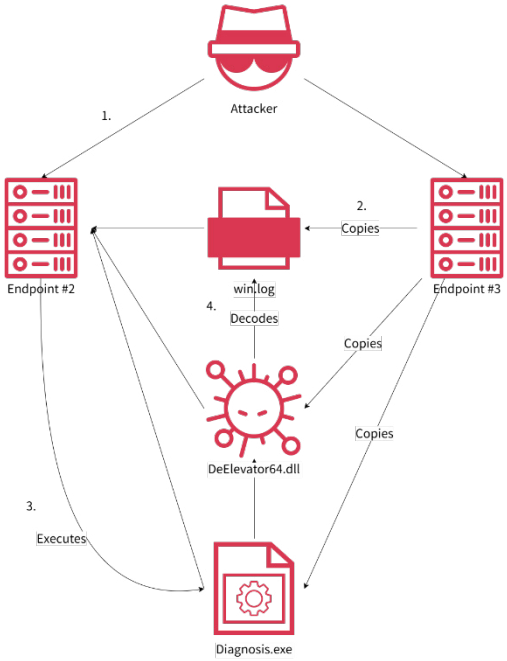


Figure 17: Calendarwalk deployed via Endpoint #3.

In December 2024, our continued research into Calendarwalk led to the discovery of another victim. This time, the victim is a Taiwanese software vendor specializing in ERP (Enterprise Resource Planning) systems. Whilst we do not have details of how the threat group was able to infiltrate the system, Amoeba's targeting of ERP systems has solid precedent, as demonstrated by last year's RevivalStone campaign that was covered by *LAC Co* at VB2024 [6]. The targeting of ERP systems is particularly concerning, as these platforms tend to hold sensitive organizational data and the personal information of various personnel.

CONCLUSION

In this paper, we have presented a comprehensive analysis of Calendarwalk, a new arsenal attributed to the China-nexus group Amoeba. Our research has revealed that the group continues its ongoing efforts to develop new and creative defence and detection evasion techniques, including what appears to be the first documented real-world exploitation of *Windows Workflow Foundation* XOML files by an APT group, as well as the novel abuse of *Google Calendar* events as a command-and-control communication channel.

The technical analysis shows a five-stage execution chain with varying levels of obfuscation, resulting in the final Calendarwalk payload being protected with a convoluted compiler-level obfuscation that necessitates runtime dynamic resolution of addresses and strings for proper analysis, rendering traditional static analysis approaches impractical. This level of protection indicates a mature understanding of modern analysis tools and defensive capabilities, reflecting the threat group's established track record of technical expertise.

From the related findings, we were also able to establish a clear attribution to Amoeba through multiple indicators, including similarities to and deployment of Chatloader variants previously associated with the group, as well as the group's interest in living off trusted sites (LOTS) through trusted web services such as *Google Drive* and *Google Calendar*, and living-off-the-land binaries (LOLBins) through the *Workflow-Foundation*-disguised shellcode execution. It is also evident that the developer(s) of Calendarwalk maintain some level of awareness to the red team communities at large, as demonstrated by the apparent timestamp correlations with the Google Calendar RAT PoC repository and snippets of code from the IhxExec project.

Through the above, we have demonstrated that the threat group has continued its evolution of TTPs over the years, and has committed to keeping researchers on their tails by implementing complicated compiler obfuscation, usage of LOLBins/LOTS, and more defensive evasion techniques.

REFERENCES

[1] dot.Net安全矩阵. Sharp4XOMLLoader: 通过执行XOML文件代码绕过安全防护 . 先知社区. 27 August 2024. https://xz.aliyun.com/news/14870.

[2] Mandiant, Inc. mandiant/flare-emu. GitHub. 27 October 2024. https://github.com/mandiant/flare-emu.

[3] CICADA8. CICADA8-Research/IHxExec. GitHub. 11 July 2024. https://github.com/CICADA8-Research/IHxExec.

[4] MrSaighnal. MrSaighnal/GCR-Google-Calendar-RAT. GitHub. 19 June 2023. https://github.com/MrSaighnal/GCR-Google-Calendar-RAT.

[5] Hiroaki, Hara and Lee, Ted. Earth Baku Returns: Uncovering the Upgraded Toolset Behind the APT Group's New Cyberespionage Campaign. Trend Micro Research. 24 August 2021. https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/earth-baku-returns.

[6] サイバー救急センター. RevivalStone: Winnti Groupによる日本組織を狙った攻撃キャンペーン | LAC WATCH. 株式会社ラック. 13 February 2025. https://www.lac.co.jp/lacwatch/report/20250213_004283.html.