Mark Your Calendar: APT41 Innovative Tactics

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

Google Cloud

Threat Intelligence

Google Threat Intelligence

Visibility and context on the threats that matter most.

Learn more

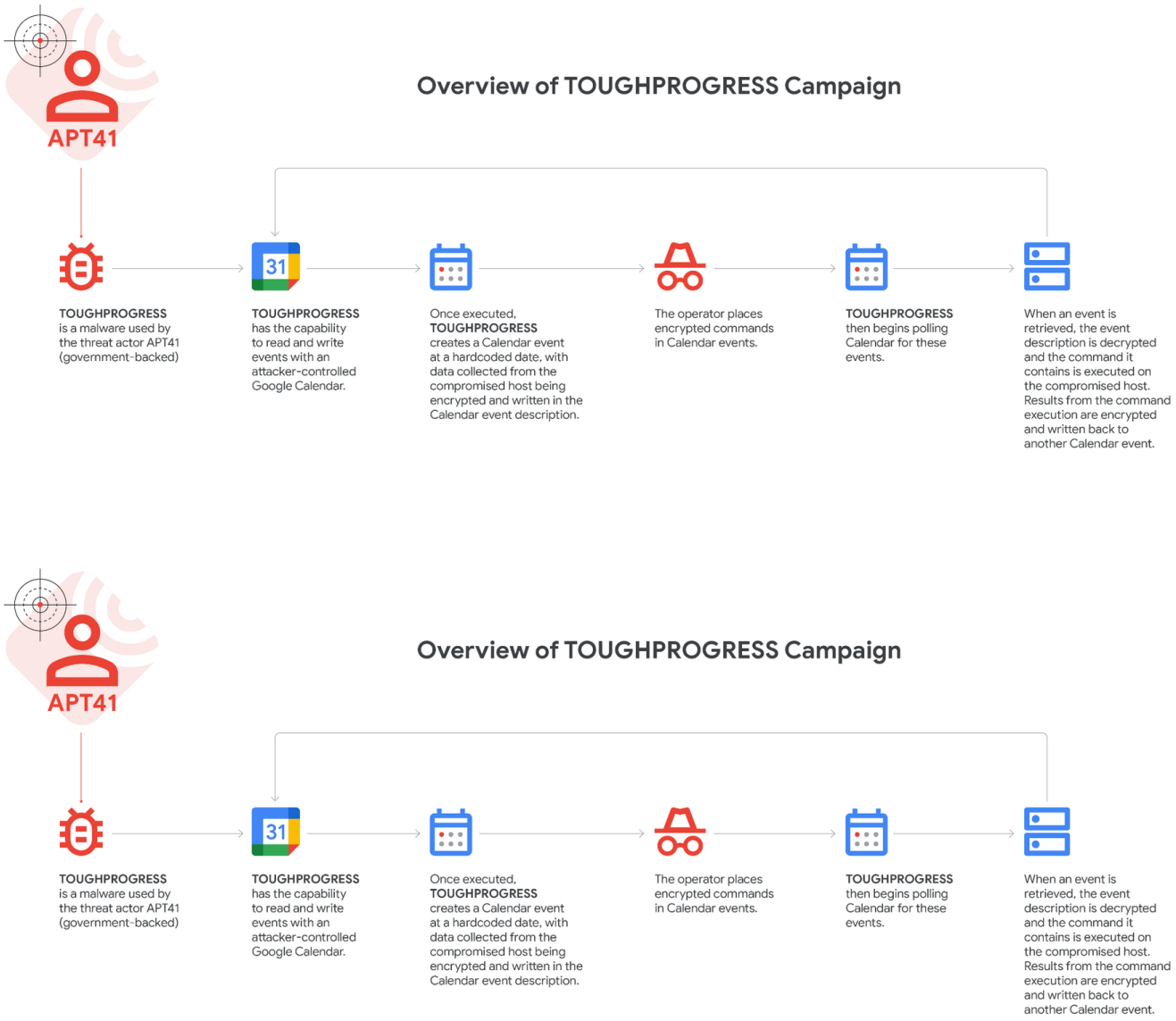
Written by: Patrick Whitsell

Google Threat Intelligence Group's (GTIG) mission is to protect Google's billions of users and Google's multitude of products and services. In late October 2024, GTIG discovered an exploited government website hosting malware being used to target multiple other government entities. The exploited site delivered a malware payload, which we have dubbed "TOUGHPROGRESS", that took advantage of Google Calendar for command and control (C2). Misuse of cloud services for C2 is a technique that many threat actors leverage in order to blend in with legitimate activity.

We assess with high confidence that this malware is being used by the PRC based actor APT41 (also tracked as HOODOO). APT41's targets span the globe, including governments and organizations within the global shipping and logistics, media and entertainment, technology, and automotive sectors.

Overview

In this blog post we analyze the malware delivery methods, technical details of the malware attack chain, discuss other recent APT41 activities, and share indicators of compromise (IOCs) to help security practitioners defend against similar attacks. We also detail how GTIG disrupted this campaign using custom detection signatures, shutting down attacker-controlled infrastructure, and protections added to Safe Browsing.



APT41
Overview of TOUGHPROGRESS Campaign
TOUGHPROGRESS
is a malware used by the threat actor APT41 (government-backed)
TOUGHPROGRESS
has the capability to read and write events with an attacker-controlled Google Calendar.
Once executed, TOUGHPROGRESS creates a Calendar event at a hardcoded date, with data collected from the compromised host being encrypted and written in the Calendar event description.
The operator places encrypted commands in Calendar events.
TOUGHPROGRESS
then begins polling Calendar for these events.
When an event is retrieved, the event description is decrypted and the command it contains is executed on the compromised host. Results from the command execution are encrypted and written back to another Calendar event.

Figure 1: TOUGHPROGRESS campaign overview

Delivery

APT41 sent spear phishing emails containing a link to the ZIP archive hosted on the exploited government website. The archive contains an LNK file, masquerading as a PDF, and a directory. Within this directory we find what looks like seven JPG images of arthropods. When the payload is executed via the LNK, the LNK is deleted and replaced with a decoy PDF file that is displayed to the user indicating these species need to be declared for export.

```
$ unzip -l 出境海關申報清單.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2024-10-23 11:00   image/
    12633  2024-10-23 10:53   image/1.jpg
    10282  2024-10-23 10:54   image/2.jpg
     8288  2024-10-23 10:54   image/3.jpg
     4174  2024-10-23 10:54   image/4.jpg
   181656  2024-10-23 10:54   image/5.jpg
   997111  2024-10-23 11:00   image/6.jpg
   124928  2024-10-23 11:00   image/7.jpg
    88604  2024-10-23 11:03   申報物品清單.pdf.lnk
---------                     -------
  1427676                     9 files
```

The files “6.jpg” and “7.jpg” are fake images. The first file is actually an encrypted payload and is decrypted by the second file, which is a DLL file launched when the target clicks the LNK.

Malware Infection Chain

This malware has three distinct modules, deployed in series, each with a distinct function. Each module also implements stealth and evasion techniques, including memory-only payloads, encryption, compression, process hollowing, control flow obfuscation, and leveraging Google Calendar for C2.

1. **PLUSDROP** - DLL to decrypt and execute the next stage in memory.
2. **PLUSINJECT** - Launches and performs process hollowing on a legitimate “svchost.exe” process, injecting the final payload.
3. **TOUGHPROGRESS** - Executes actions on the compromised Windows host. Uses Google Calendar for C2.

TOUGHPROGRESS Analysis

TOUGHPROGRESS begins by using a hardcoded 16-byte XOR key to decrypt embedded shellcode stored in the sample’s “.pdata” region. The shellcode then decompresses a DLL in memory using COMPRESSION_FORMAT_LZNT1. This DLL layers multiple obfuscation techniques to obscure the control flow.

1. Register-based Indirect Calls
2. Dynamic Address Arithmetic
3. 64-bit register overflow
4. Function Dispatch Table

The registered-based indirect call is used after dynamically calculating the address to store in the register. This calculation involves two or more hardcoded values that intentionally overflow the 64-bit register. Here is an example calling CreateThread.

```
.text:000000018009F563 mov     rbx, 38AC9AC0BDDA8007h
.text:000000018009F56D mov     rax, cs:off_18013DEF0 ; 0C7536540C234EA39h
.text:000000018009F574 add     rax, rbx        ; rax = 0x00000001800F6A40
.text:000000018009F577 mov     [rsp+1960h+var_1938], 0 ; ThreadId
.text:000000018009F580 mov     dword ptr [rsp+1960h+var_1940], 0 ; Run immediately
.text:000000018009F588 lea     r8, lpStartAddress
.text:000000018009F58F xor     ecx, ecx
.text:000000018009F591 xor     edx, edx
.text:000000018009F593 xor     r9d, r9d
.text:000000018009F596 call    rax             ; CreateThread in Dispatch Table
.text:000000018009F563 mov     rbx, 38AC9AC0BDDA8007h
.text:000000018009F56D mov     rax, cs:off_18013DEF0 ; 0C7536540C234EA39h
.text:000000018009F574 add     rax, rbx        ; rax = 0x00000001800F6A40
.text:000000018009F577 mov     [rsp+1960h+var_1938], 0 ; ThreadId
.text:000000018009F580 mov     dword ptr [rsp+1960h+var_1940], 0 ; Run immediately
.text:000000018009F588 lea     r8, lpStartAddress
.text:000000018009F58F xor     ecx, ecx
.text:000000018009F591 xor     edx, edx
.text:000000018009F593 xor     r9d, r9d
.text:000000018009F596 call    rax             ; CreateThread in Dispatch Table
```

Figure 2: Register-based indirect call with dynamic address arithmetic and 64-bit overflow

We can reproduce how this works using Python “ctypes” to simulate 64-bit register arithmetic. Adding the two values together overflows the 64-bit address space and the result is the address of the function to be called.

```
remnux@remnux:~$ python3 -q
>>> import ctypes
>>> rbx = 0x38AC9AC0BDDA8007
>>> rax = 0x0C7536540C234EA39
>>> hex(rax + rbx)
'0x100000001800f6a40'
>>> hex(ctypes.c_int64(rax + rbx).value)
'0x1800f6a40'
remnux@remnux:~$ python3 -q
>>> import ctypes
>>> rbx = 0x38AC9AC0BDDA8007
>>> rax = 0x0C7536540C234EA39
>>> hex(rax + rbx)
'0x100000001800f6a40'
>>> hex(ctypes.c_int64(rax + rbx).value)
'0x1800f6a40'
```

Figure 3: Demonstration of 64-bit address overflow

```
.text:00000001800F6A40 ; HANDLE __stdcall CreateThread(
.text:00000001800F6A40 CreateThread proc near
.text:00000001800F6A40 jmp     cs:__imp_CreateThread
.text:00000001800F6A40 CreateThread endp
.text:00000001800F6A40 ; HANDLE __stdcall CreateThread(
.text:00000001800F6A40 CreateThread proc near
.text:00000001800F6A40 jmp     cs:__imp_CreateThread
.text:00000001800F6A40 CreateThread endp
```

Figure 4: CreateThread in Dispatch Table

These obfuscation techniques manifest as a Control Flow Obfuscation tactic. Due to the indirect calls and arithmetic operations, the disassembler cannot accurately recreate a control flow graph.

Calendar C2

TOUGHPROGRESS has the capability to read and write events with an attacker-controlled Google Calendar. Once executed, TOUGHPROGRESS creates a zero minute Calendar event at a hardcoded date, 2023-05-30, with data collected from the compromised host being encrypted and written in the Calendar event description.

The operator places encrypted commands in Calendar events on 2023-07-30 and 2023-07-31, which are predetermined dates also hardcoded into the malware. TOUGHPROGRESS then begins polling Calendar for these events. When an event is retrieved, the event description is decrypted and the command it contains is executed on the compromised host. Results from the command execution are encrypted and written back to another Calendar event.

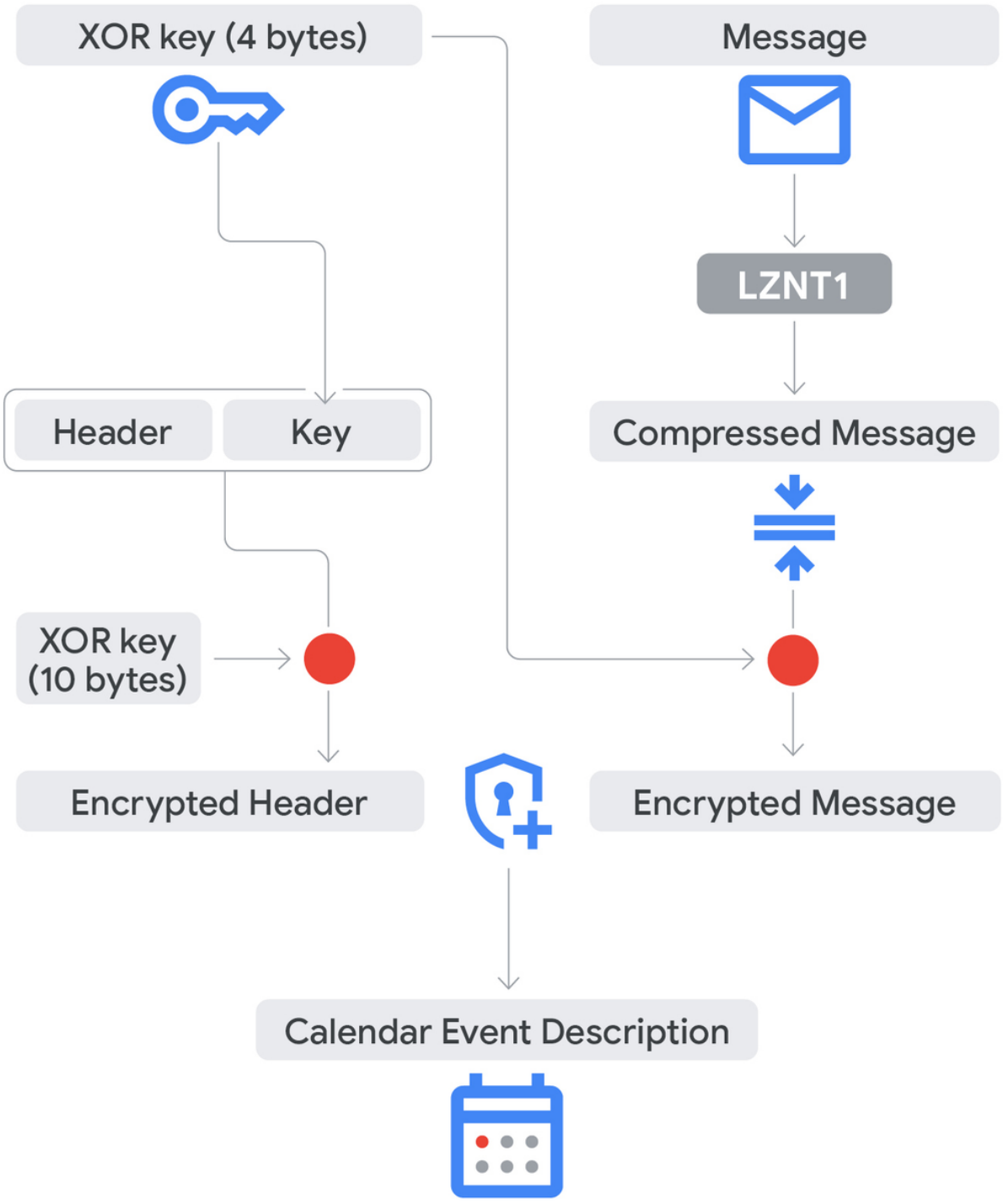
In collaboration with the Mandiant FLARE team, GTIG reverse engineered the C2 encryption protocol leveraged by TOUGHPROGRESS. The malware uses a hardcoded 10-byte XOR key and generates a per-message 4-byte XOR key.

1. Compress message with LZNT1
2. Encrypt the message with a 4-byte XOR key
3. Append the 4-byte key at the end of a message header (10 bytes total)
4. Encrypt the header with the 10-byte XOR key

5. Prepend the encrypted header to the front of the message

6. The combined encrypted header and message is the Calendar event description

Encryption Routine



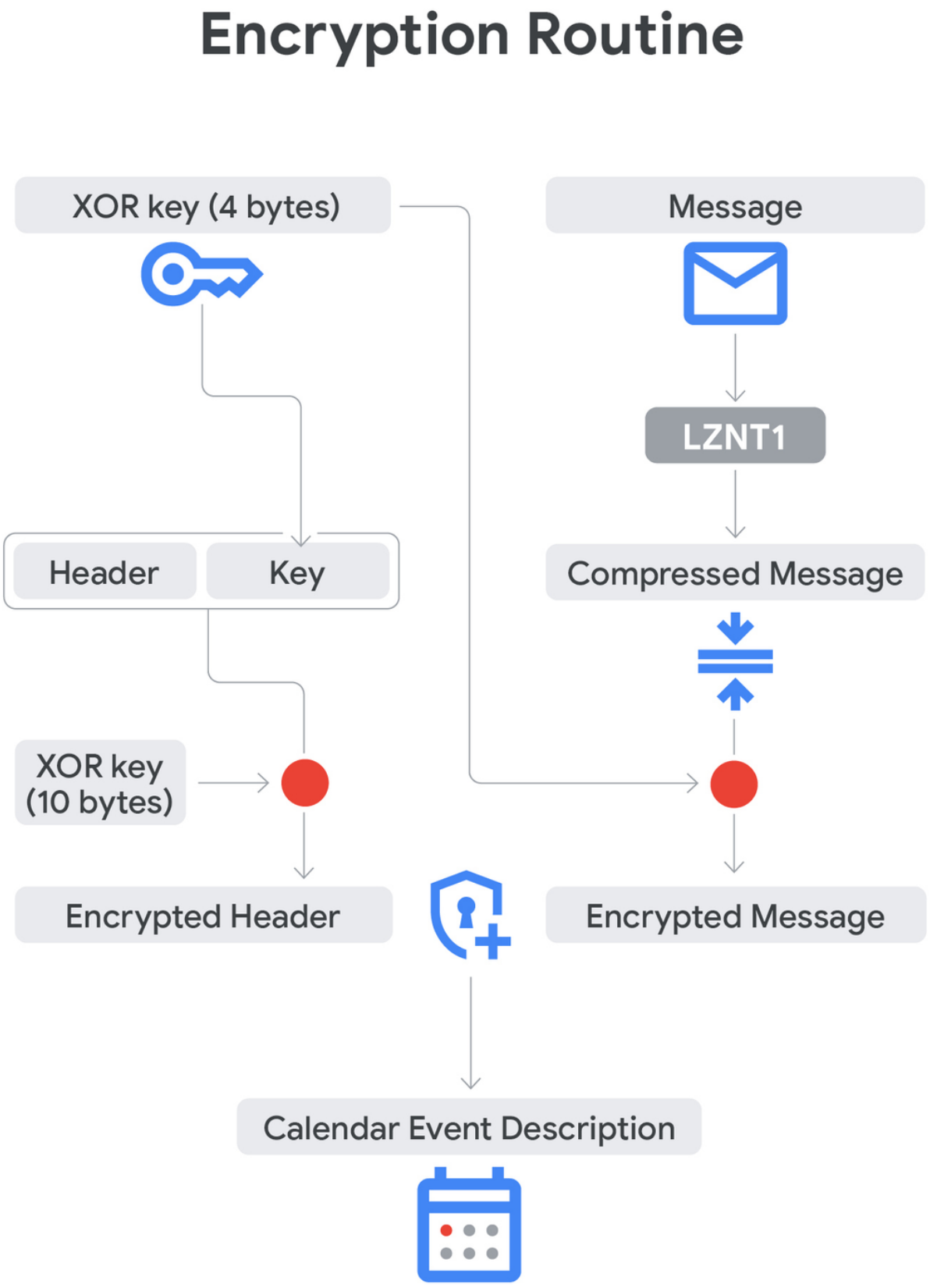
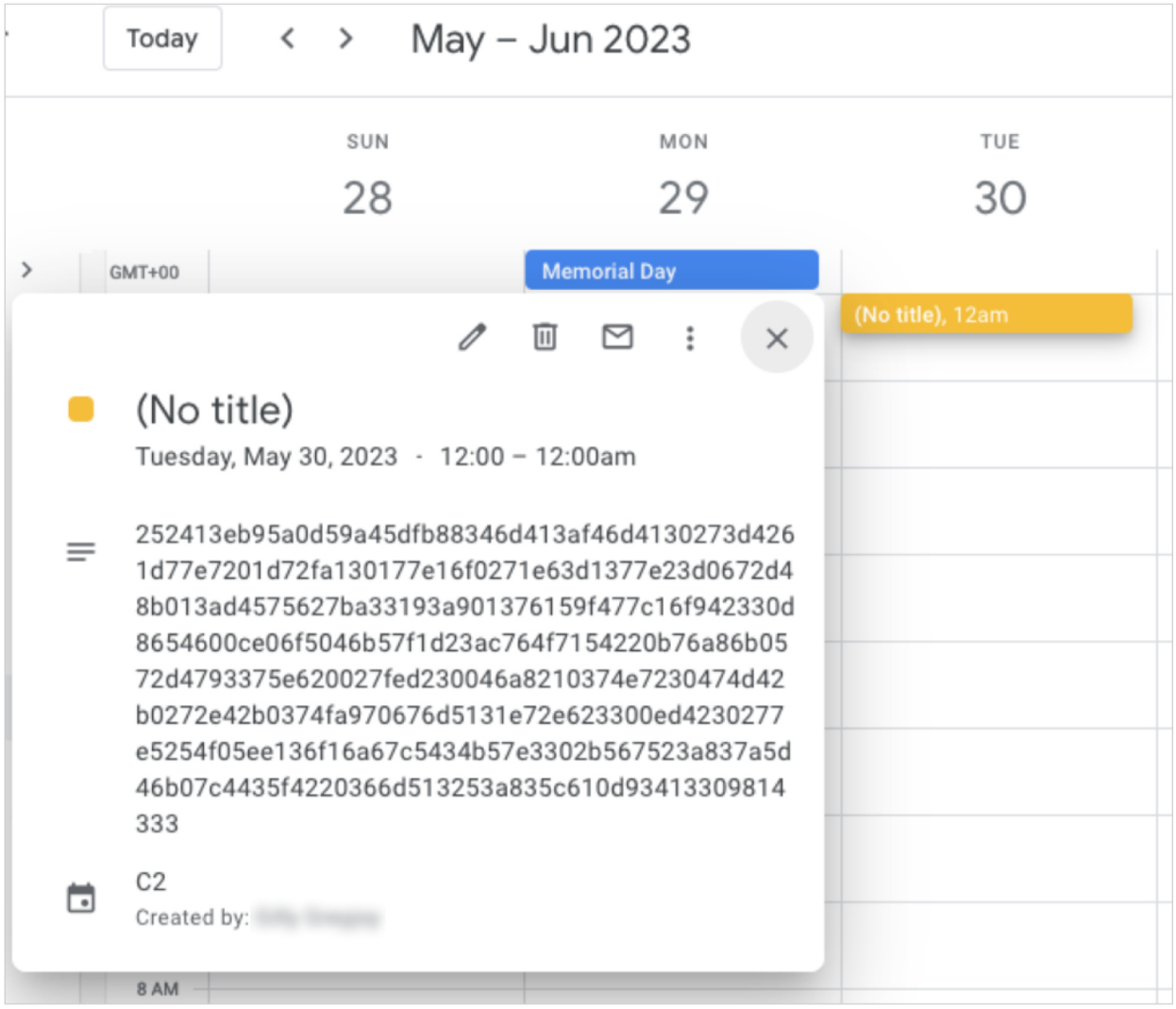


Figure 5: TOUGHPROGRESS encryption routine for Calendar Event Descriptions


Today
May – Jun 2023
SUN
28
MON
29
TUE
30
GMT+00
Memorial Day
(No title), 12am
(No title)
Tuesday, May 30, 2023 · 12:00 – 12:00am
252413eb95a0d59a45dfb88346d413af46d4130273d426
1d77e7201d72fa130177e16f0271e63d1377e23d0672d4
8b013ad4575627ba33193a901376159f477c16f942330d
8654600ce06f5046b57f1d23ac764f7154220b76a86b05
72d4793375e620027fed230046a8210374e7230474d42
b0272e42b0374fa970676d5131e72e623300ed4230277
e5254f05ee136f16a67c5434b57e3302b567523a837a5d
46b07c4435f4220366d513253a835c610d93413309814
333
C2
Created by:
8 AM

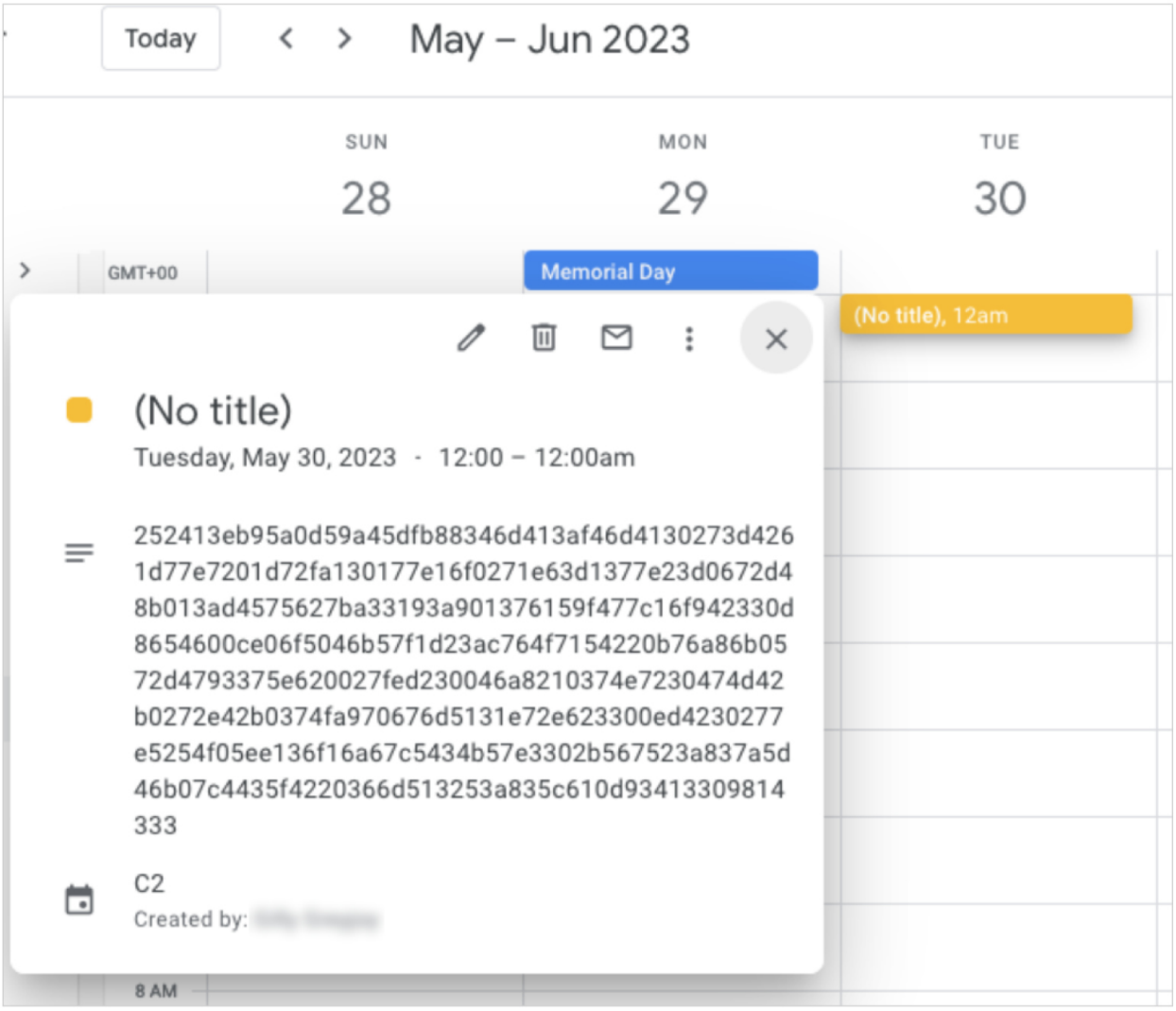


Figure 6: Example of a Calendar event created by TOUGHPROGRESS

Disrupting Attackers to Protect Google, Our Users, and Our Customers

GTIG's goal is not just to monitor threats, but to counter and disrupt them. At Google, we aim to protect our users and customers at scale by proactively blocking malware campaigns across our products.

To disrupt APT41 and TOUGHPROGRESS malware, we have developed custom fingerprints to identify and take down attacker-controlled Calendars. We have also terminated attacker-controlled Workspace projects, effectively dismantling the infrastructure that APT41 relied on for this campaign. Additionally, we updated file detections and added malicious domains and URLs to the Google Safe Browsing blocklist.

In partnership with Mandiant Consulting, GTIG notified the compromised organizations. We provided the notified organizations with a sample of TOUGHPROGRESS network traffic logs, and information about the threat actor, to aid with detection and incident response.

Protecting Against Ongoing Activity

GTIG has been actively monitoring and protecting against APT41's attacks using Workspace apps for several years. This threat group is known for their creative malware campaigns, sometimes leveraging Workspace apps.

- Google Cloud's Office of the CISO published the April 2023 Threat Horizons Report detailing HOODOO's use of Google Sheets and Google Drive for malware C2.
- In October 2024, Proofpoint published a report attributing the VOLDEMORT malware family to APT41.
- The DUSTTRAP malware family, reported by GTIG and Mandiant in July of 2024, used Public Cloud hosting for C2.

In each case, GTIG identified and terminated the attacker-controlled Workspace projects and infrastructure APT41 relied on for these campaigns.

Free Web Hosting Infrastructure

Since at least August 2024, we have observed APT41 using free web hosting tools for distributing their malware. This includes VOLDEMORT, DUSTTRAP, TOUGHPROGRESS and likely other payloads as well. Links to these free hosting sites have been sent to hundreds of targets in a variety of geographic locations and industries.

APT41 has used Cloudflare Worker subdomains the most frequently. However, we have also observed use of InfinityFree and TryCloudflare. The specific subdomains and URLs here have been observed in previous campaigns, but may no longer be in use by APT41.

Cloudflare Workers

- word[.]msapp[.]workers[.]dev
- cloud[.]msapp[.]workers[.]dev

TryCloudflare

- term-restore-satisfied-hence[.]trycloudflare[.]com
- ways-sms-pmc-shareholders[.]trycloudflare[.]com

InfinityFree

- resource[.]infinityfreeapp[.]com
- pubs[.]infinityfreeapp[.]com

APT41 has also been observed using URL shorteners in their phishing messages. The shortened URL redirects to their malware hosted on free hosting app subdomains.

- https[:]//lihi[.]cc/6dekU
- https[:]//tinyurl[.]com/hycev3y7
- https[:]//my5353[.]com/nWyTf
- https[:]//reurl[.]cc/WNr2Xy

All domains and URLs in this blog post have been added to the Safe Browsing blocklist. This enables a warning on site access and prevents users from downloading the malware.

Indicators of Compromise

The IOCs in this blog post are also available as a collection in Google Threat Intelligence.

Hashes

Name	Hashes (SHA256 / MD5)
出境海關申報清單.zip	469b534bec827be03c0823e72e7b4da0b84f53199040705da203986ef154406a 876fb1b0275a653c4210aaf01c2698ec
申報物品清單.pdf.lnk	3b88b3efbdc86383ee9738c92026b8931ce1c13cd75cd1cda2fa302791c2c4fb 65da1a9026cf171a5a7779bc5ee45fb1
6.jpg	50124174a4ac0d65bf8b6fd66f538829d1589edc73aa7cf36502e57aa5513360 1ca609e207edb211c8b9566ef35043b6
7.jpg	151257e9dfda476cdafd9983266ad3255104d72a66f9265caa8417a5fe1df5d7 2ec4eeeabb8f6c2970dcbffdcdbd60e3

Domains

- word[.]msapp[.]workers[.]dev
- cloud[.]msapp[.]workers[.]dev

- term-restore-satisfied-hence[.]trycloudflare[.]com
- ways-sms-pmc-shareholders[.]trycloudflare[.]com
- resource[.]infinityfreeapp[.]com
- pubs[.]infinityfreeapp[.]com

URL Shortener Links

- https[:]//lihi[.]cc/6dekU
- https[:]//lihi[.]cc/v3OyQ
- https[:]//lihi[.]cc/5nlgd
- https[:]//lihi[.]cc/edcOv
- https[:]//lihi[.]cc/4z5sh
- https[:]//tinyurl[.]com/mr42t4yv
- https[:]//tinyurl[.]com/hycev3y7
- https[:]//tinyurl[.]com/mpa2c5wj
- https[:]//tinyurl[.]com/3wnz46pv
- https[:]//my5353[.]com/ppOH5
- https[:]//my5353[.]com/nWyTf
- https[:]//my5353[.]com/fPUcX
- https[:]//my5353[.]com/ZwEkm
- https[:]//my5353[.]com/vEWiT
- https[:]//reurl[.]cc/WNr2Xy

Calendar

- 104075625139-l53k83pb6jbbc2qbreo4i5a0vepen41j.apps.googleusercontent.com
- https[:]//www[.]googleapis[.]com/calendar/v3/calendars/ff57964096cadc1a8733cf566b41c9528c89d30edec86326c723932c1e79ebf0@group

YARA Rules

```
rule G_Backdoor_TOUGHPROGRESS_LNK_1 {
        meta:
                author = "GTIG"
                date_created = "2025-04-29"
                date_modified = "2025-04-29"
                md5 = "65da1a9026cf171a5a7779bc5ee45fb1"
                rev = 1
        strings:
                $marker = { 4C 00 00 00 }
                $str1 = "rundll32.exe" ascii wide
                $str2 = ".\\image\\7.jpg,plus" wide
                $str3 = "%PDF-1"
                $str4 = "PYL="
        condition:
                $marker at 0 and all of them
}
```

```
rule G_Dropper_PLUSDROP_1 {
        meta:
                author = "GTIG"
                date_created = "2025-04-29"
                date_modified = "2025-04-29"
                md5 = "9492022a939d4c727a5fa462590dc0dd"
                rev = 1
        strings:
                $decrypt_and_launch_payload = { 48 8B ?? 83 ?? 0F 0F B6 ?? ?? ??
30 04 ?? 48 FF ?? 49 3B ?? 72 ?? 80 [1-5] 00 75 ?? B? 5B 55 D2 56 [0-8] E8
[4-32] 33 ?? 33 ?? FF D? [0-4] FF D? }
        condition:
                uint16(0) == 0x5a4d and all of them
}
```

Additional YARA Rules

This is a second dropper used to launch PLUSDROP in another TOUGHPROGRESS campaign.

```
rule G_Dropper_TOUGHPROGRESS_XML_1 {
    meta:
        author = "GTIG"
        description = "XML lure file used to launch a PLUSDROP dll."
        md5 = "dccbb41af2fcf78d56ea3de8f3d1a12c"
    strings:
        $str1 = "System.Convert.FromBase64String"
        $str2 = "VirtualAlloc"
        $str3 = ".InteropServices.Marshal.Copy"
        $str4 = ".DllImport"
        $str5 = "kernel32.dll"
        $str6 = "powrprof.dll"
        $str7 = ".Marshal.GetDelegateForFunctionPointer"
    condition:
        uint16(0)!= 0x5A4D and all of them and filesize > 500KB and
filesize < 5MB
}
```

PLUSBED is an additional stage observed in other TOUGHPROGRESS campaigns.

```
rule G_Dropper_PLUSBED_2 {
        meta:
                author = "GTIG"
                date_created = "2025-04-29"
                date_modified = "2025-04-29"
                md5 = "39a46d7f1ef9b9a5e40860cd5f646b9d"
                rev = 1
        strings:
                $api1 = { BA 54 B8 B9 1A }
                $api2 = { BA 78 1F 20 7F }
                $api3 = { BA 62 34 89 5E }
                $api4 = { BA 65 62 10 4B }
                $api5 = { C7 44 24 34 6E 74 64 6C 66 C7 44 24 38 6C 00 FF D0 }
        condition:
                uint16(0) != 0x5A4D and all of them
}
```

Posted in

[Threat Intelligence]