CLAROTY

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 research: Uri Katz, Vera Mens

Pwn2Own Toronto 2022 - IoT

$40,000

$40,000

WD MyCloud Pro PR4100

Synology DSM 920+

https

NAS Cloud Platforms

$40,000

$40,000

WD MyCloud Pro PR4100

Synology DSM 920+

https

NAS Cloud Platforms

My Cloud OS 5

Back Up. Access. Collaborate.

Synology®

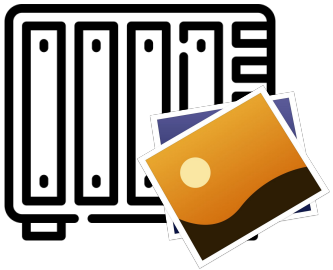
QuickConnect

Anywhere

Home

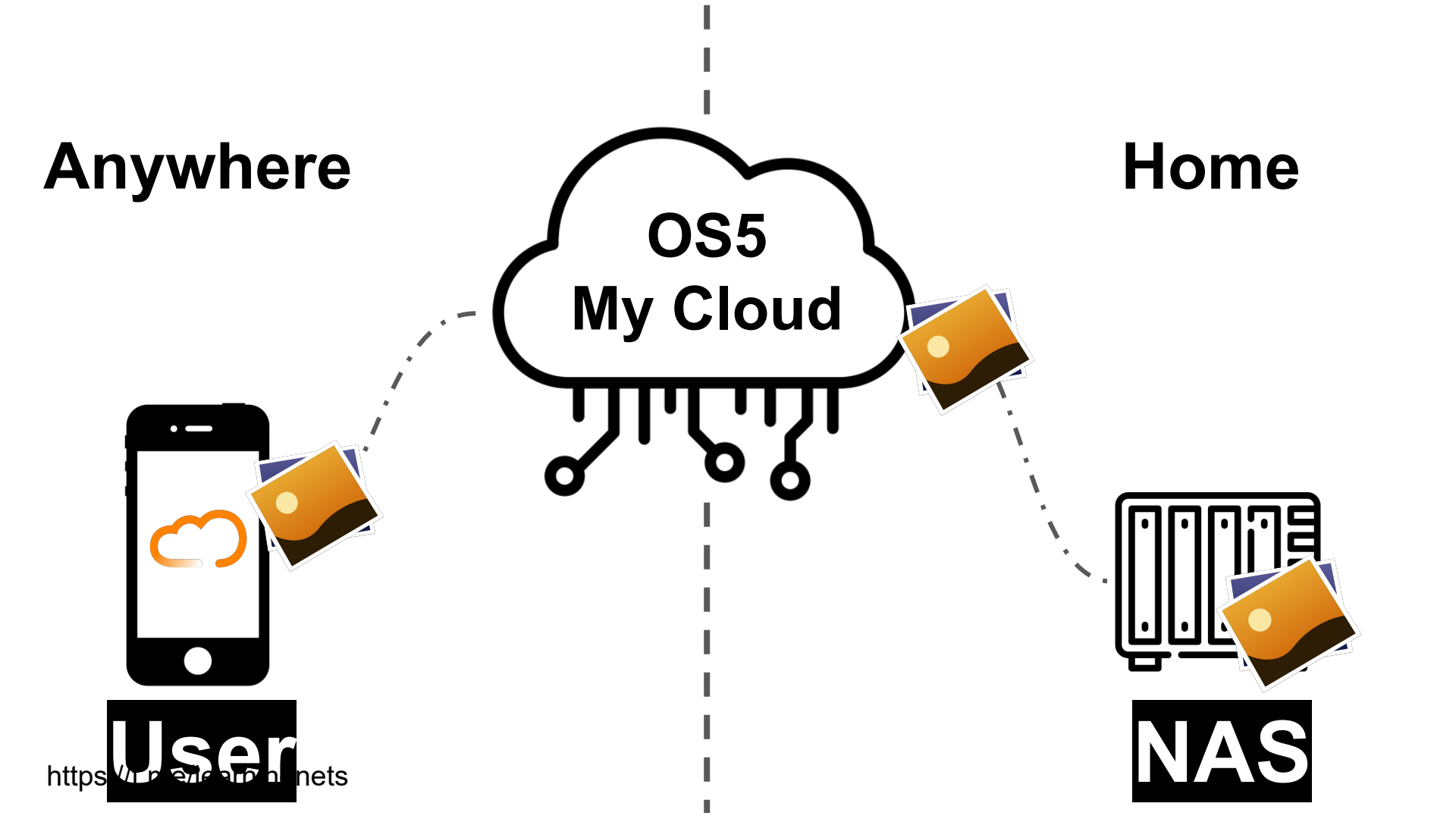
OS5
My Cloud

User



NAS

https://[illegible].nets

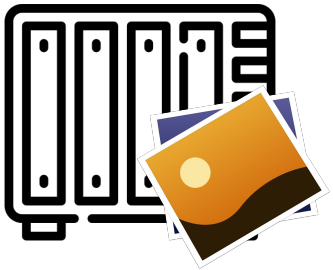

Anywhere
Home
OS5
My Cloud
User
NAS

Anywhere

Home

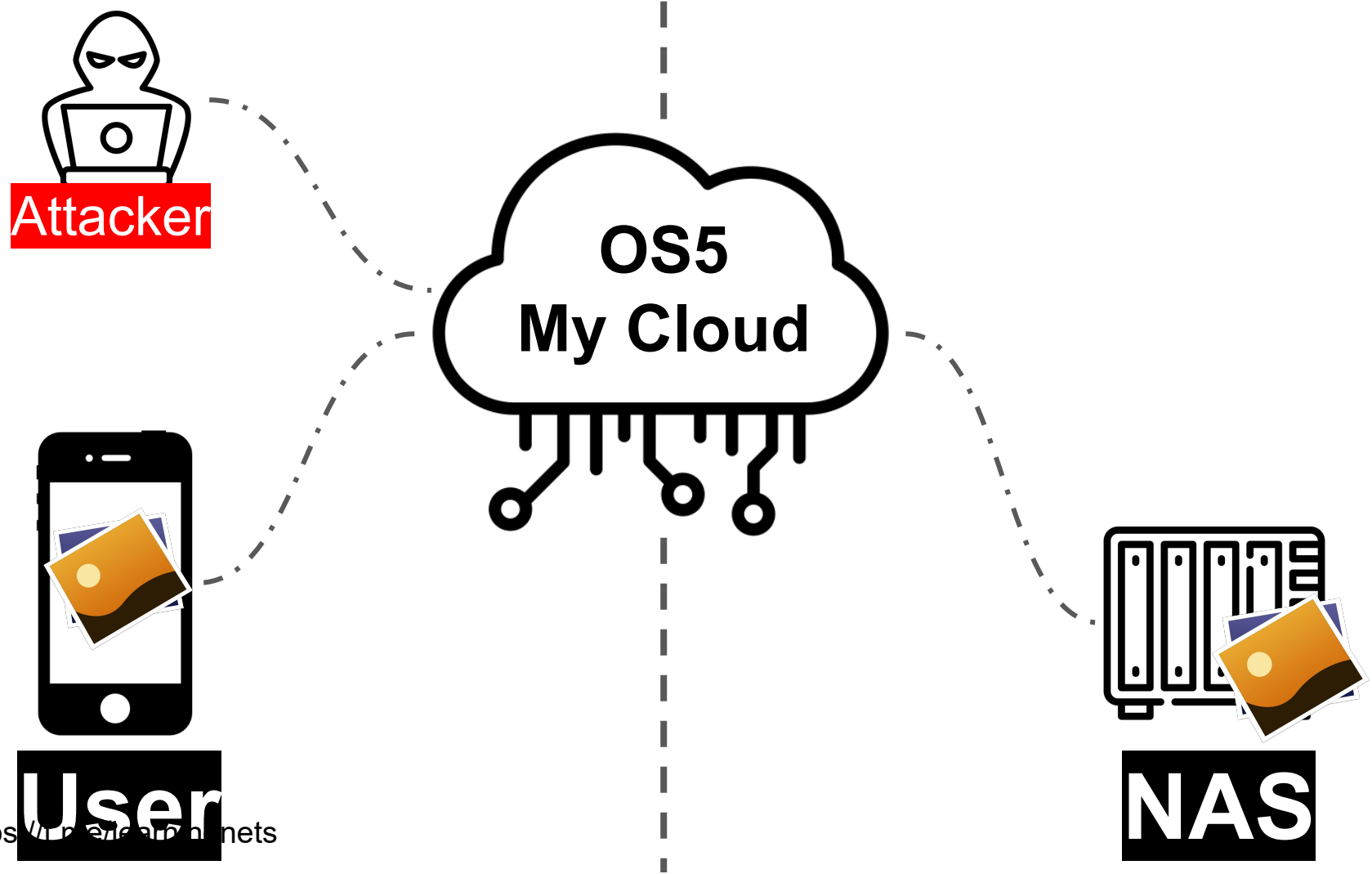
OS5
My Cloud

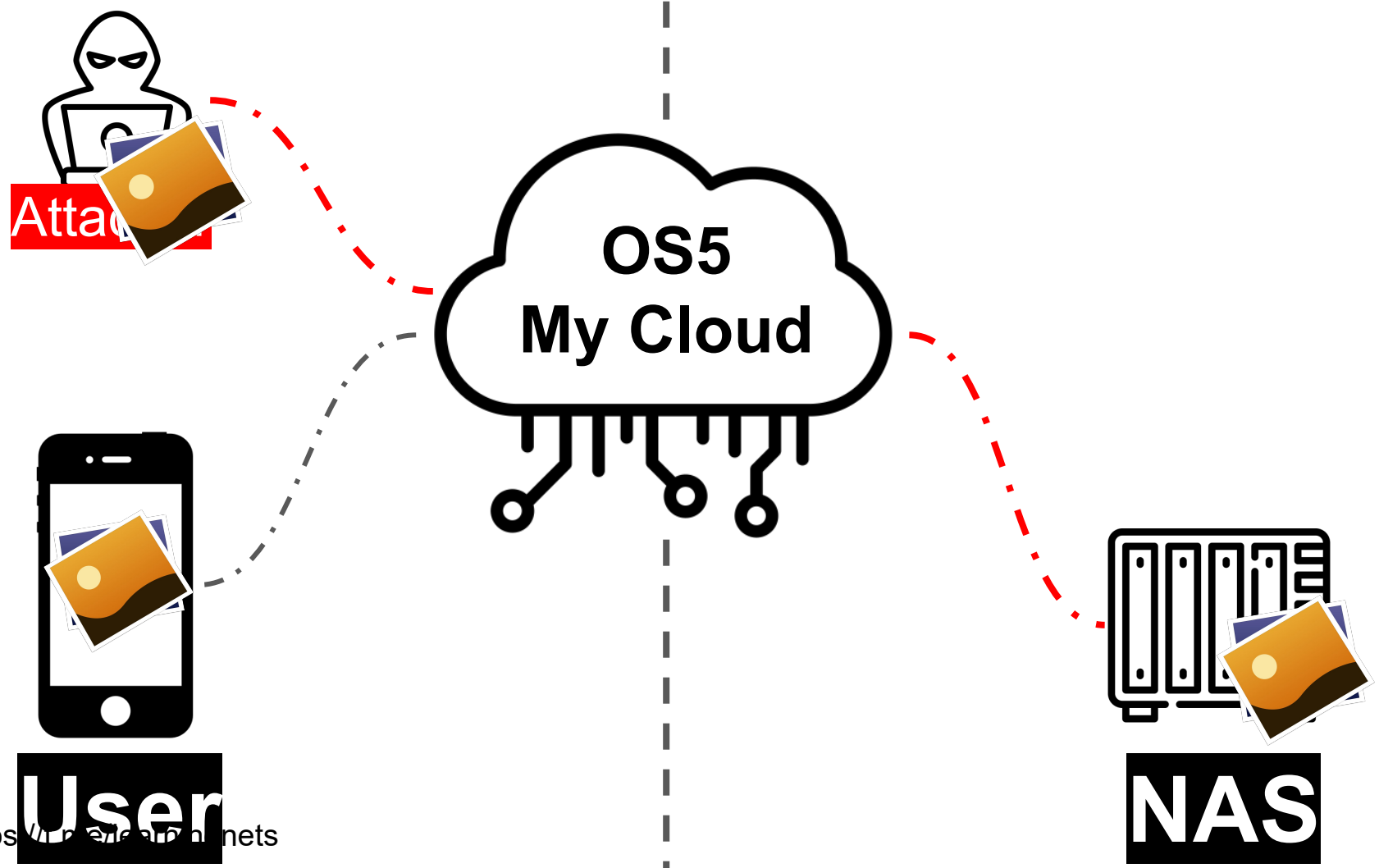
User

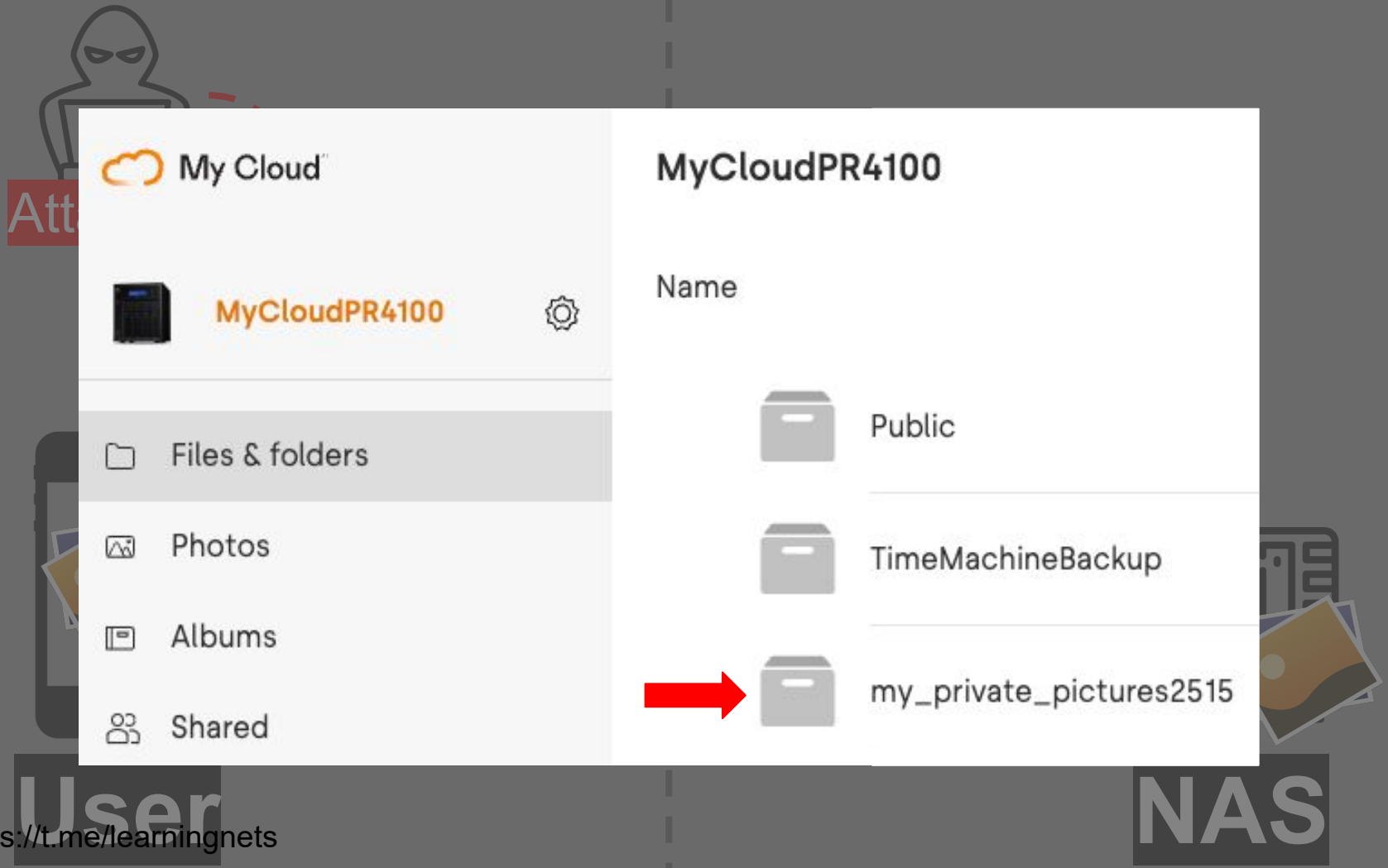


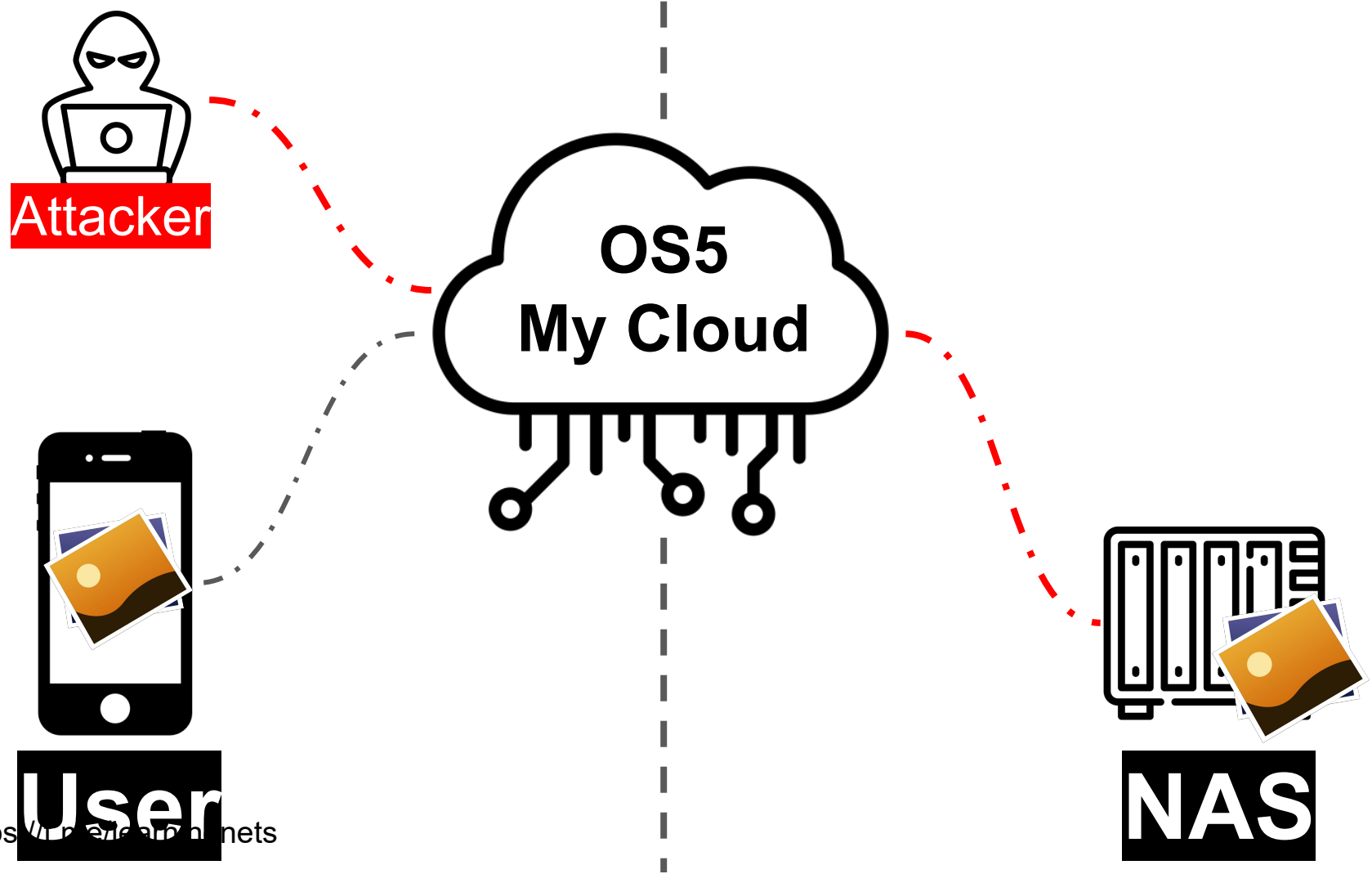
NAS

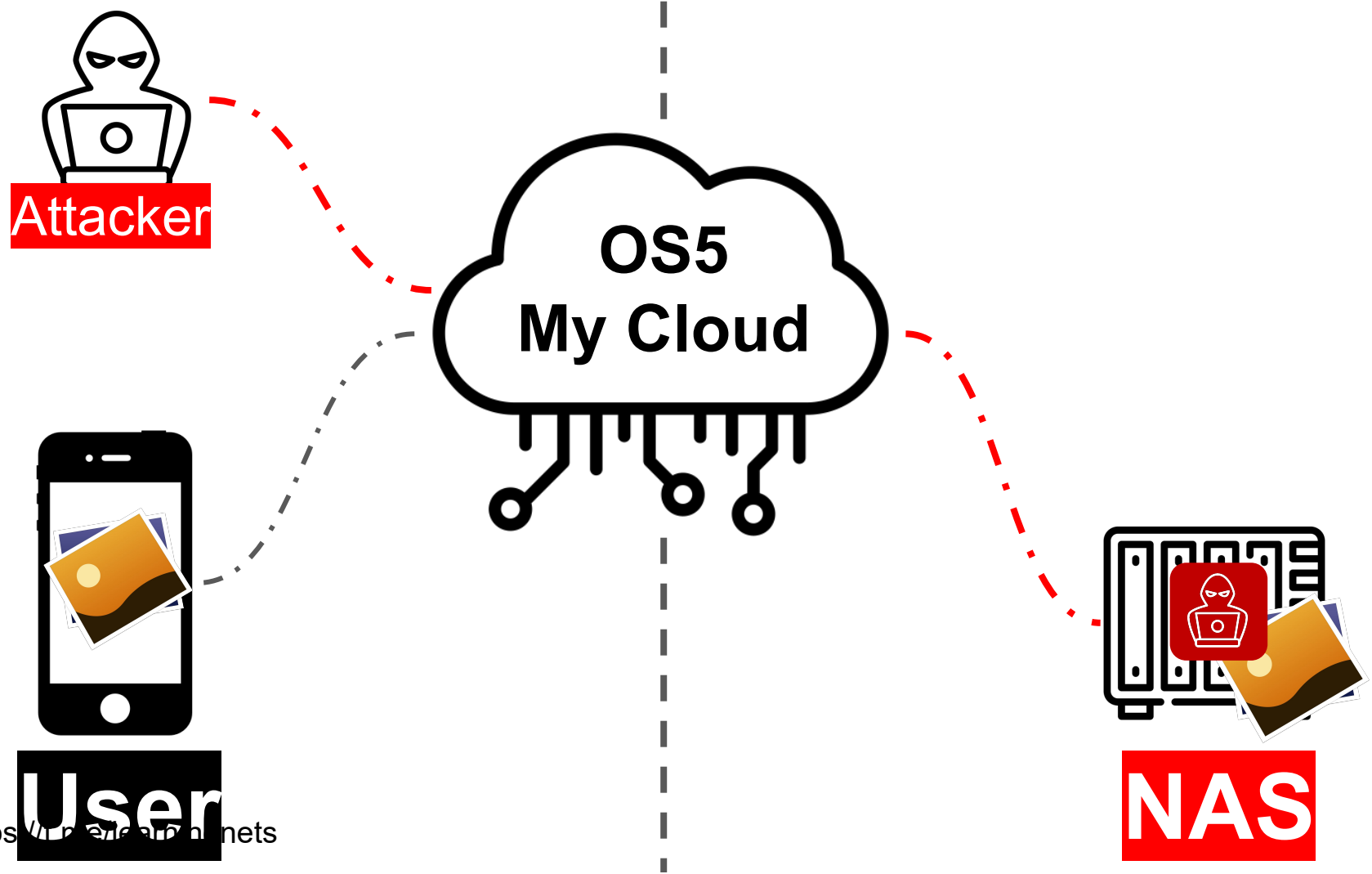
https://pixabay.com/vectors/nets


Attacker
OS5
My Cloud
User
NAS


OS5
My Cloud
User
NAS



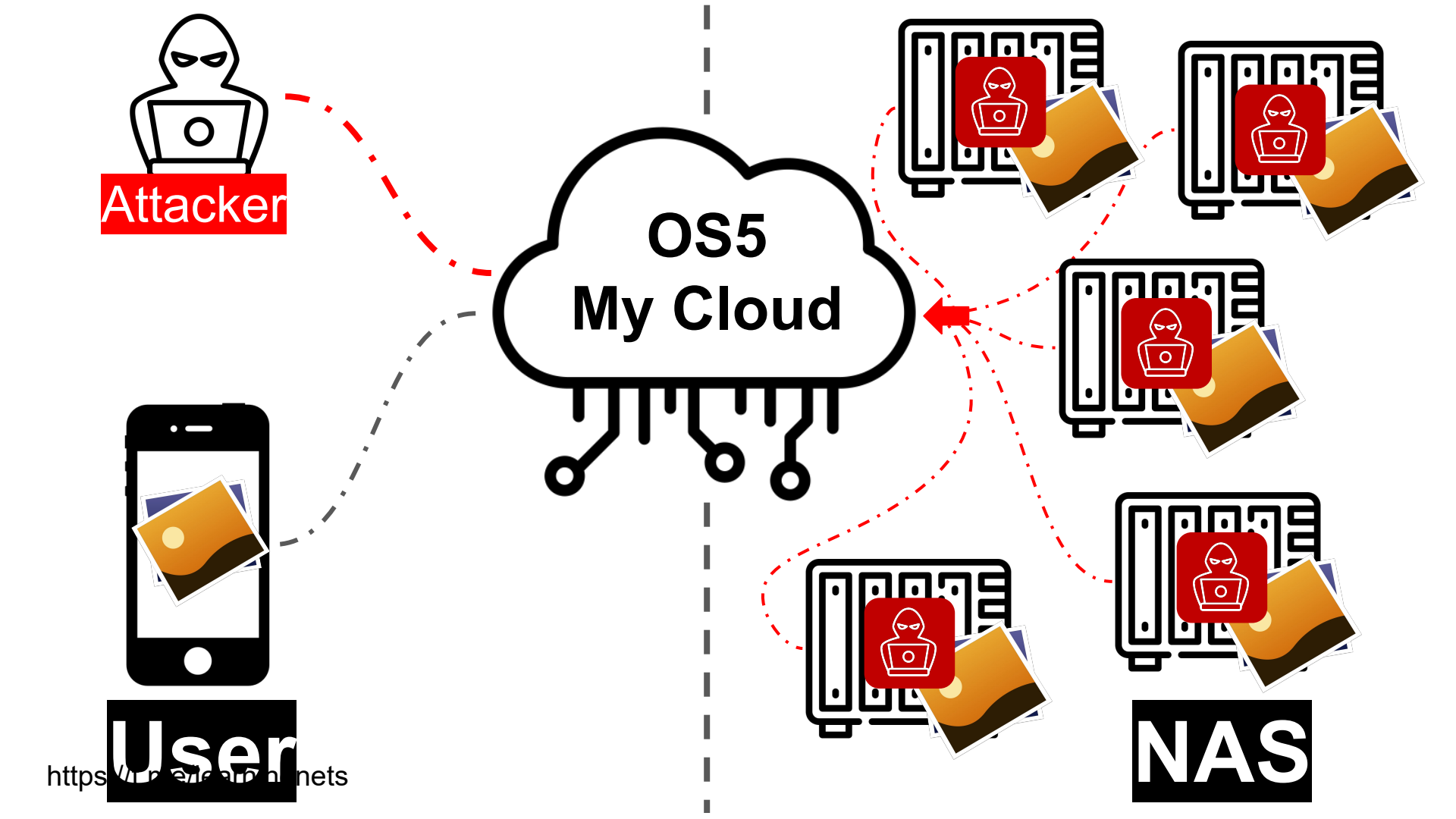

Attacker
OS5
My Cloud
User
NAS


Attacker
OS5
My Cloud
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


Attacker
OS5
My Cloud
User
NAS

Western Digital.

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: 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 - Firmware
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

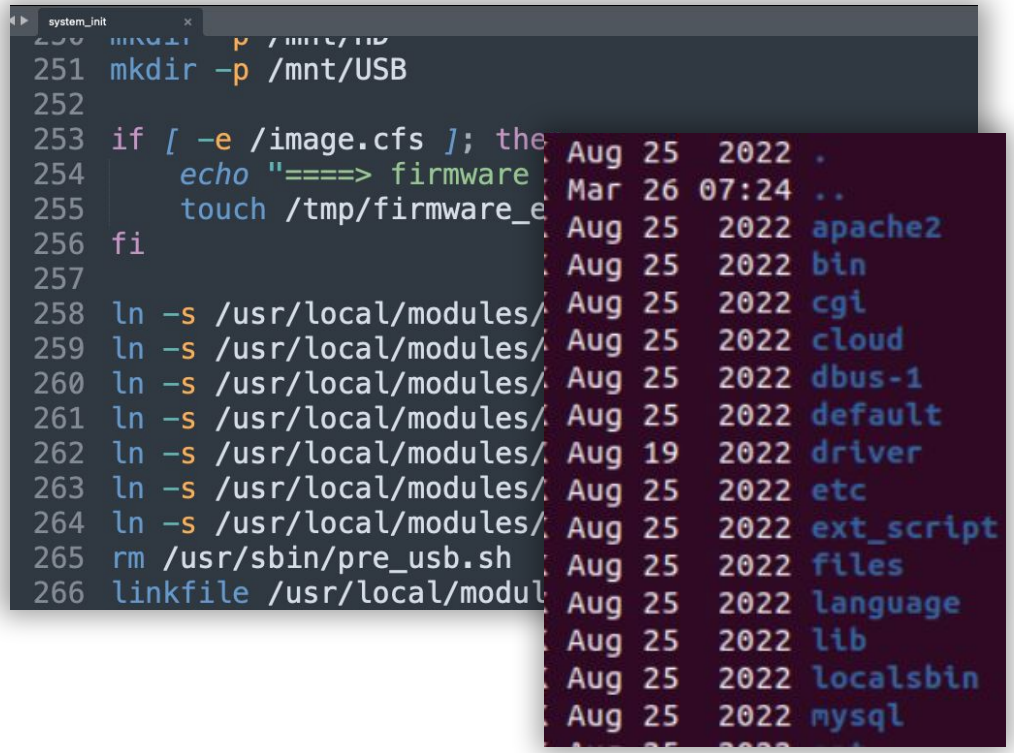
Firmware not encrypted, easy to extract the filesystem using `binwalk`

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

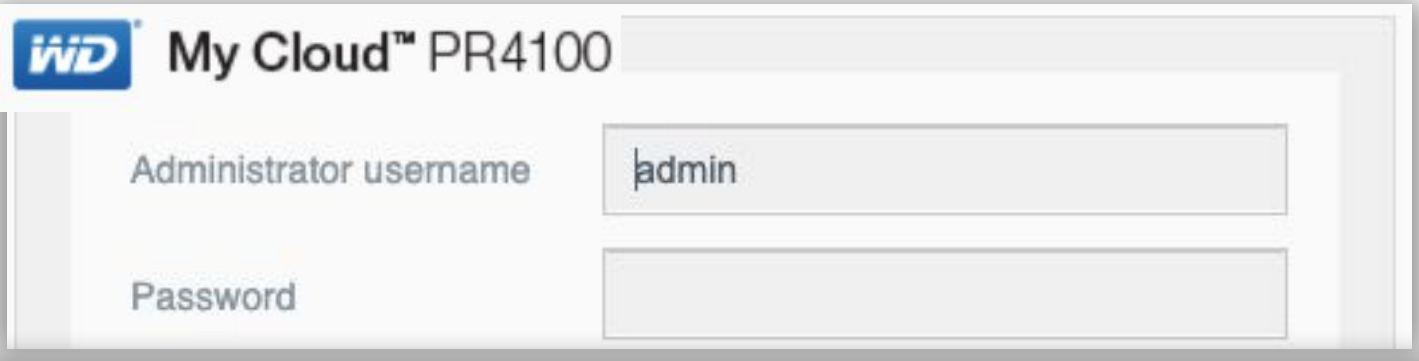
```
parallels@parallels-Parallels-Virtual-Platform:~/tests$ binwalk WDMyCloudPR4100_5.24.108_prod.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
128           0x80            uImage header, header size: 64 bytes, header CRC: 0x97DA6B4A, created: 2022-08-
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-
type: gzip, image name: "Initramfs"
4854800       0x4A1410        gzip compressed data, maximum compression, from Unix, last modified: 2022-08-26
8629155       0x83ABA3        Squashfs filesystem, little endian, version 4.0, compression:xz, size: 28093768
28950959      0x1147EAF       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`**

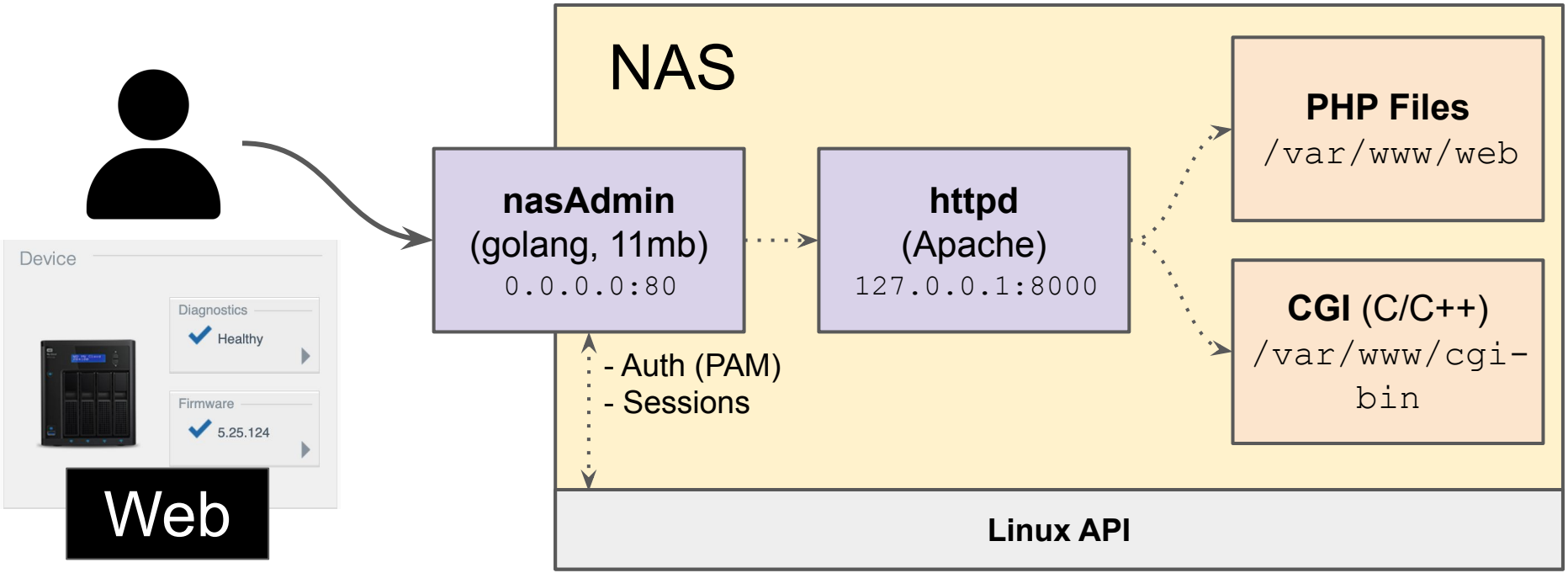


Emulating WD Device




```
root[redacted] ~ # ps aux | grep httpd
 2967 root      36344 S    httpd -f /usr/local/apache2/conf/httpd.con
 3445 root      36432 S    httpd -f /usr/local/apache2/conf/httpd.con
 3446 root      1979m S    httpd -f /usr/local/apache2/conf/httpd.con
 5832 root       4704 S    grep httpd
root[redacted] ~ # 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

NAS

nasAdmin
(golang, 11mb)
0.0.0.0:80

- Auth (PAM)
- Sessions

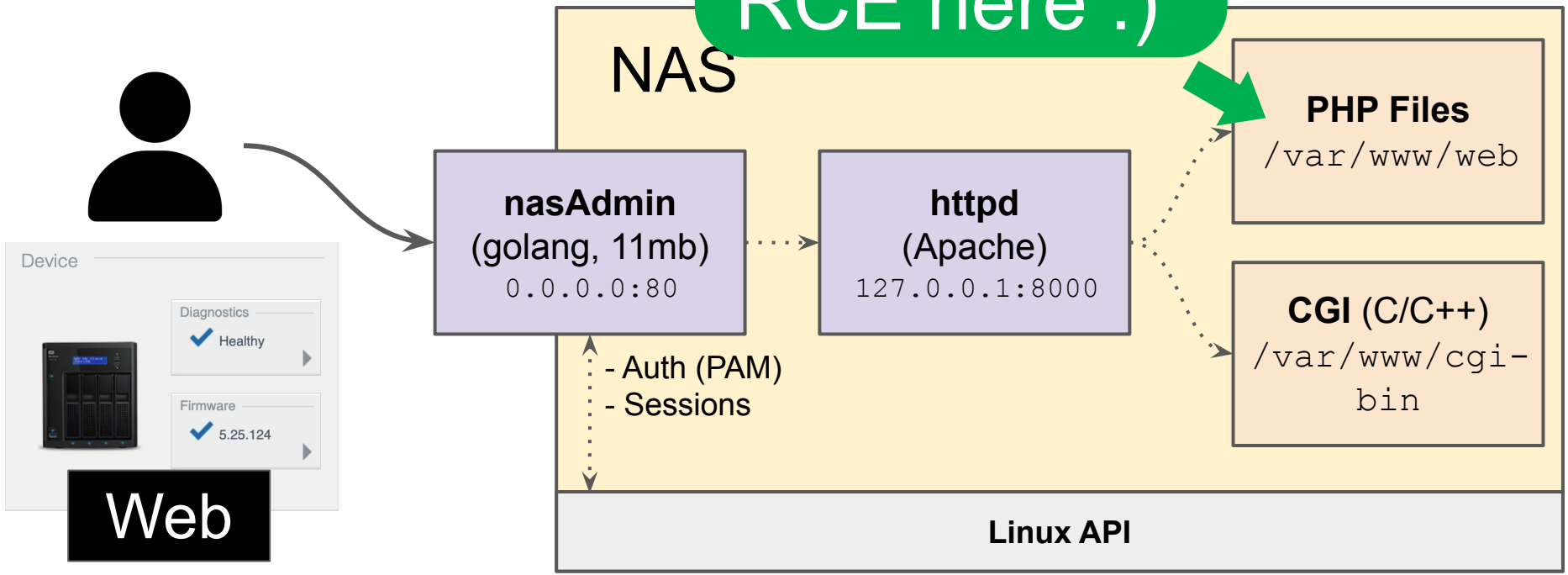
httpd
(Apache)
127.0.0.1:8000

PHP Files
/var/www/web

CGI (C/C++)
/var/www/cgi-bin

Linux API

Web Management Archi

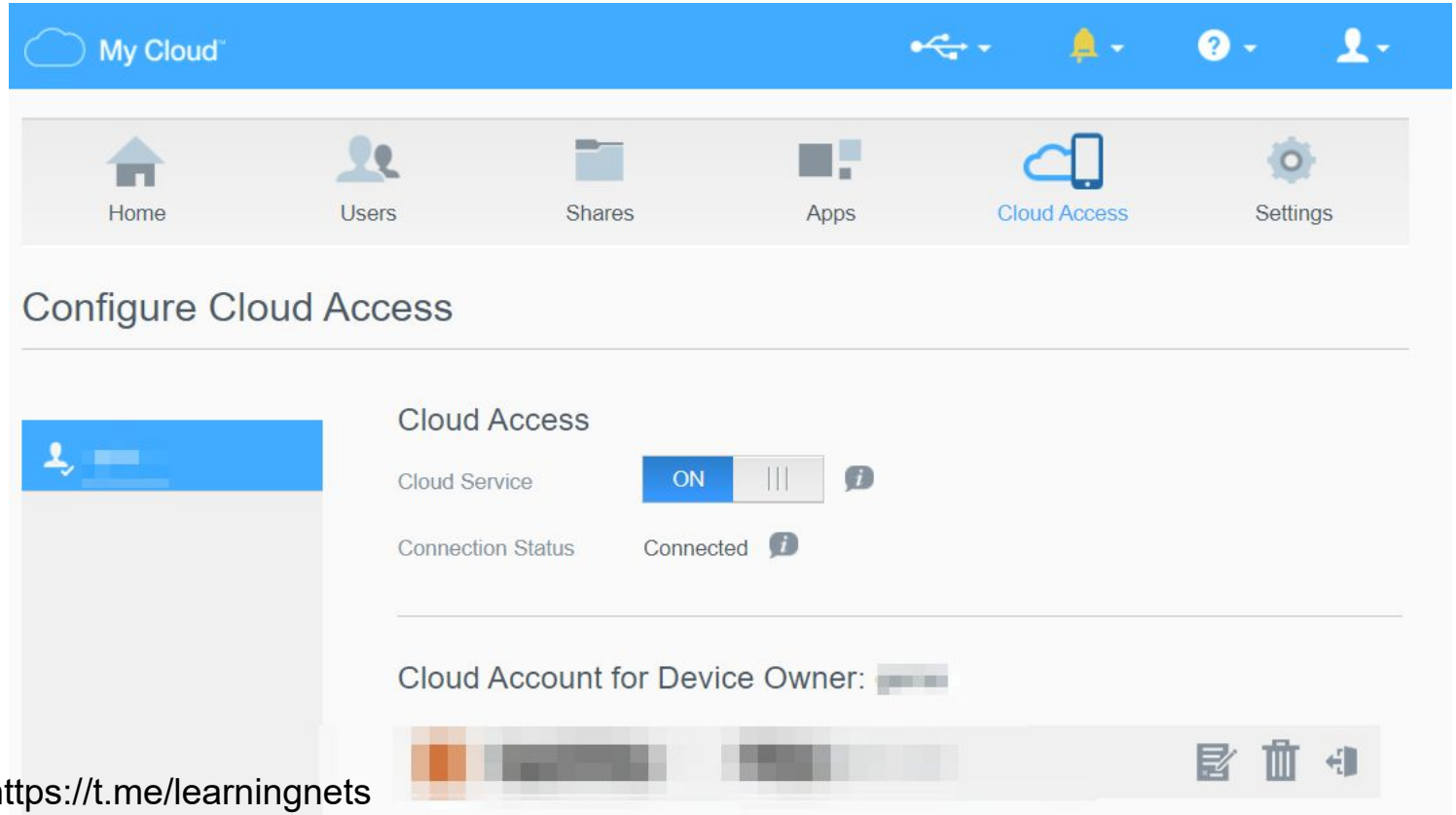


Post Auth OS Command Injection in

```
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'));
```

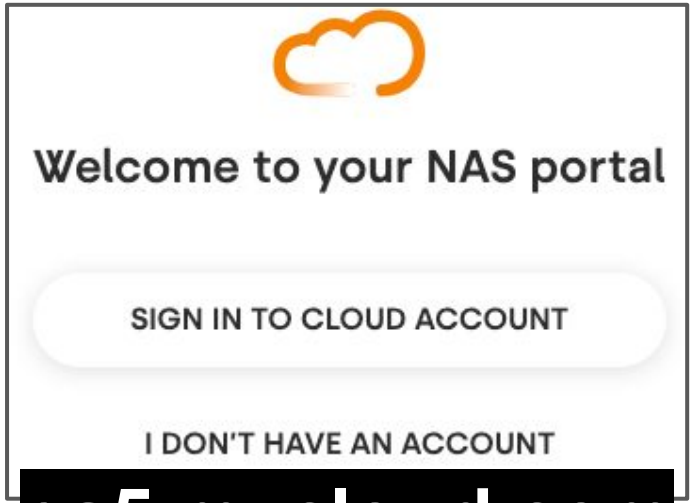
injection point

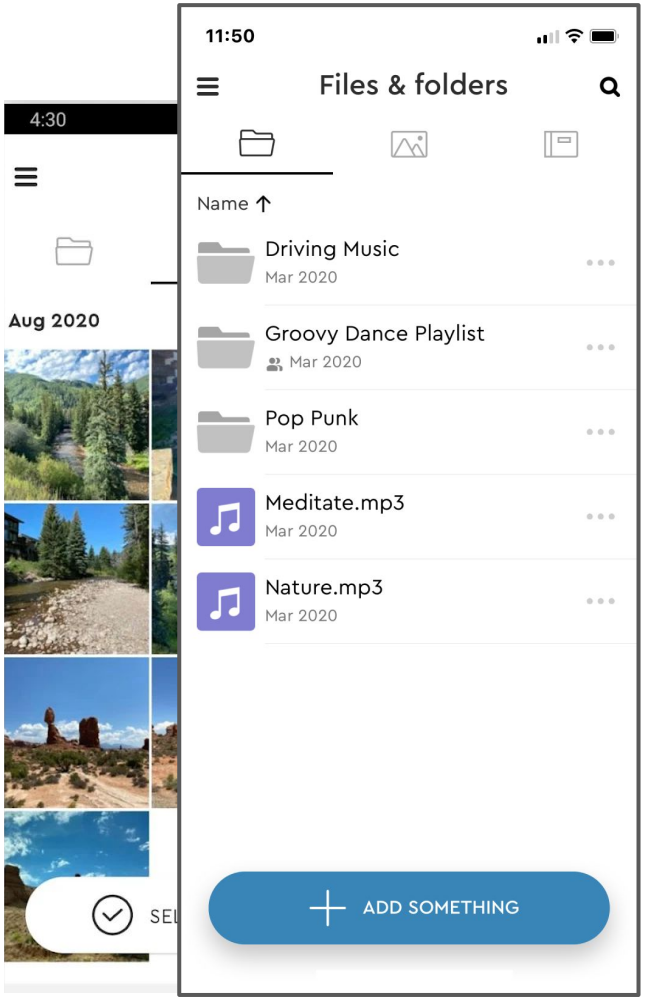
Let’s Explore OS 5 My Cloud!



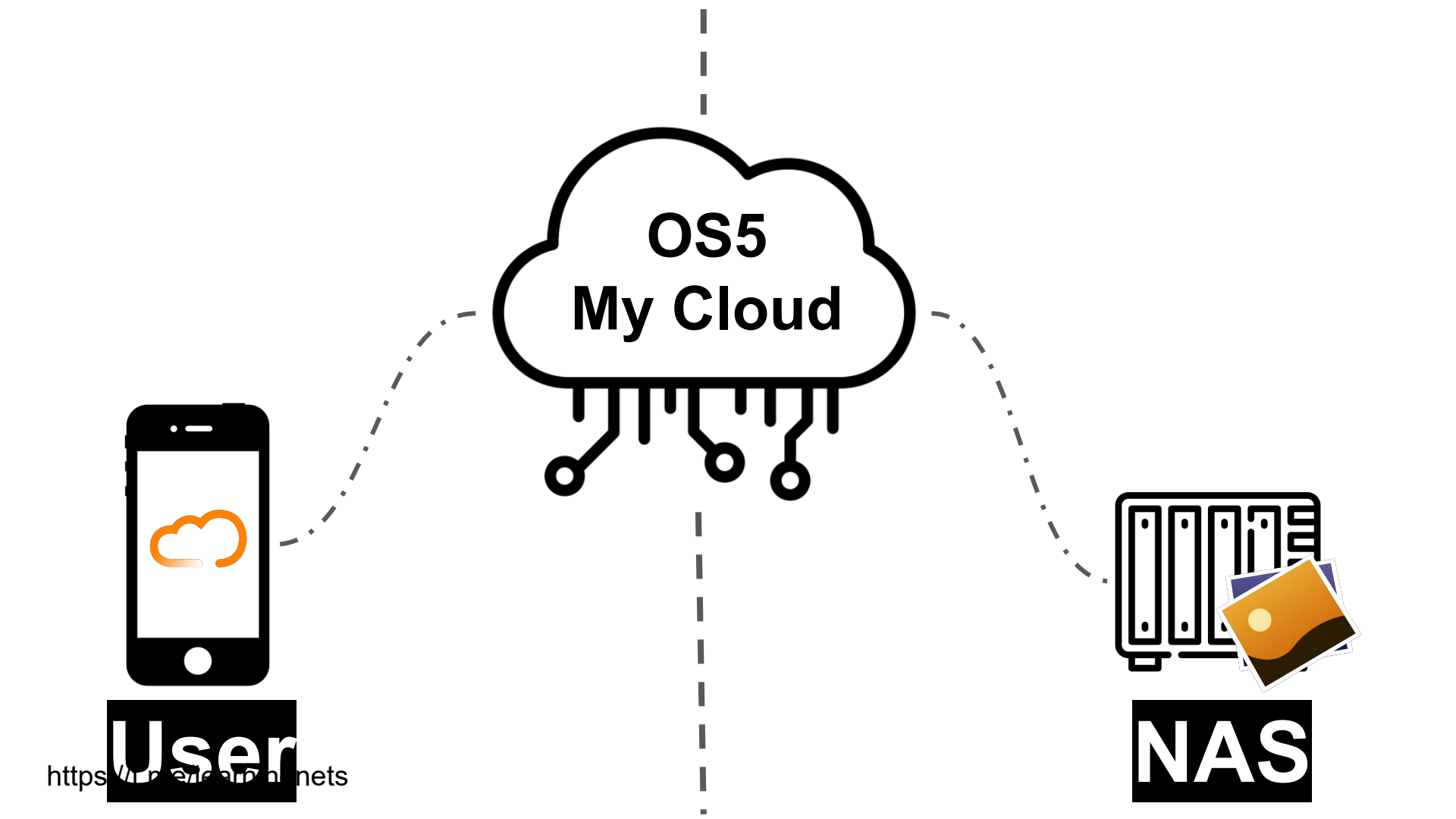
OS 5 My Cloud

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



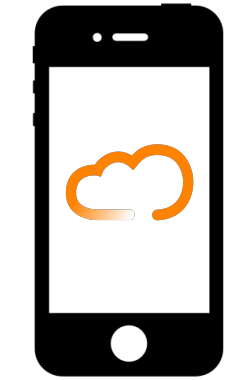


os5.mycloud.com


OS5
My Cloud
User
NAS

WD Cloud Account

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



User

OS5
My Cloud

https://[illegible].nets

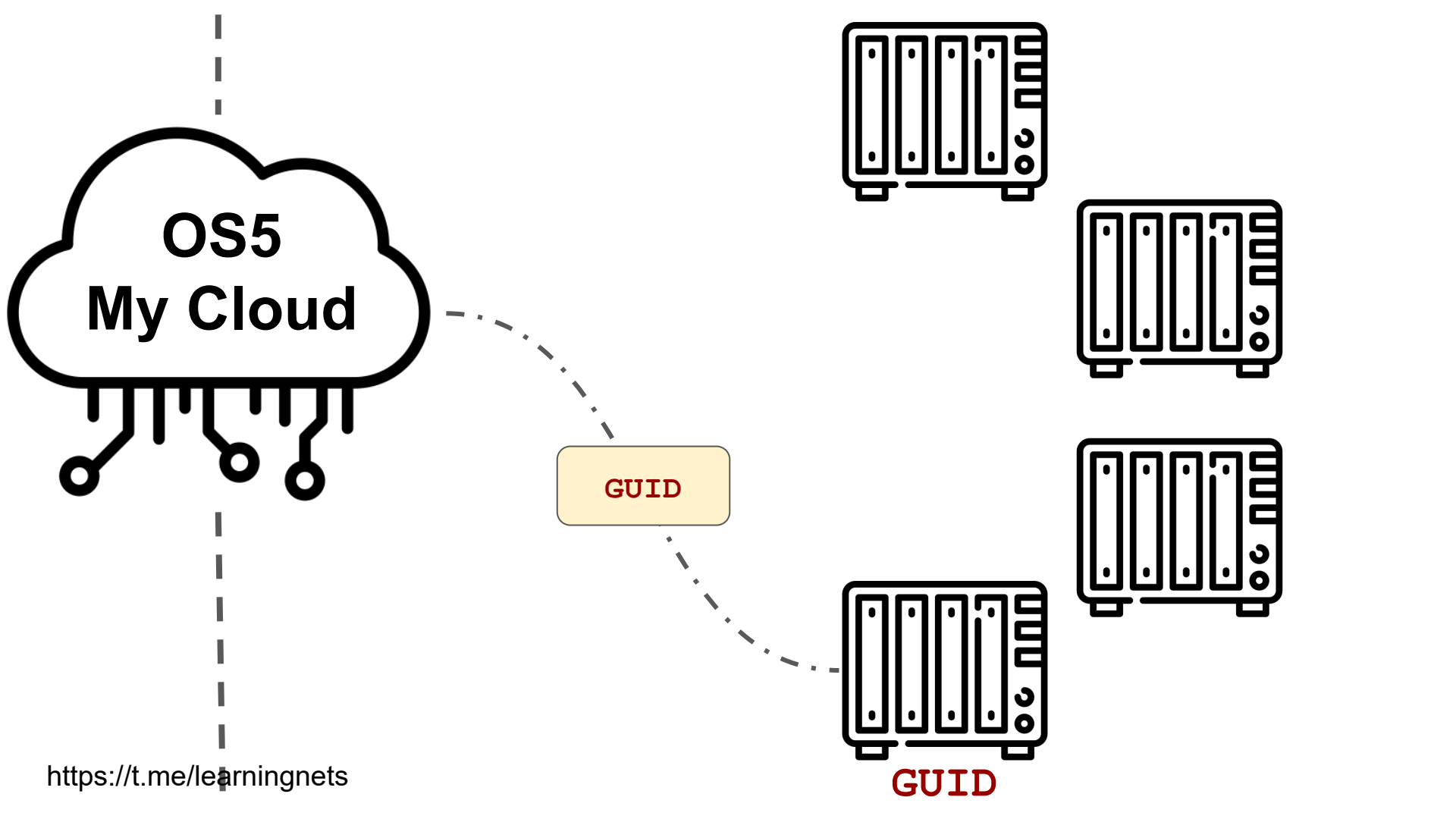
https://prod-940b1d9bbad9901.wdckeystone.com/
6eff0470-804a-4d12-9e3d-88f090de36f0/sdk/v1/volumes

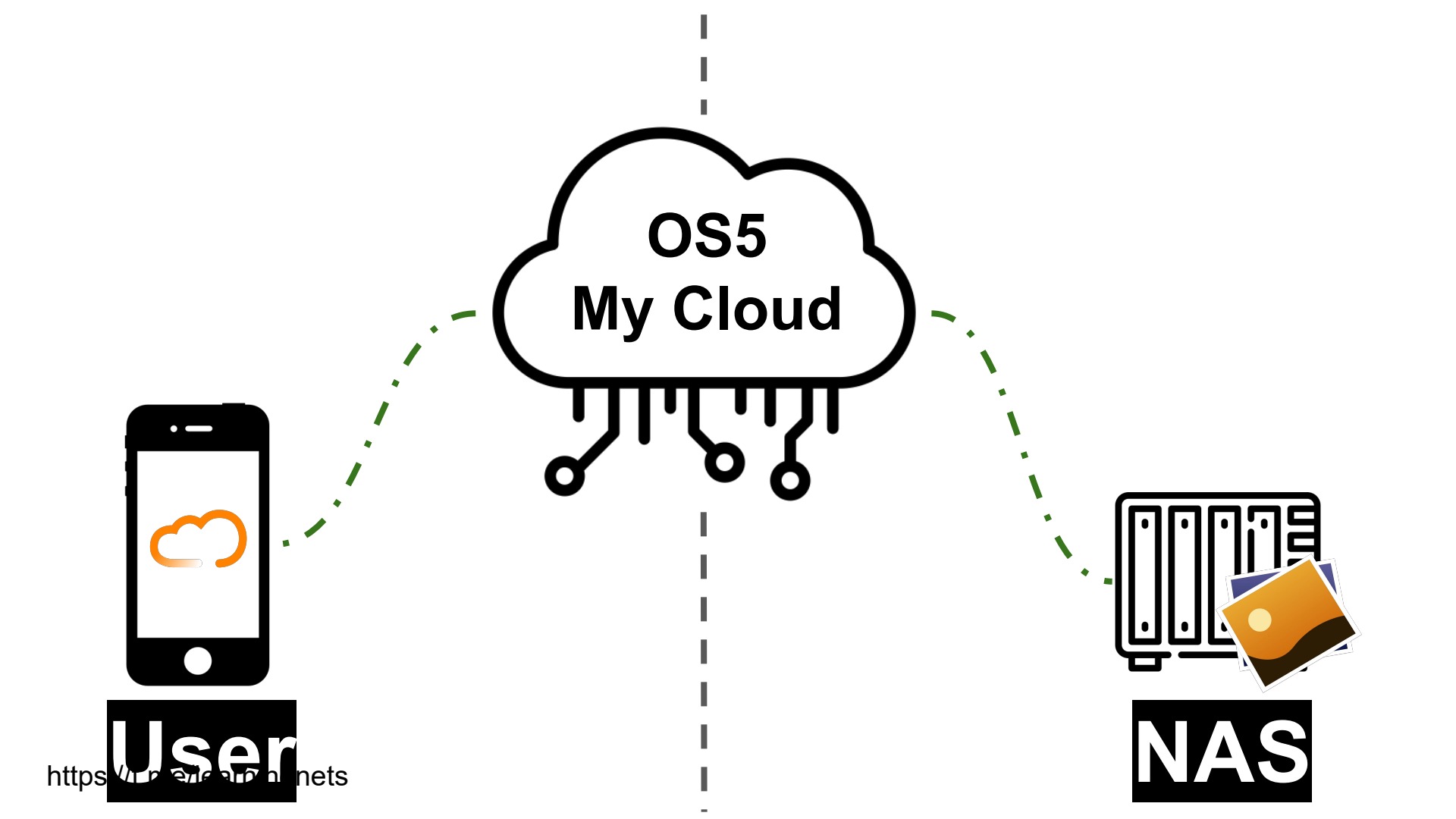
User

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

OS5
My Cloud

https://[illegible]nets



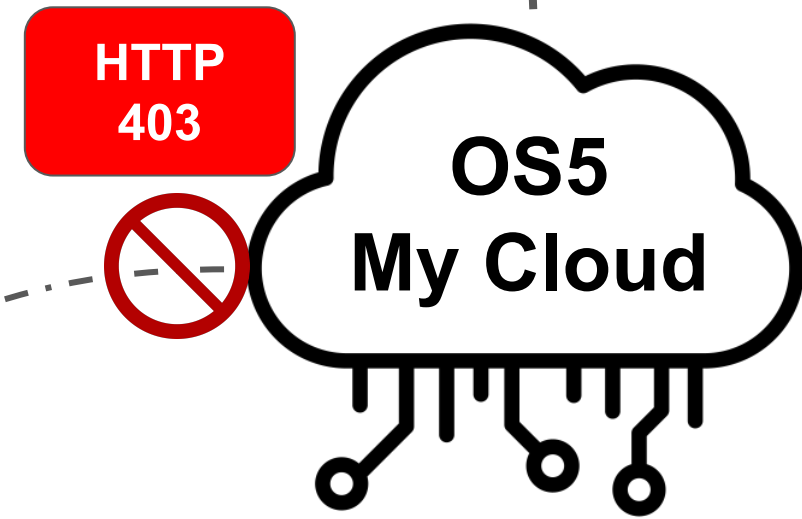

OS5
My Cloud
User
NAS

The Obvious Question

GET

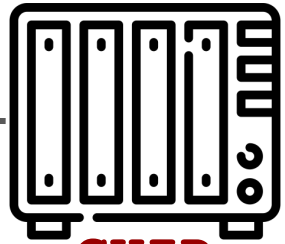
`https://prod.wdckeystone.com/GUID/sdk/v1/volumes`

Trying to access random device without authentication - Blocked



BUT!

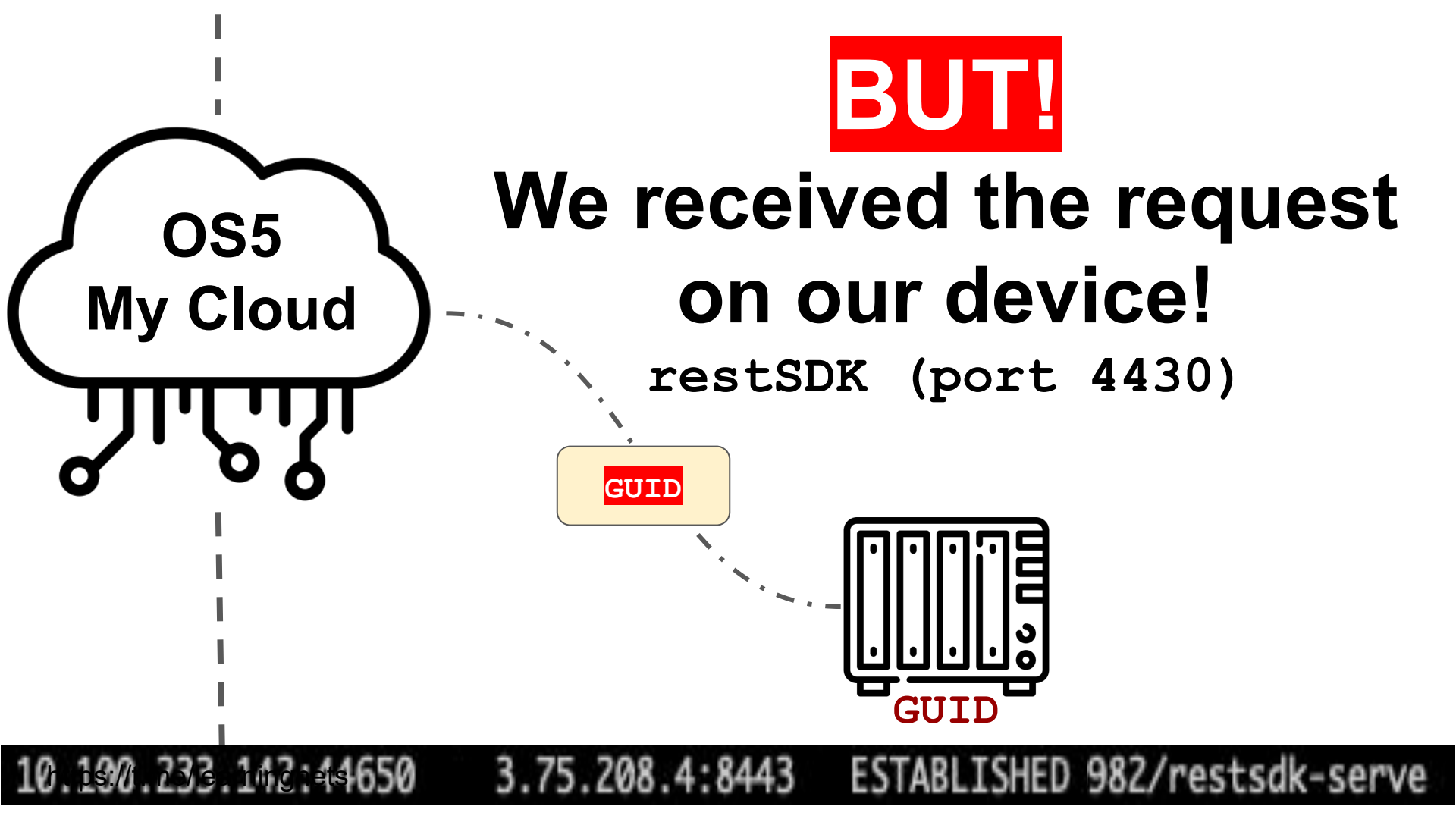
GUID



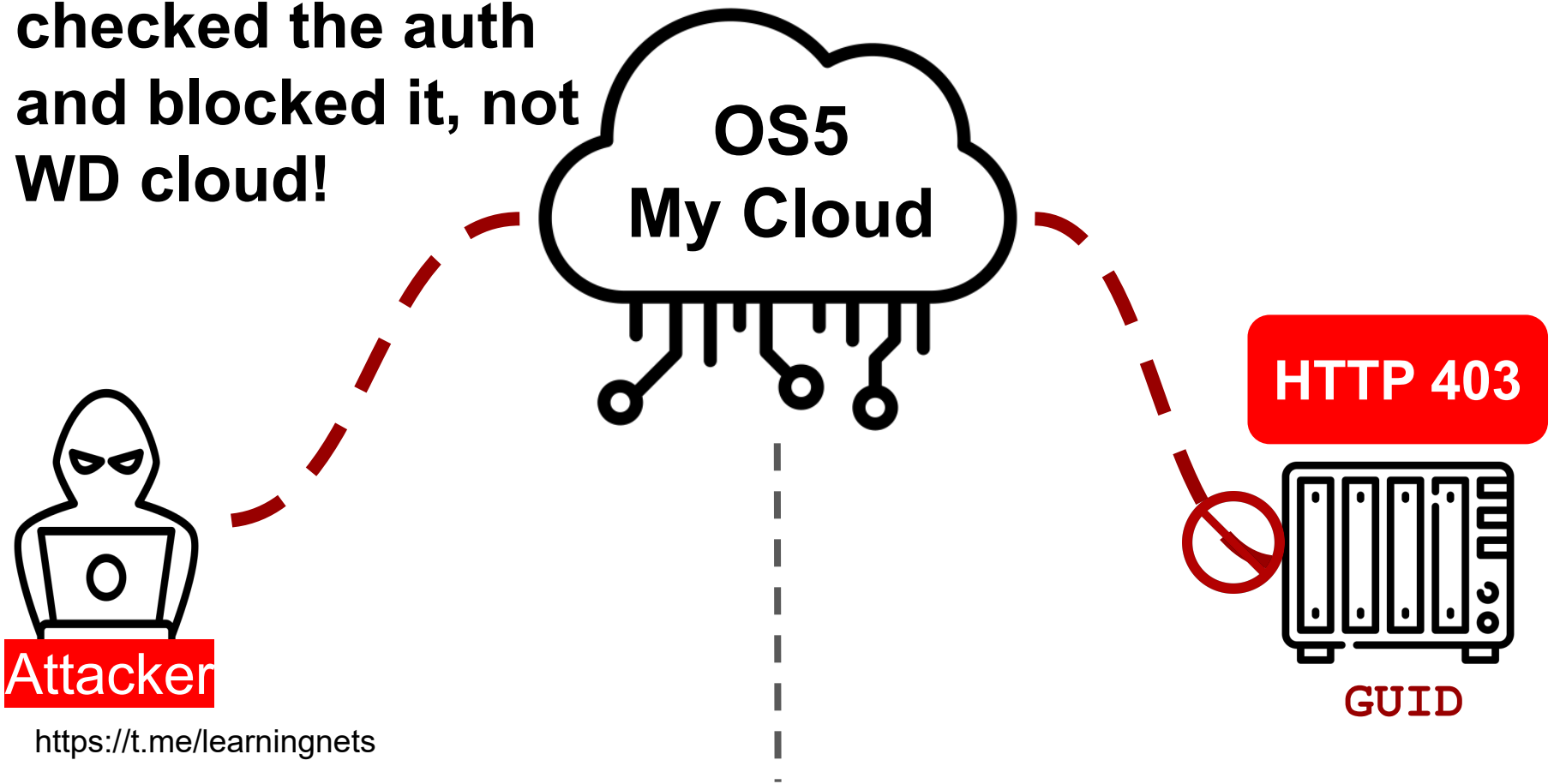
GUID

10.100.233.143:44650 3.75.208.4:8443 ESTABLISHED 982/restsdk-serve

https://t.co/sedringcets


BUT!
We received the request on our device!
restSDK (port 4430)
OS5
My Cloud
GUID
GUID
10.100.233.143:44650 3.75.208.4:8443 ESTABLISHED 982/restsdk-serve
https://t.me/s3curingnets

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



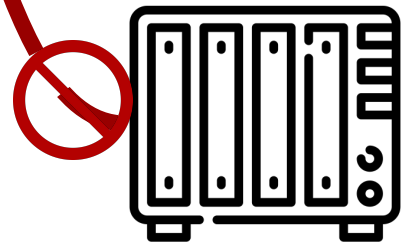
The end device checked the auth and blocked it, not WD cloud!

OS5
My Cloud

This is home (NAT bypass?)

HTTP 403

Attacker



GUID

Our Plan to Exploit All Devices

We “just” need to:

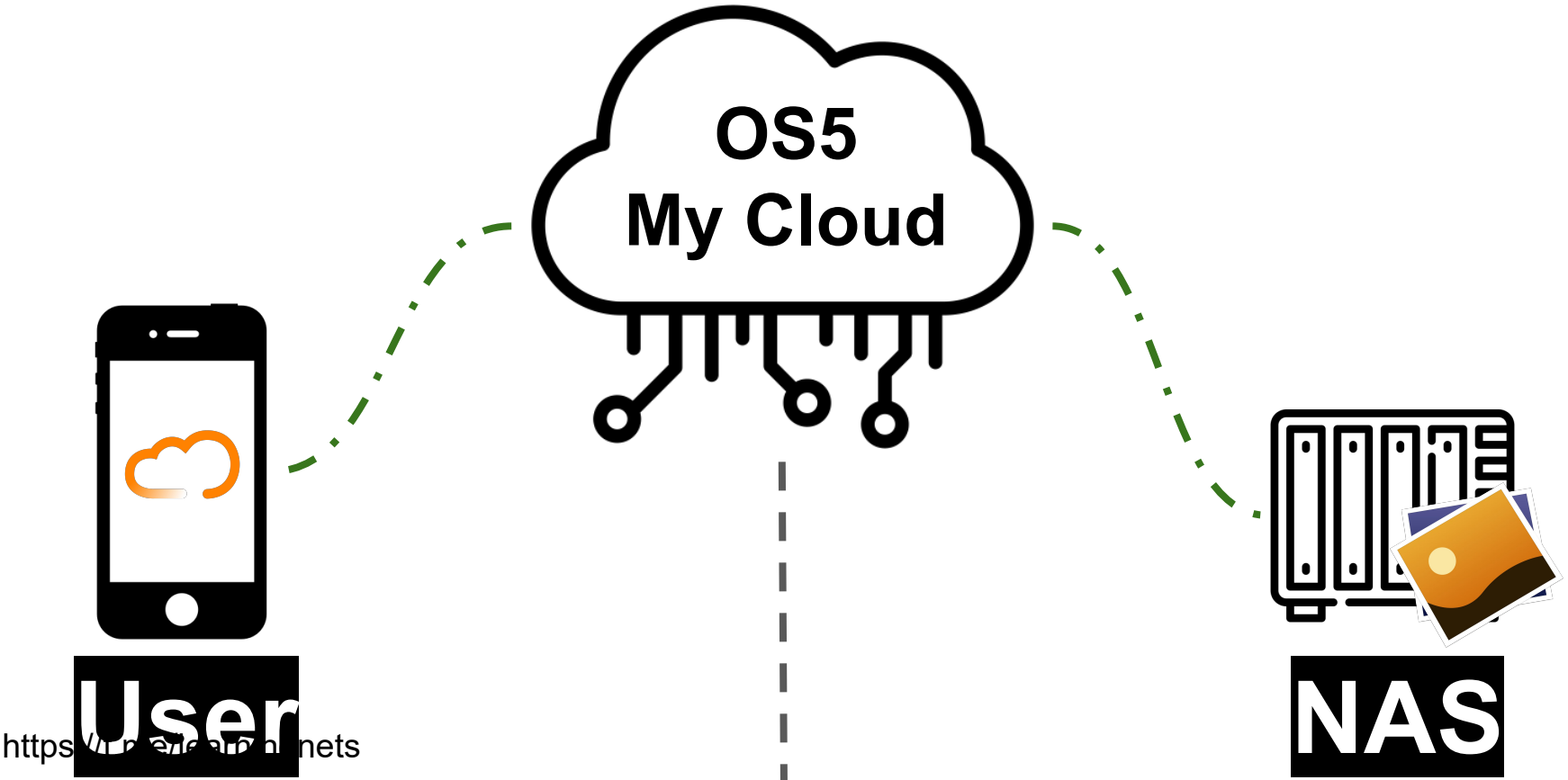
- **[?]** Break 128 bit random GUID
- **[?]** Find auth bypass
- **[?]** Find RCE

Our Plan to Exploit All Devices

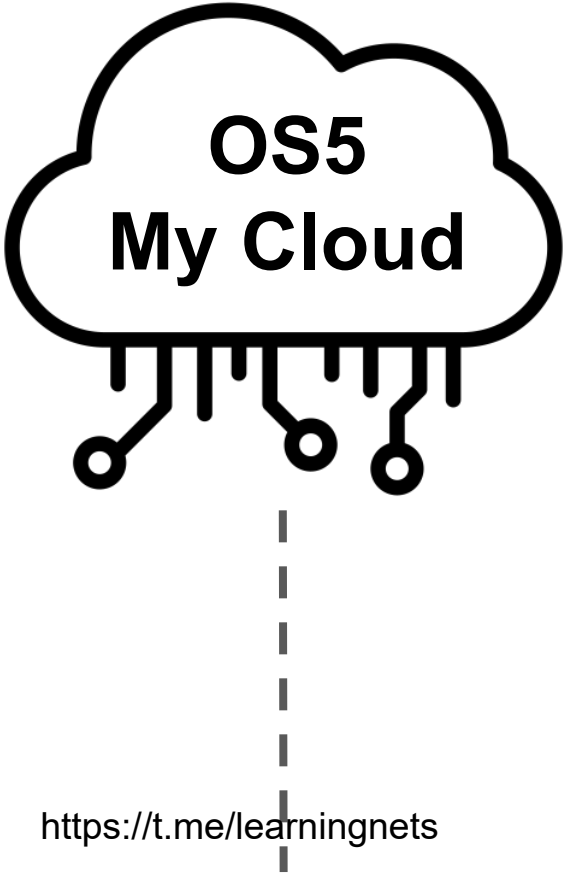
We “just” need to:

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

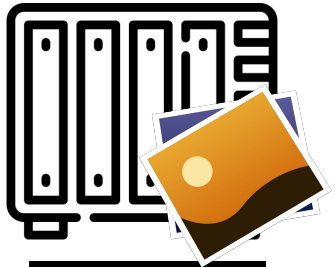
Accessing Externally



Accessing From Home (LAN)

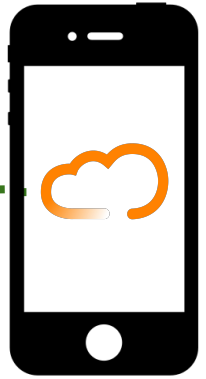


User



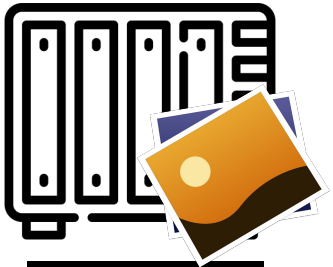
NAS

Accessing From Home (LAN) - Redirect



User

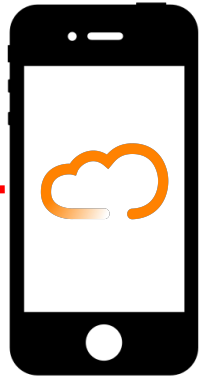
Watch
10Gb
movie



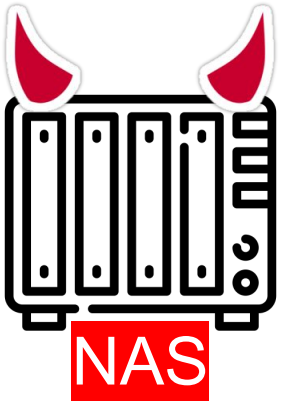
NAS

much more practical to access directly via LAN

But is it Really Our Device?



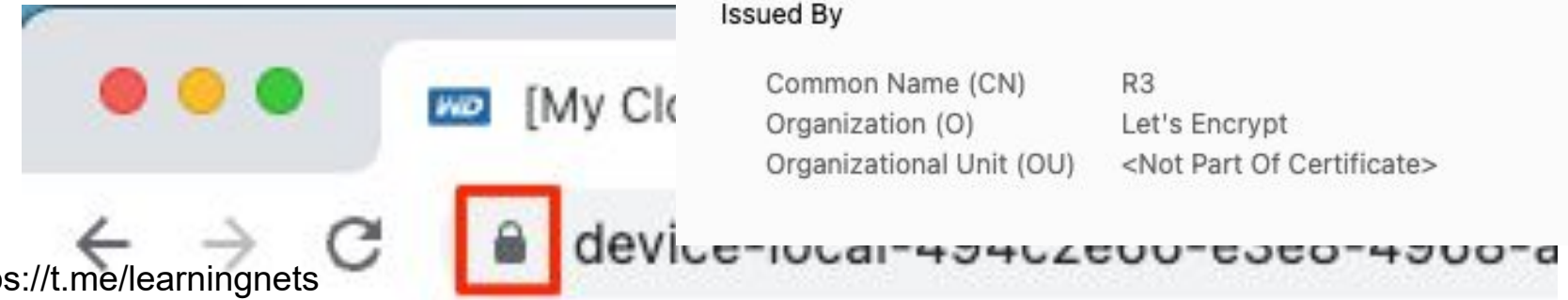
User

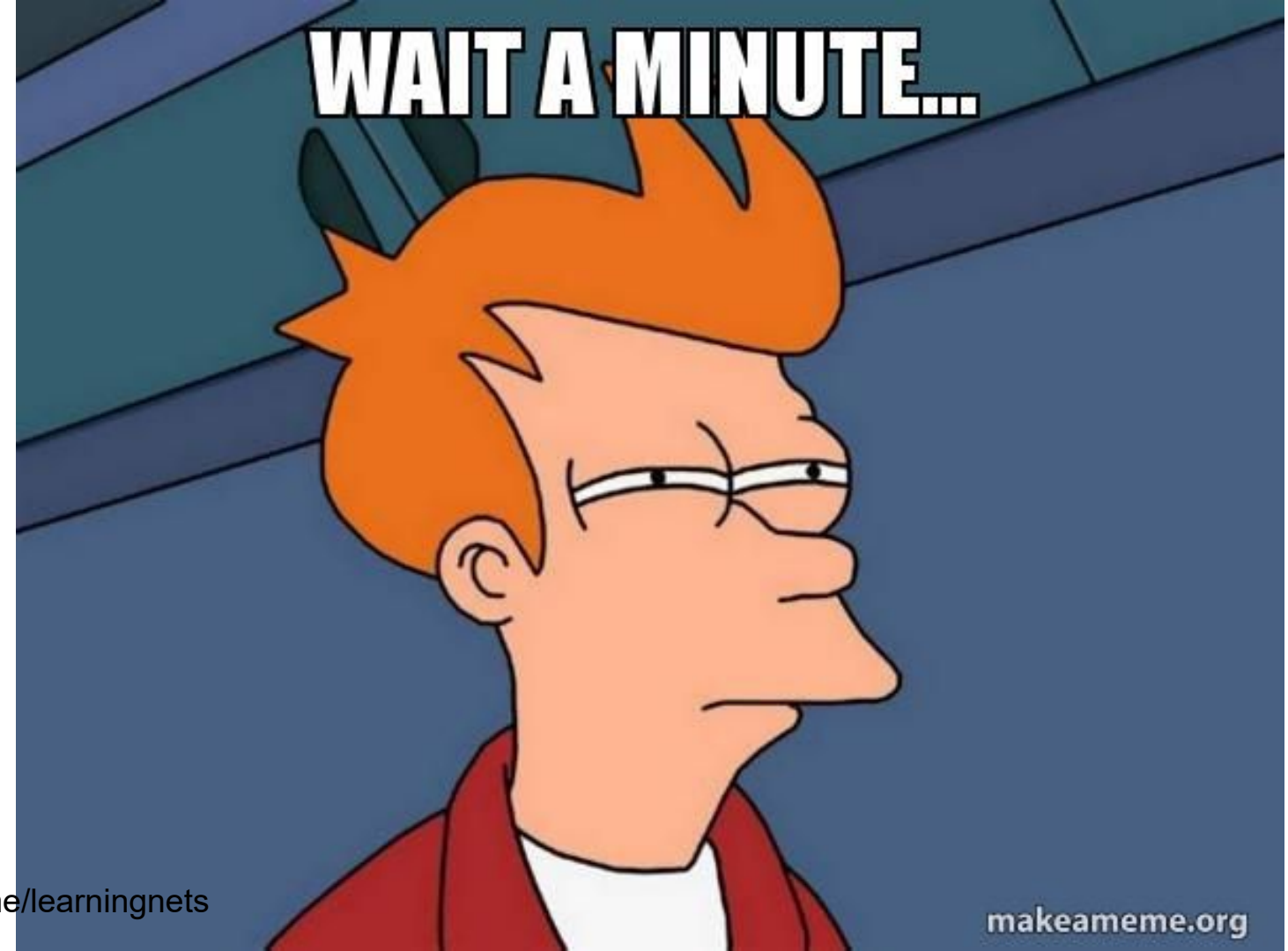


NAS

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

Our device GUID

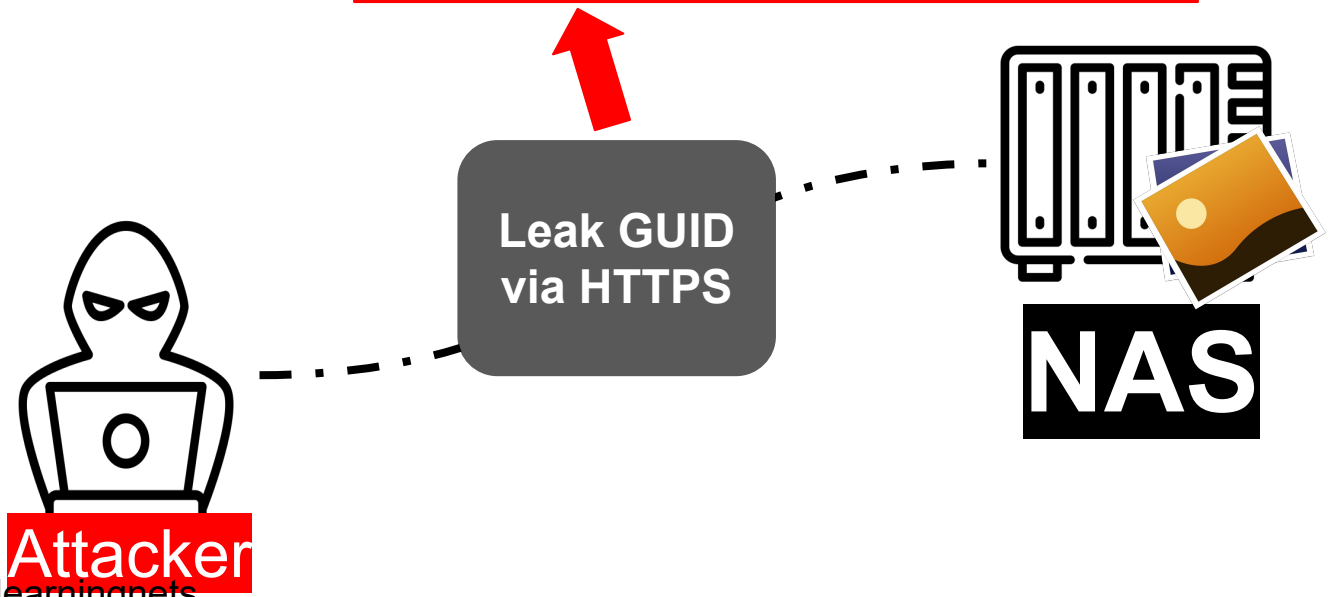
Issued To

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>

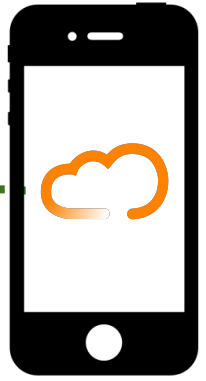
```
→  Downloads openssl s_client -showcerts -connect 169.254.142.221:8543 |  op
depth=2 C = US, O = Internet Security Research Group, CN = ISRG Root X1
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = R3
verify return:1
depth=0 CN = device-local-494c2e66-e3e8-4968-aa04-7ab1cdfcade7.remotewd.com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



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?

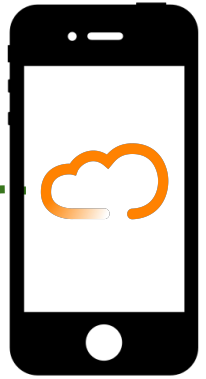


User

???

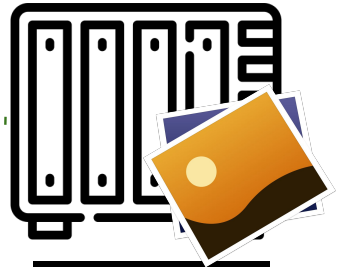
NAS

Leaking GUIDs via LAN is not Enough



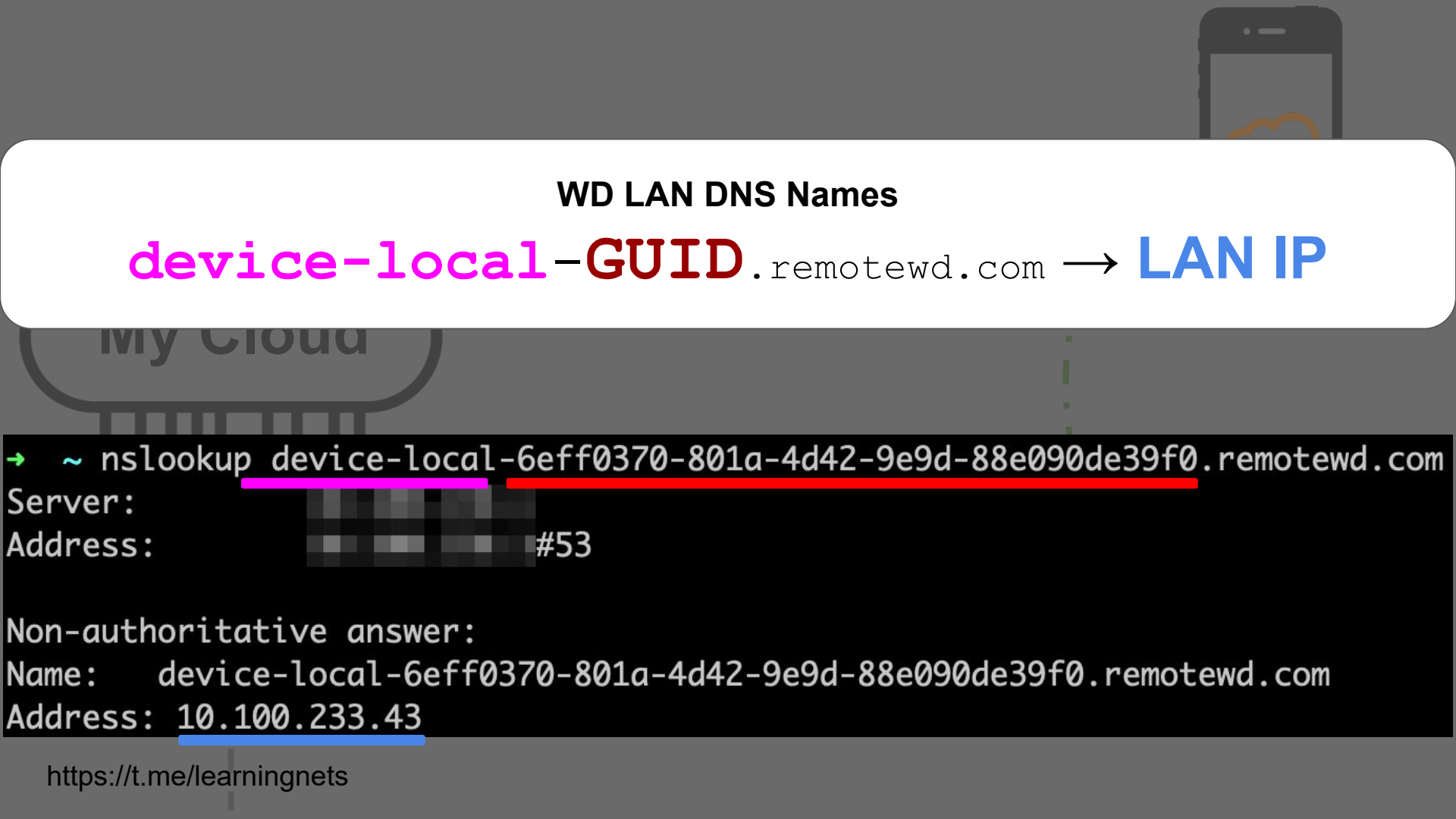
User

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

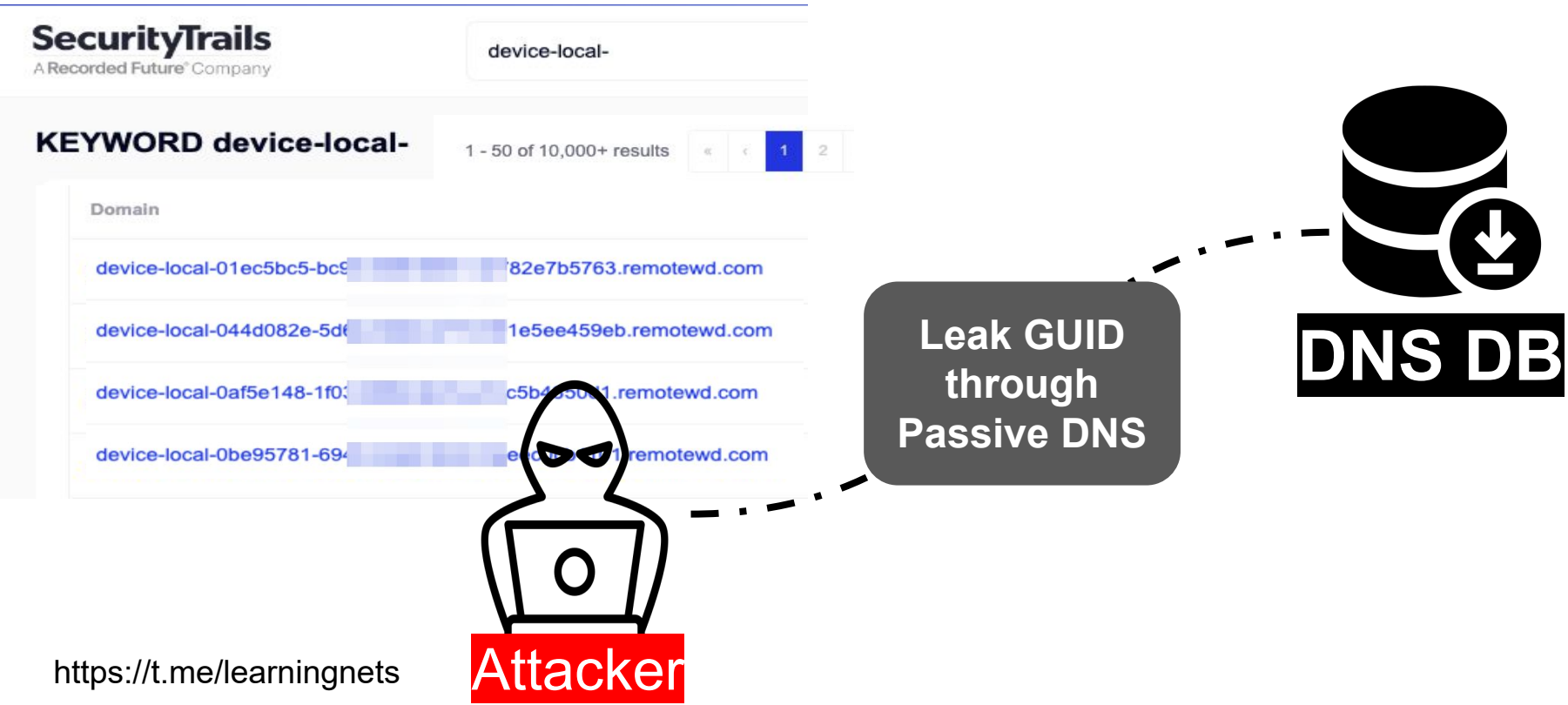


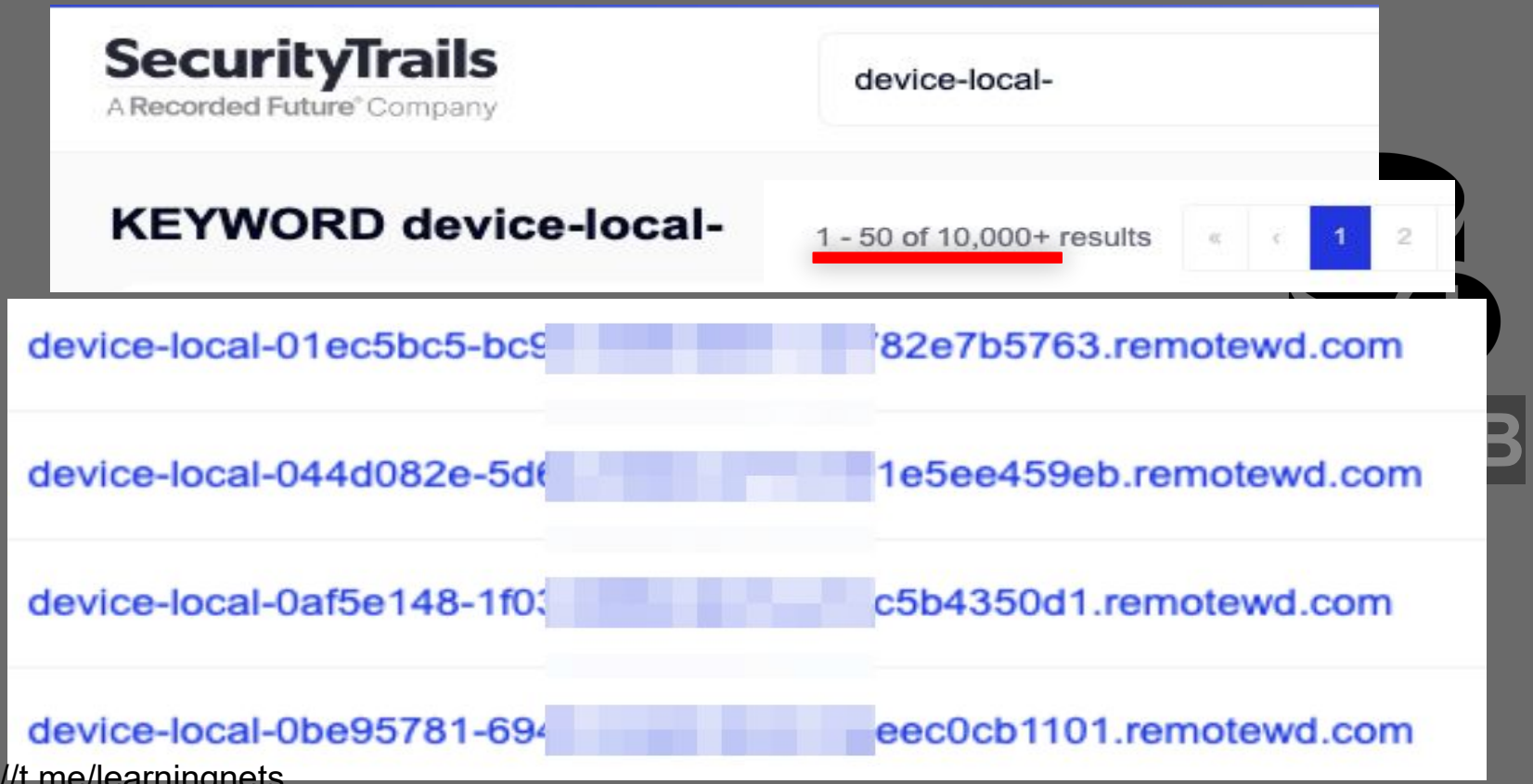
NAS

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



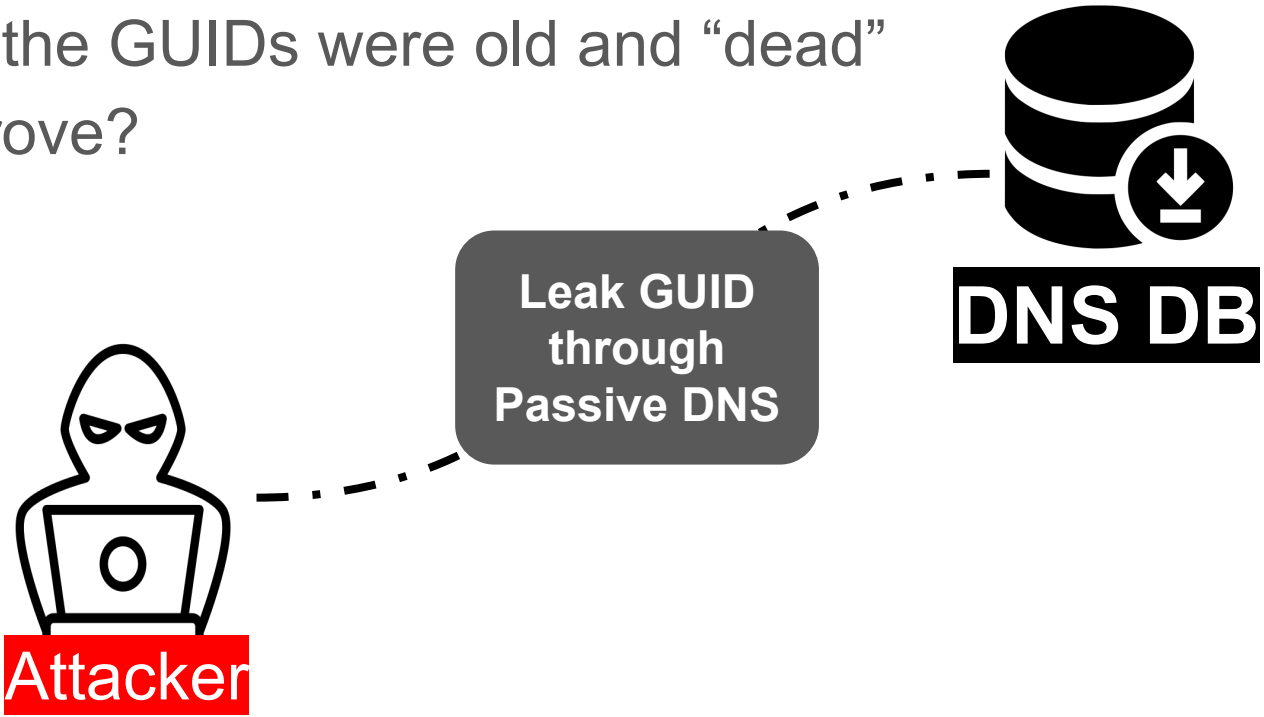
Leak GUIDs Through Passive DNS





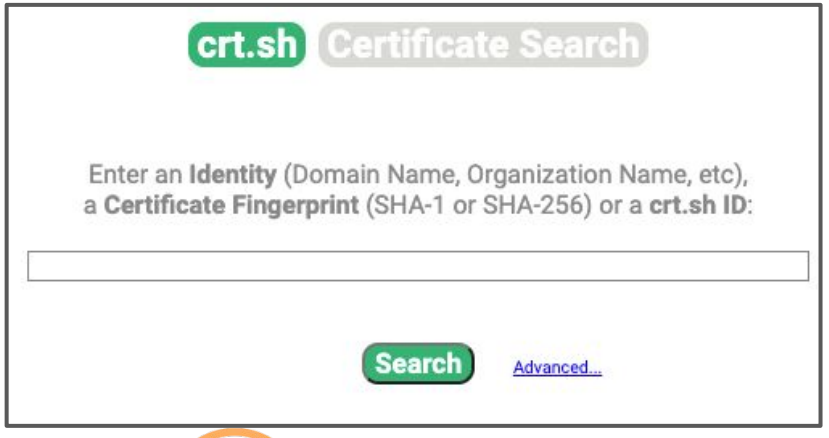
Thousands of Old History DNS Records is Not Enough

- 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



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.

censys

Certificates (device-local-*.remotewd.com) and labels=`unexpired`

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

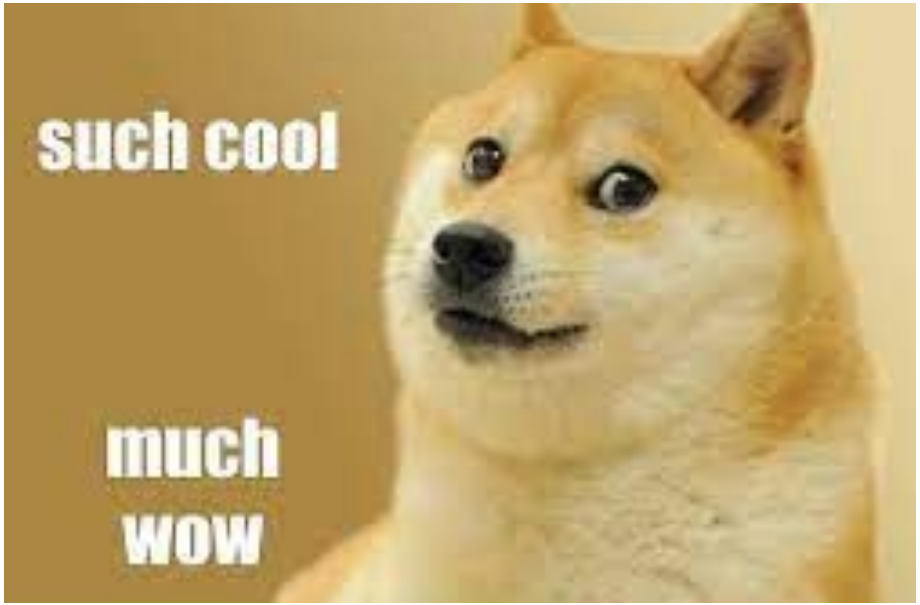
Results: 1,517,264 Time: 8.17s

CN=device-local-0efa3850-e3[redacted]-6ad6286baa5f.remotewd.com
Let's Encrypt R3
2023-06-18 — 2023-09-16
device-local-0efa3850-e34b-4d[redacted]baa5f.remotewd.com

CN=device-local-fd830d3b-c4[redacted]-05f38b31c948.remotewd.com
Let's Encrypt R3
2023-06-18 — 2023-09-16
device-local-fd830d3b-c445-4d[redacted]1c948.remotewd.com

CN=device-local-9e2405a2-ca[redacted]-0bbf1c04af29.remotewd.com
Let's Encrypt R3
2023-06-18 — 2023-09-16
device-local-9e2405a2-ca29-44[redacted]04af29.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

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

Public tools to receive continuous stream

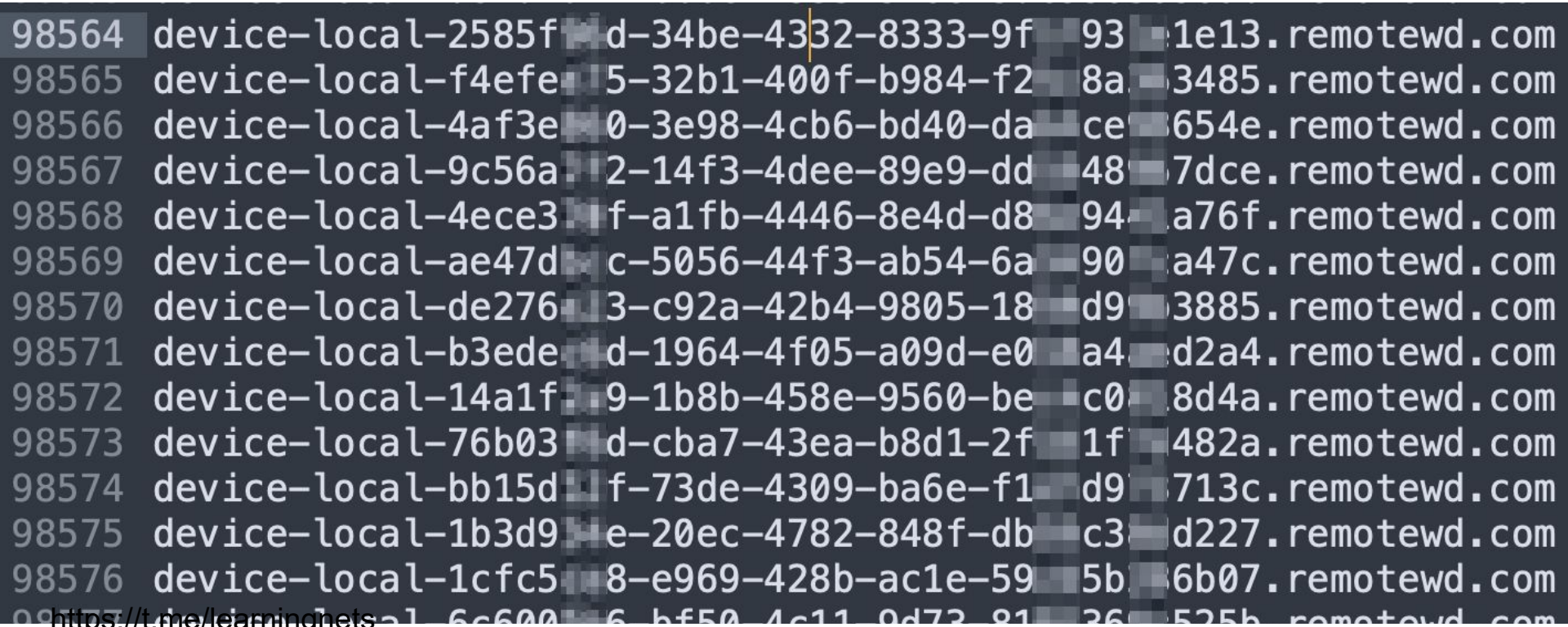
- https://certstream.calidog.io/
- https://ct.cloudflare.com/
- https://nikita-kun.github.io/certificate-transparency-root-explorer/

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!

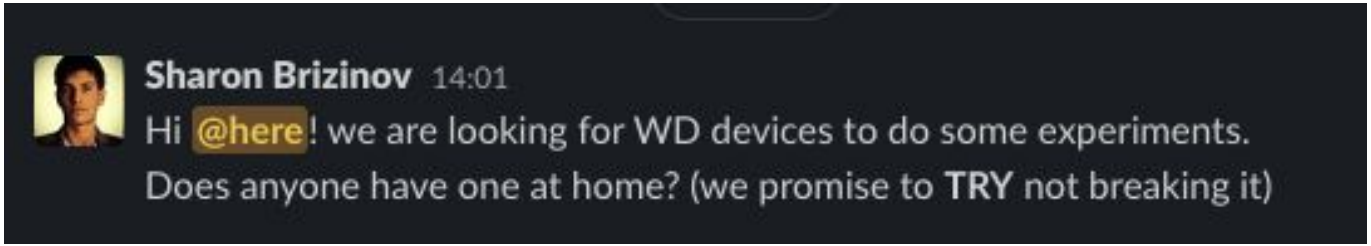


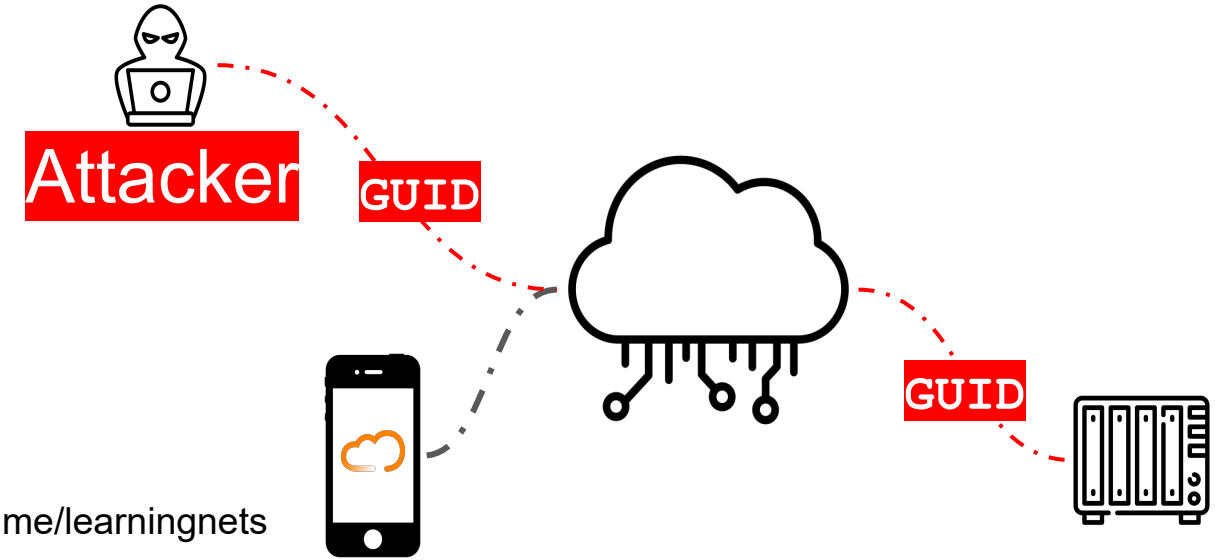
Our Plan to Exploit All Devices

We “just” need to:

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

First, We Tried the Naive Approach





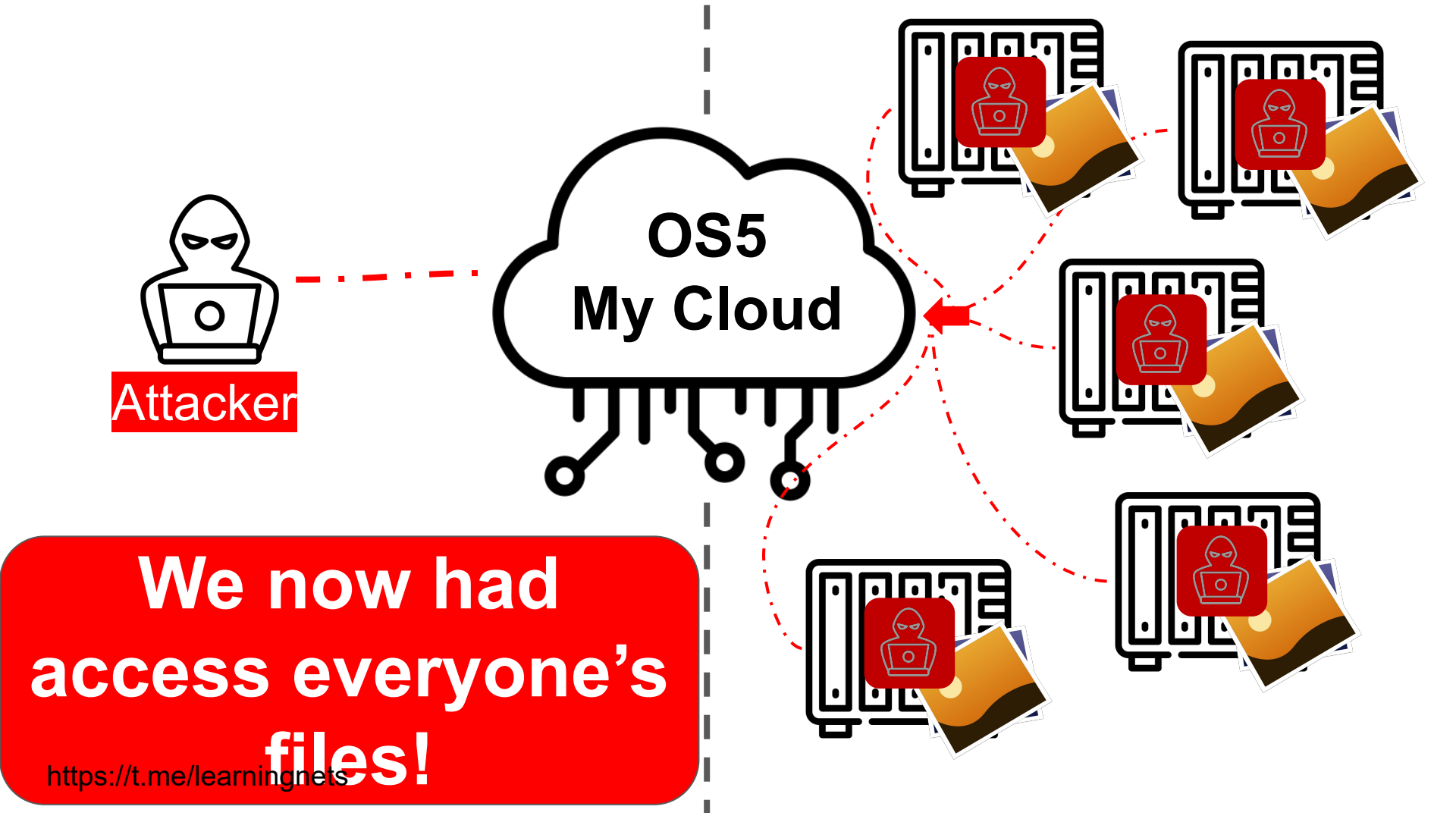
Pretty Raw Hex

```
GET /623[redacted]38/sdk/v1/changes?pageToken=AAAA[redacted]AAAAA&limit=
50%2C20 HTTP/2
Host: prod-534d57eabe6cde9.wdckeystone.com
Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
Pragma: no-cache
X-Correlation-Id: w_g:0b[redacted]
Accept-Language: en-US,en;q=0.8
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer
eyJhbGciOiJSUzI1N[redacted]ZCI6IlFUQkdPVFJCUW[redacted]FSTBNRF
[redacted]VZNE5FS[redacted]wczovL2F1dGgwLmFjY[redacted]dpdGFsL
[redacted]
[redacted]bn7y4Oh[redacted]g3IJJYg
[redacted]HmuTEzY[redacted]iXmUWuu
[redacted]DQgEXVU[redacted]sJO-DHT
SL_emyvSAkg
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
Cache-Control: no-cache
Sec-Ch-Ua-Platform: "macOS"
Accept: */*
Origin: https://os5.mycloud.com
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://os5.mycloud.com/
Accept-Encoding: gzip, deflate
```

GUID

Pretty Raw Hex Render

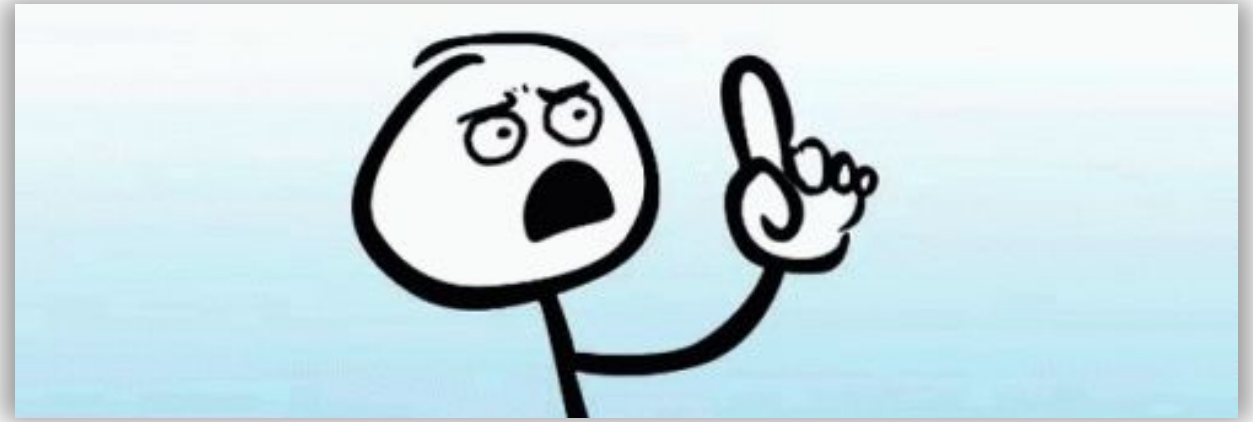
```
HTTP/2 200 OK
Access-Control-Allow-Origin: *
Access-Control-Expose-Headers: Date, Vary, Content-Encoding, Etag, Content
Content-Type: application/json
Date: Sun, 18 Sep 2022 16:00:52 GMT
Etag: "vukELyPbPf[redacted]eJuM4s1gpM8VAYI"
Vary: Origin
Content-Length: 1950

{
  "pageToken":"AAAAAA[redacted]KQ",
  "changes":[
    {
      "fileID":"zhg4xw[redacted]24j4a"
    },
    {
      "fileID":"cqc43[redacted]kzpda"
    },
    {
      "fileID":"lw3k4[redacted].72qqh"
    },
    {
      "fileID":"ydcrmw[redacted]wae7y"
    },
    {
      "fileID":"k22o4l[redacted]tiq5m"
    },
    {
      "fileID":"7kdq4c[redacted]bdl5n"
    },
    {
      "fileID":"ibqk2l[redacted]nofbw"
    },
    {
      "fileID":"skvkqc[redacted]rzjnr"
    },
    {
      "fileID":"vt3xbe[redacted]u2rm2"
    },
    {
      "fileID":"uzvm3[redacted]ye6yi"
    },
    {
      "fileID":"whbihc[redacted]qbcmq"
    },
```



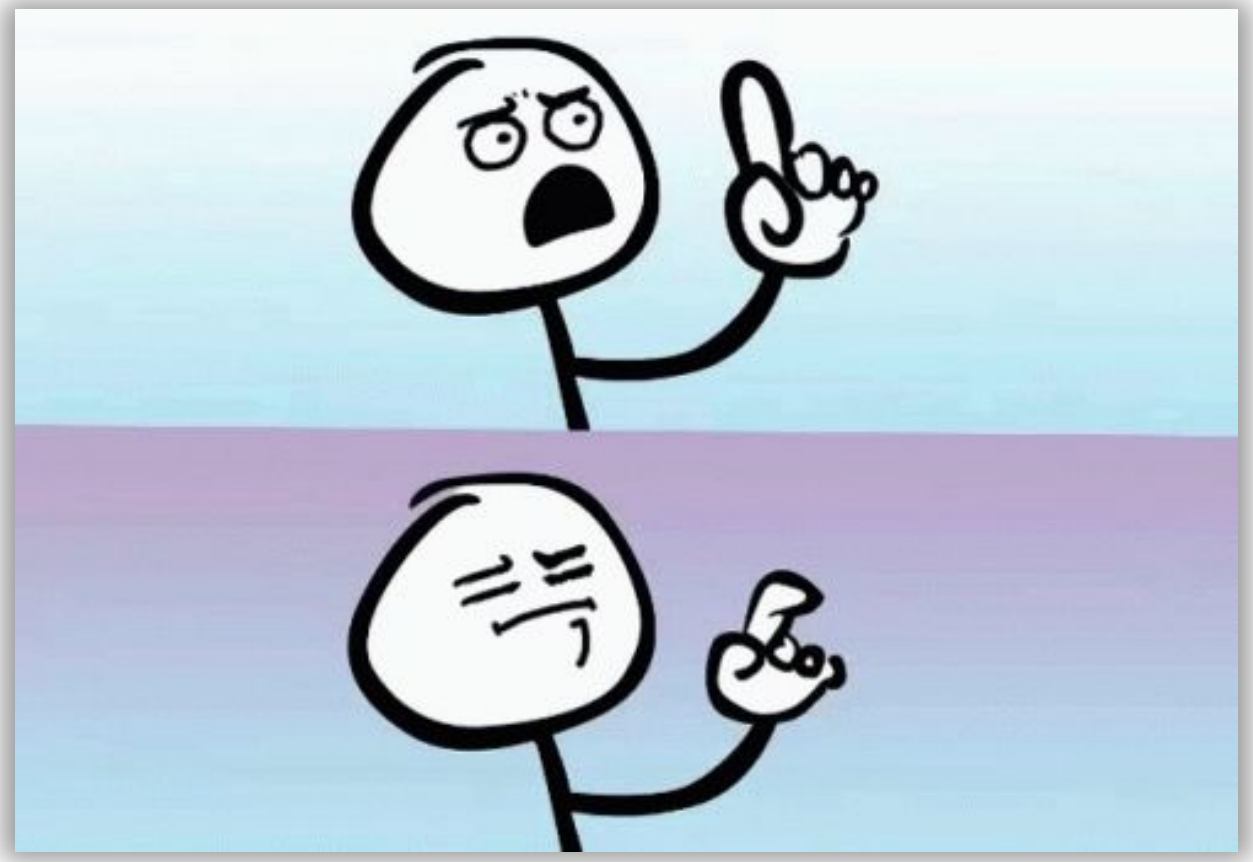
Our Plan to Exploit All Devices

We “just” need to:

- [x] ~~Break~~ Leak all GUIDs
- [x] 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!

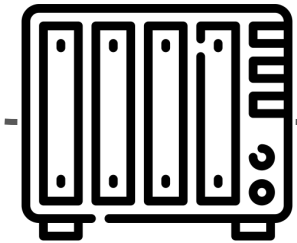
```
root@MyCloudPR4100 ~ # ps aux | grep restsdk
restsdk-server -minimal -configPath /usr/loc
```

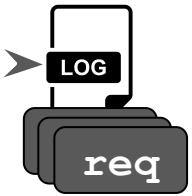
RestSDK - Enable Logging

Special **http2-golang** debug flags for detailed tracing

- `export GODEBUG=http2debug=2`


```
log","file":"log.go","fn":"ErrorWriter.W
,"line":643,"message":"2022/10/27 03:21:
ttp2: decoded hpack field header field
":path\" = \'/sdk/v1/devicePerms\'\n"}]}
"file":"log.go","fn":"(*Logger).Output",
ime":1661485041,"githash":"bbcbc2f","imp
th":"log","line":195,"msgid":"error","tr
[{"importPath":"github.com/wdc-csbu/gosh
```



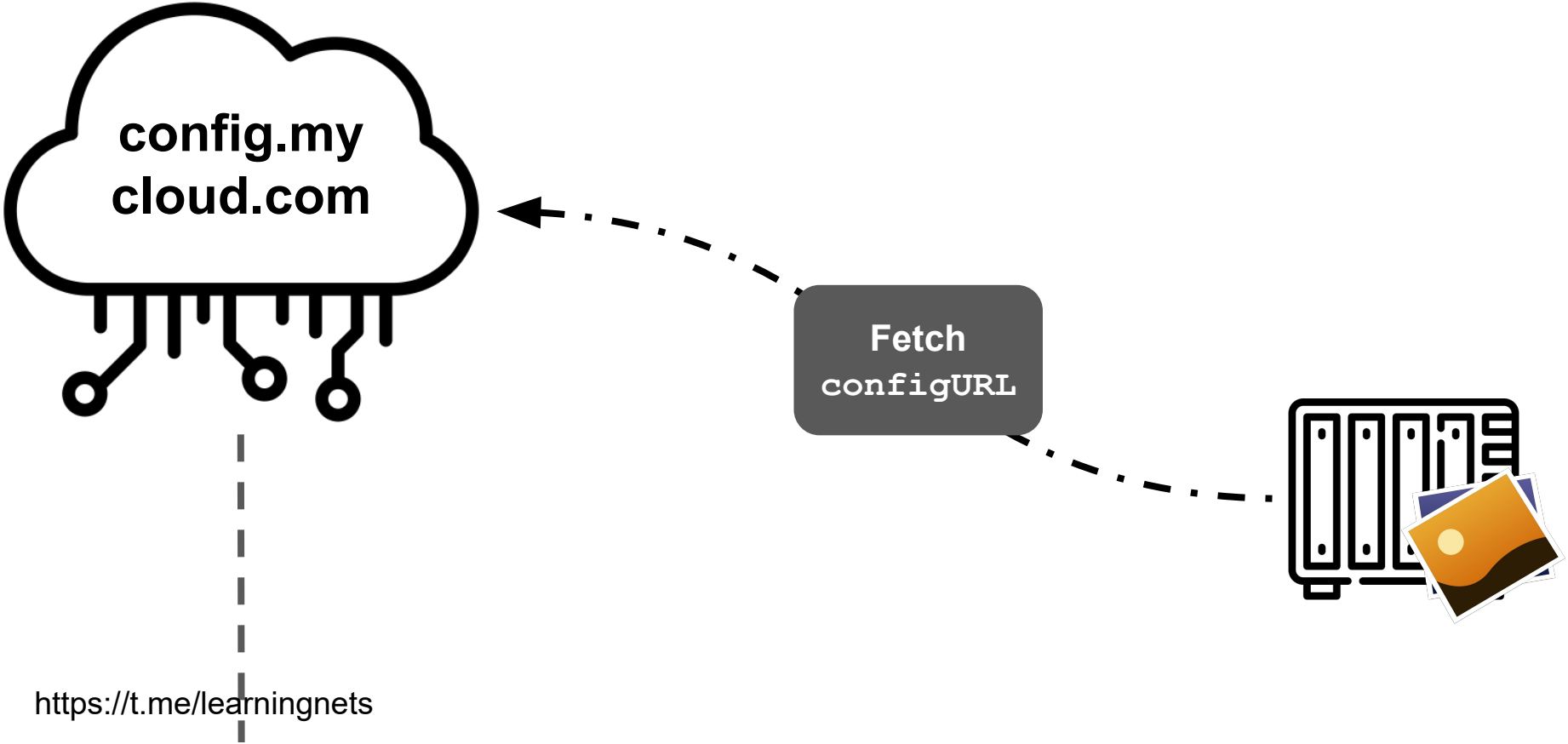
RestSDK - MITM

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

- 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`

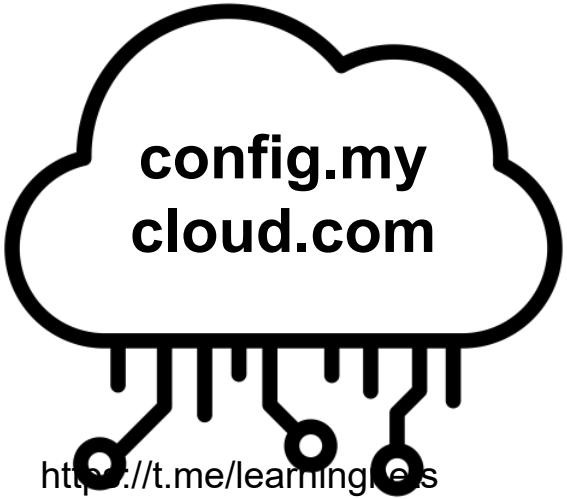
Get Cloud Configuration



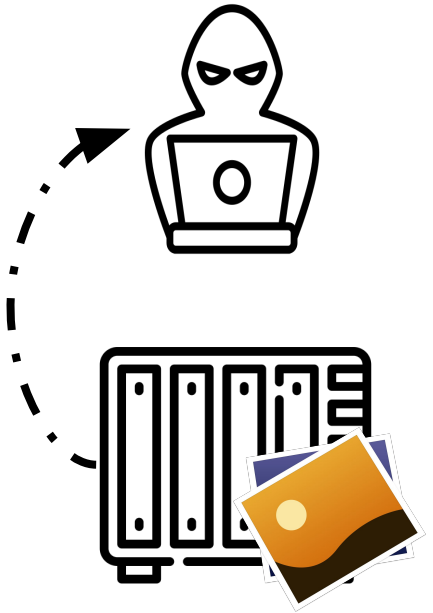
RestSDK - MITM - Step 1: Point Config to Us

Step 1
Edit `configURL`
to point to us

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



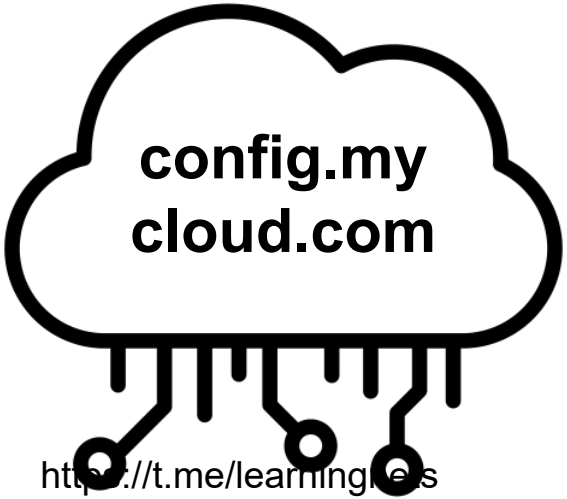
Fetch
`configURL`



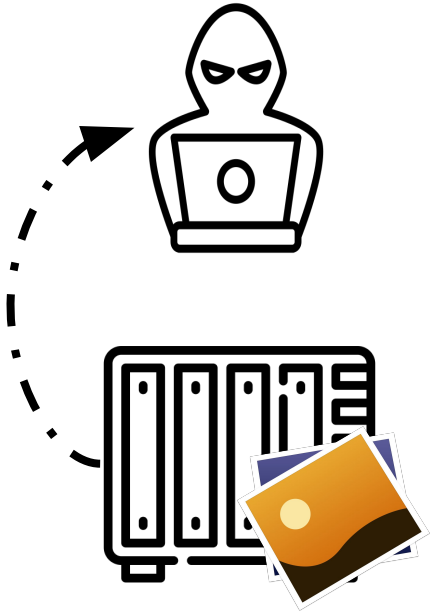
RestSDK - MITM - Step 2: Downgrade HTTPS

Step 2
Downgrade to HTTP

```
rvice.tsm.url": "http://prod-proxy.wdckeystone.com",
rvice.proxy.url": "http://prod-proxy.wdckeystone.com",
rvice.ota.url": "http://prod-gateway.wdckeystone.com/ota",
alytics.url": "http://prod-gateway.wdckeystone.com/logreceiver/re
bclient.new_session.url": "http://home.mycloud.com/sessions/new",
```

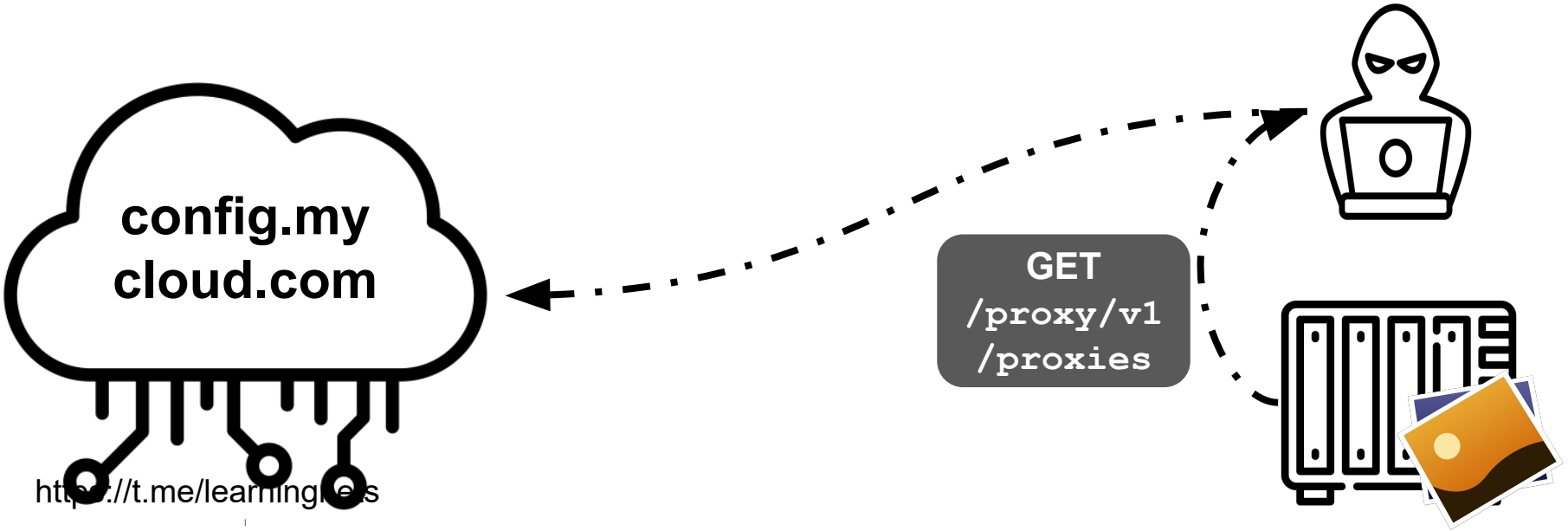


Fetch
`configURL`



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)

RestSDK - MiTM - Step 4: MiTM HTTPS, too

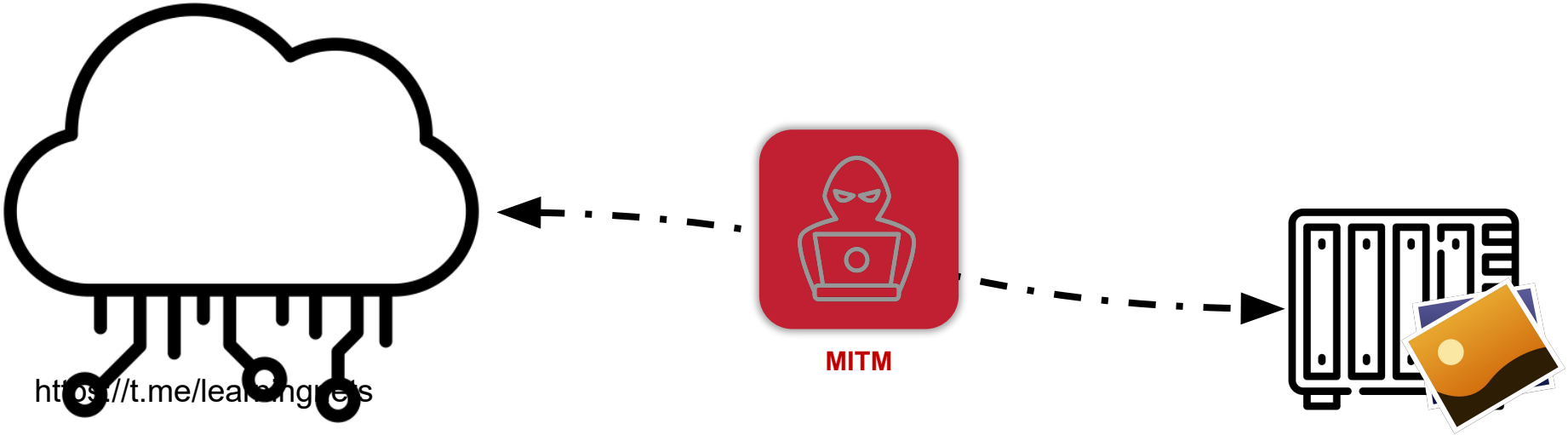
We must MiTM HTTPS -

Create fake certs + add ourselves as the CA

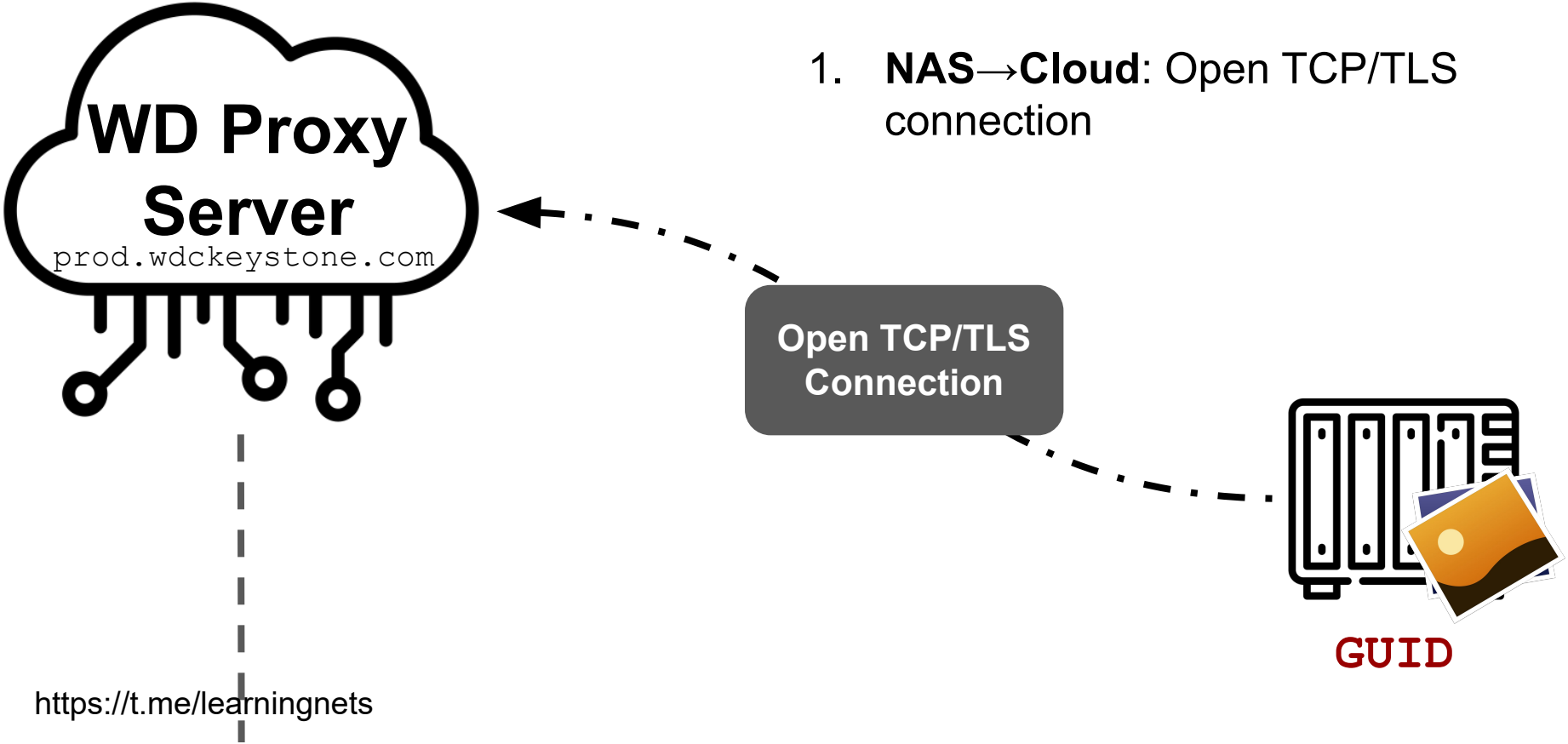
```
→  keys openssl x509 -in wd.specific.crt -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            ae:80:85:14:3f:2c:a8:ad
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=prod-b083c6039a50497.wdckeystone.com
        Validity
            Not Before: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



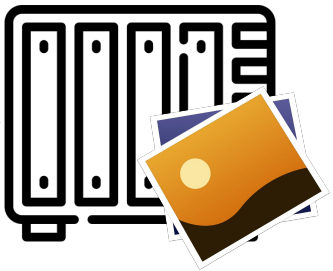
WD Cloud Tunnel Establishment



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

WD Cloud Tunnel Establishment

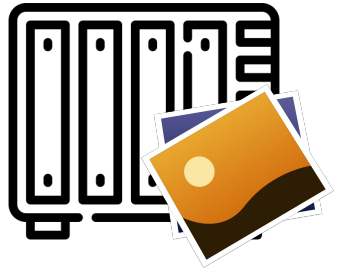
1. **NAS→Cloud**: Open TCP/TLS connection
2. **Cloud→NAS**: GET /sdk/v1/proxyConnect



GUID

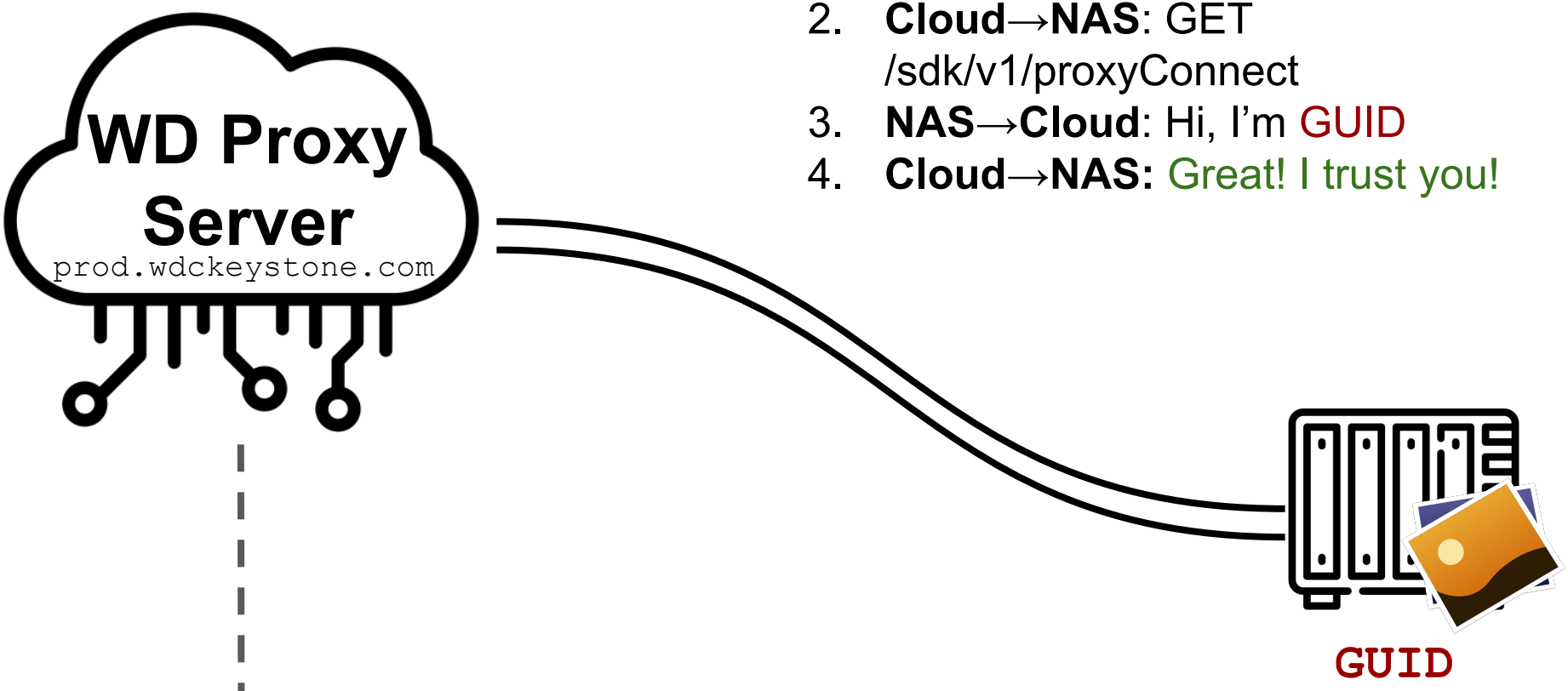
1. **NAS→Cloud**: Open TCP/TLS connection
2. **Cloud→NAS**: GET /sdk/v1/proxyConnect
3. **NAS→Cloud**: Hi, I'm GUID

200 OK
Hi, I'm GUID

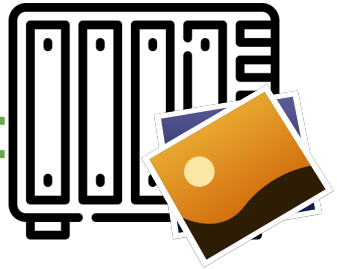


GUID

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!



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

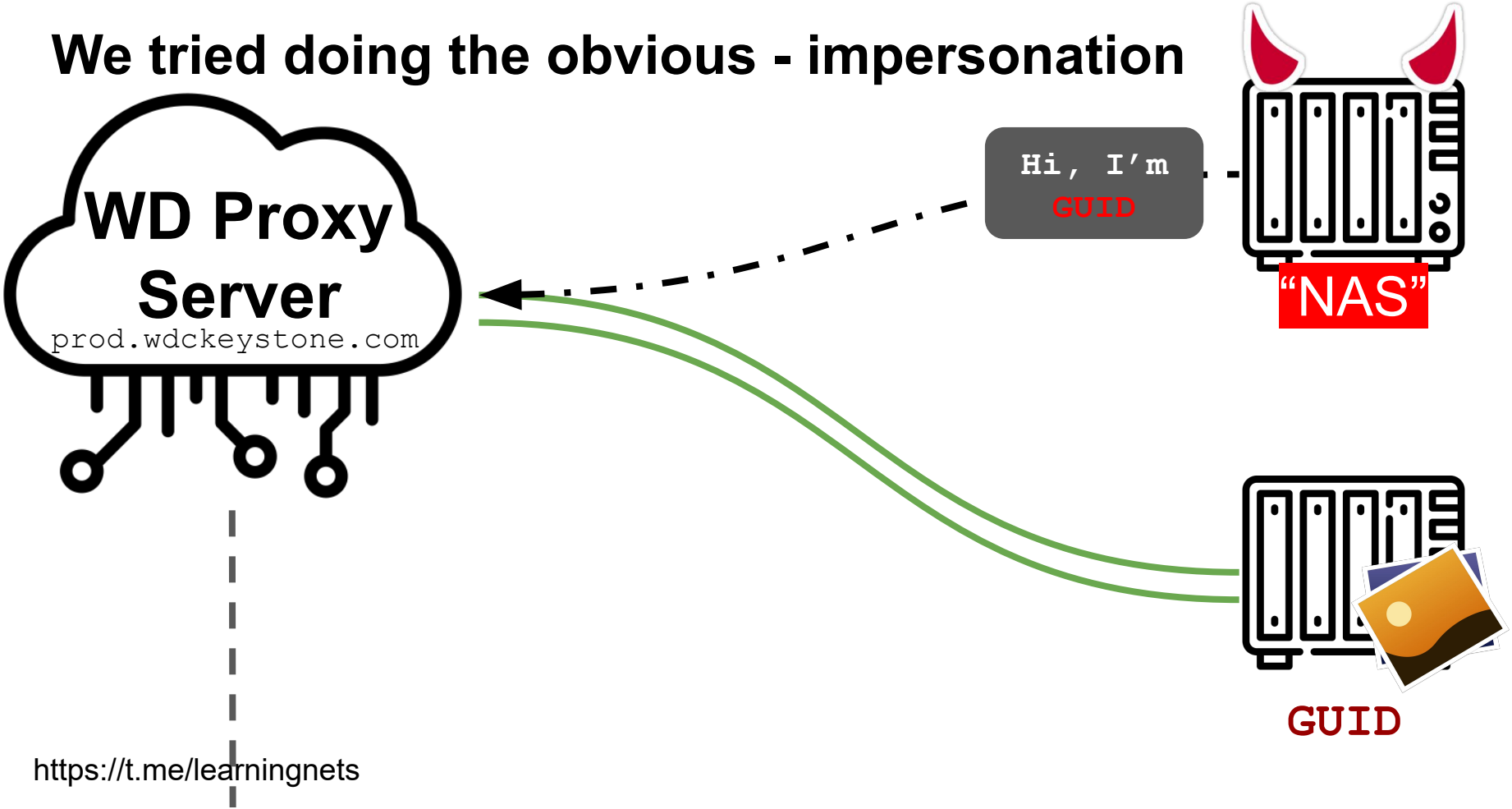
n TCP/TLS

ect
'm GUID
at! I trust you!
unnel

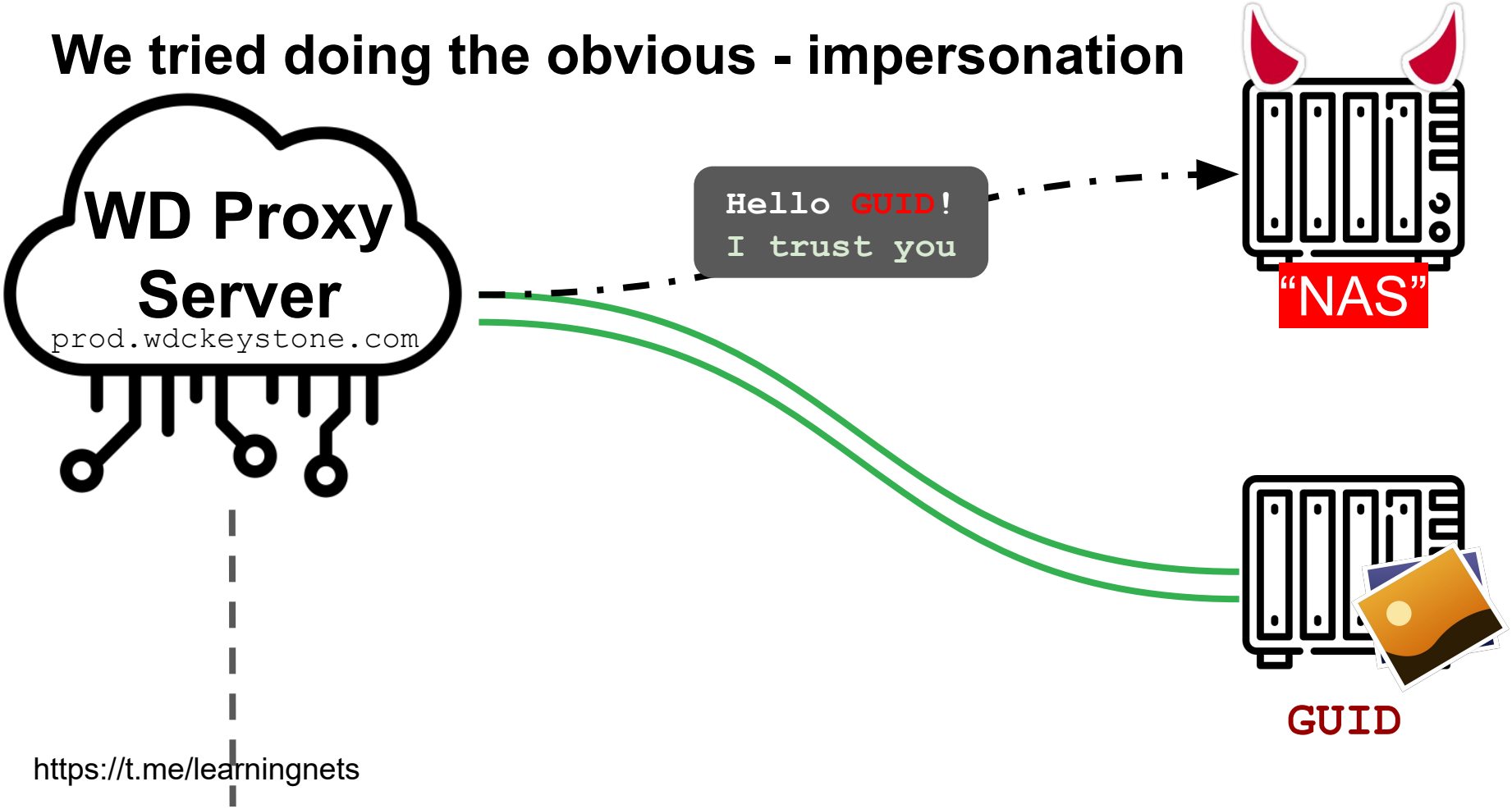
WD P
Serv
prod.wdckey

GUID

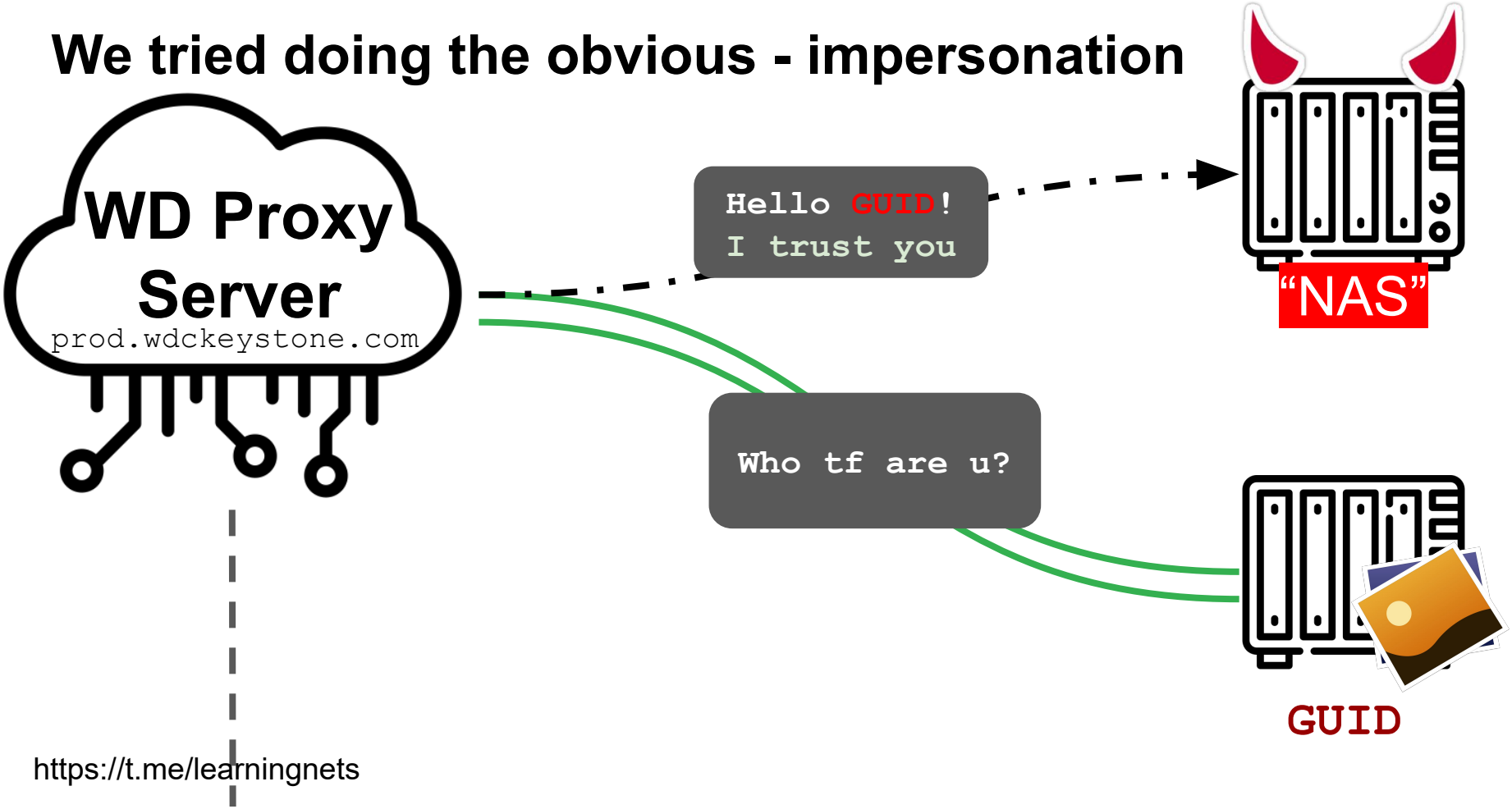
We tried doing the obvious - impersonation



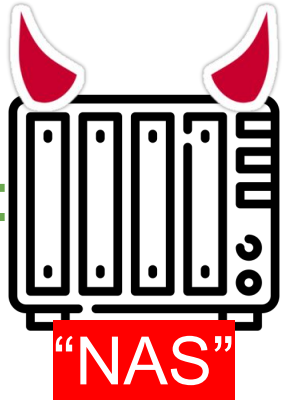
We tried doing the obvious - impersonation

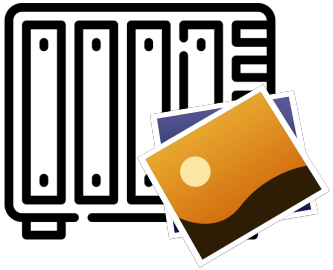


We tried doing the obvious - impersonation



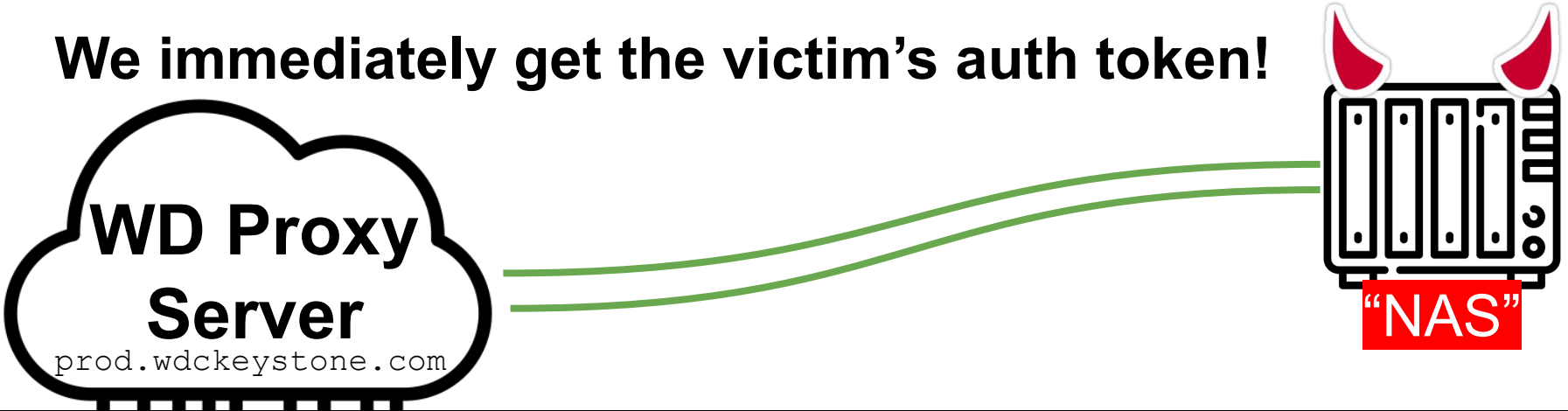
We tried doing the obvious - impersonation



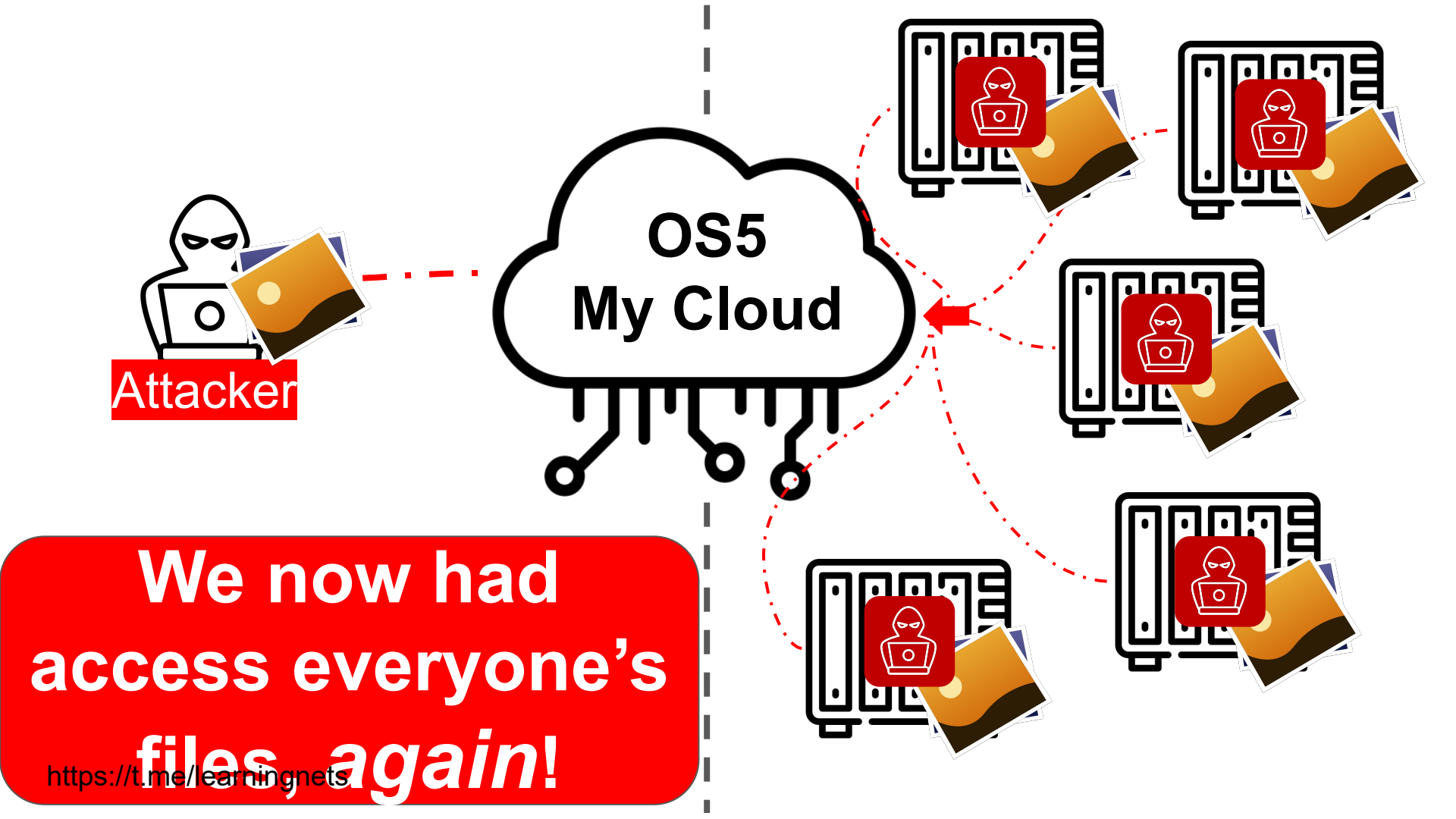


GUID

We immediately get the victim's auth token!




```
GET /sdk/v1/volumes?pretty=false&limit=1000 HTTP
authority: 127.0.0.1
authorization: Bearer eyJhbGc                    Is
                5jb20iLCJodHRw                   hd
                ZXZpY2VfYXR0YW                   Vg
```



Our Plan to Exploit All Devices

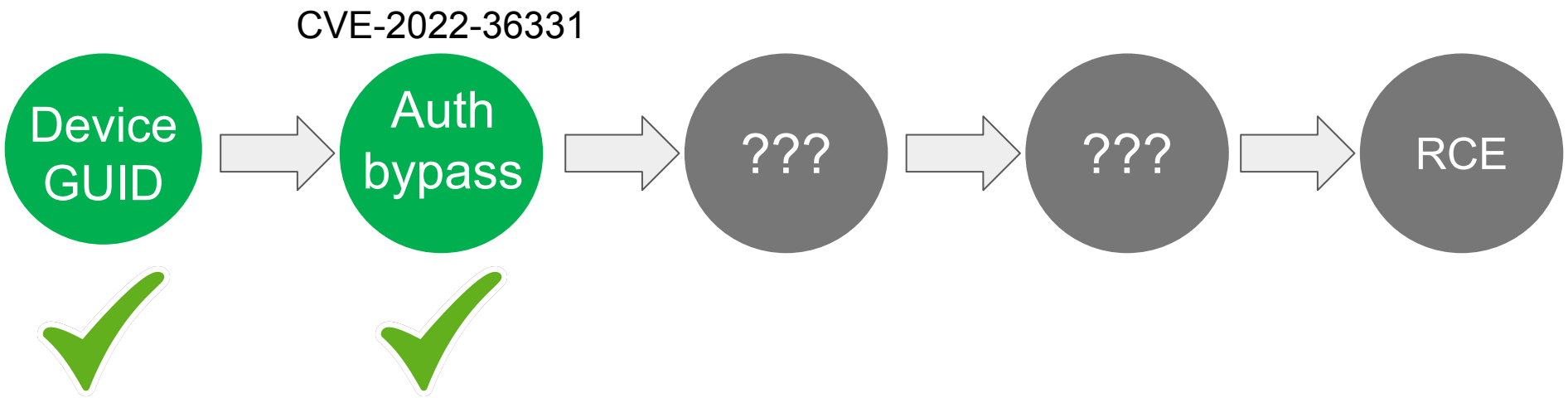
We “just” need to:

- [x] Break 128 bit random GUID
- [x] Find auth bypass, twice!
- [ ] **Find RCE**

Leveraging Cloud Access to Achieve RCE

Sadly, admin !== RCE

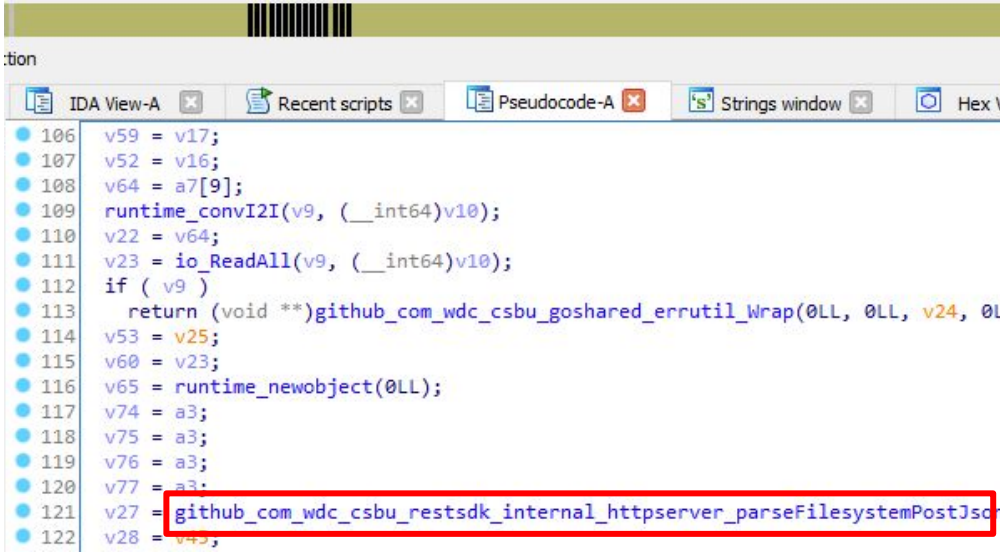
We have auth bypass, what's the attack surface?

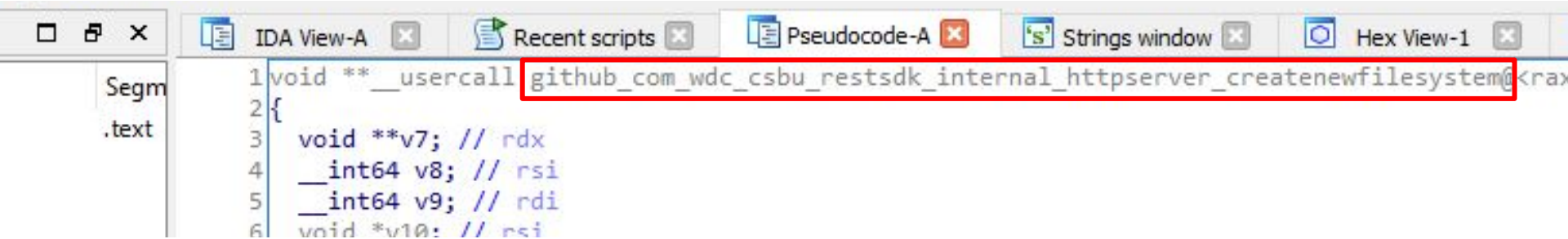


restSDK RE

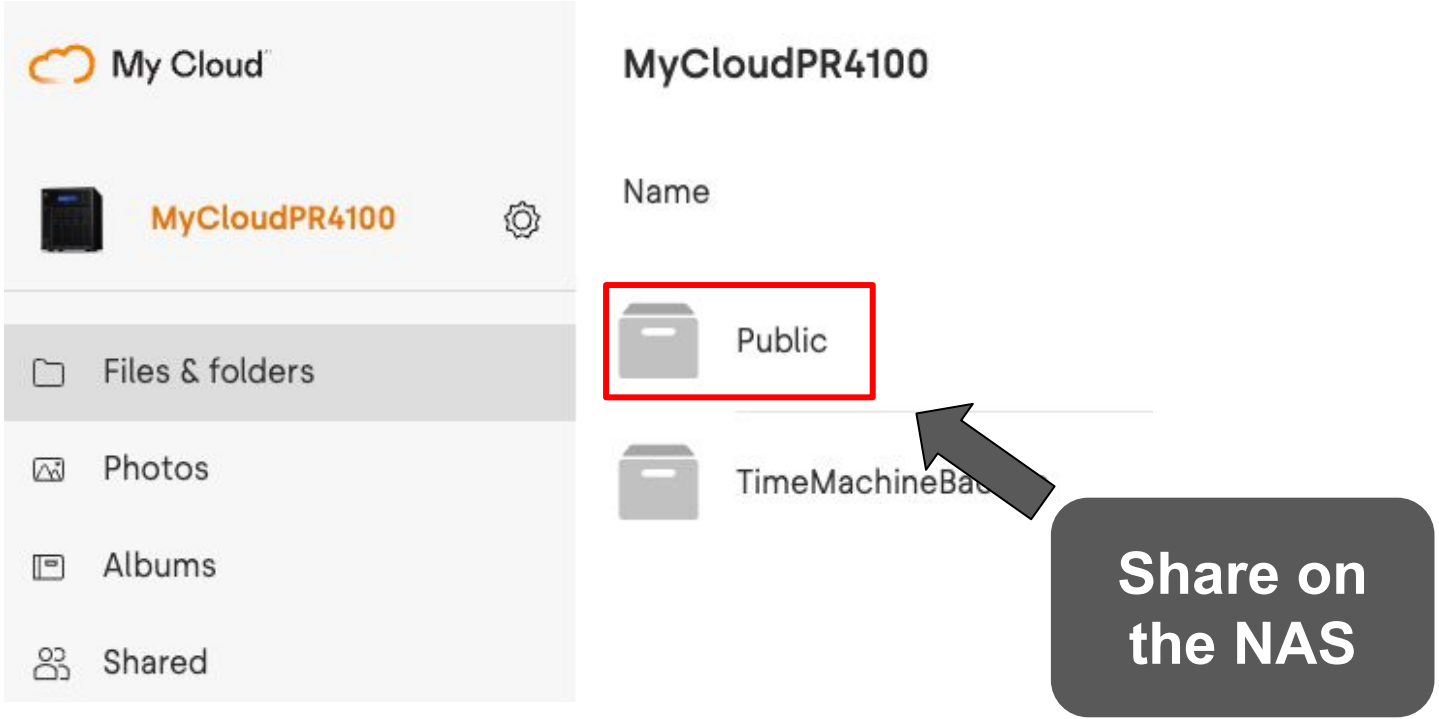
Reverse-engineered hundreds of golang functions

Tried to understand what API is available to auth users

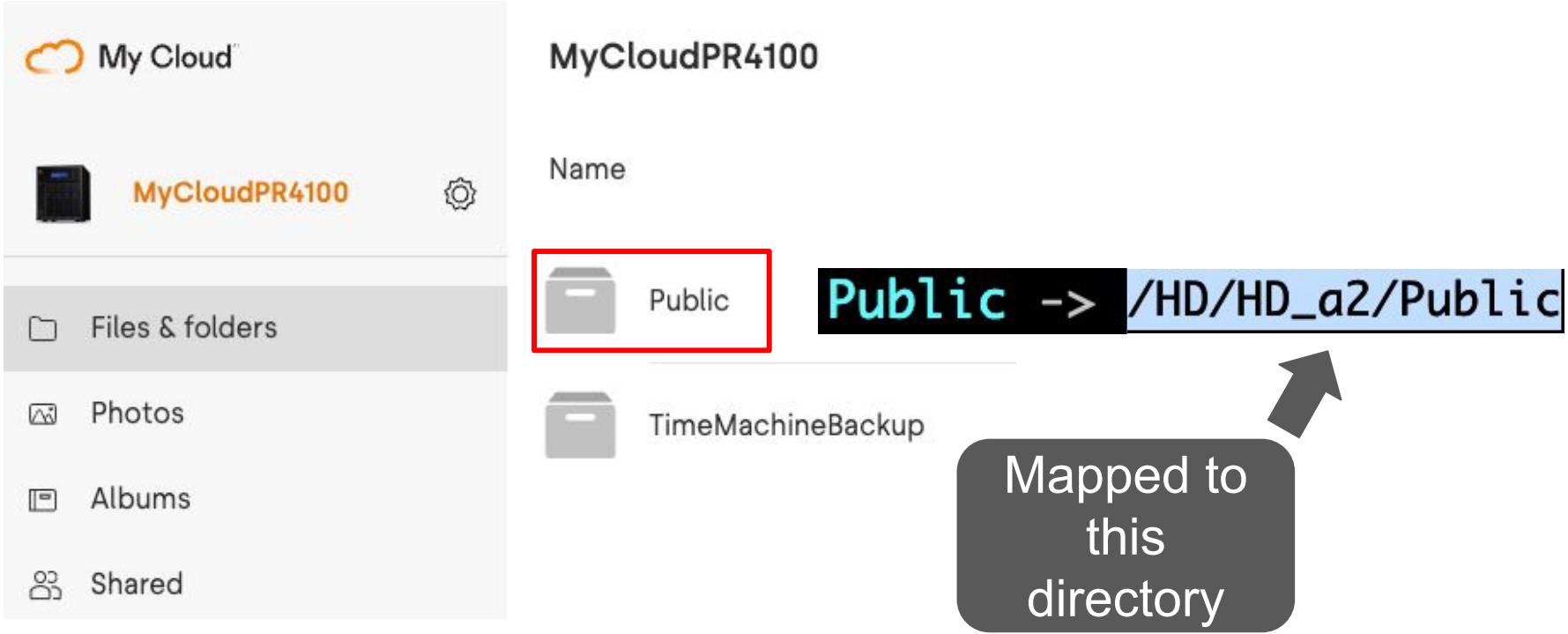




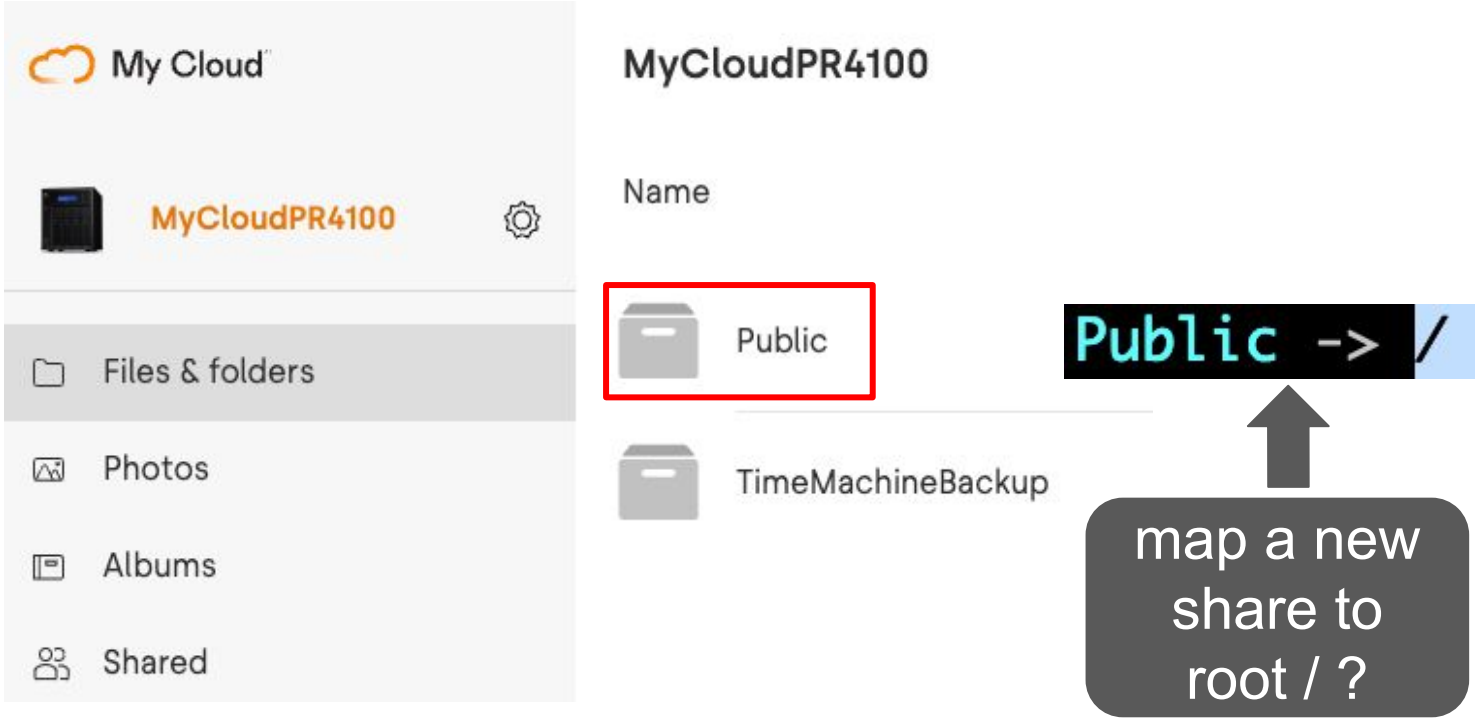
OS5 My Cloud Application - Shares and Mounts



OS5 My Cloud Application - Shares and Mounts

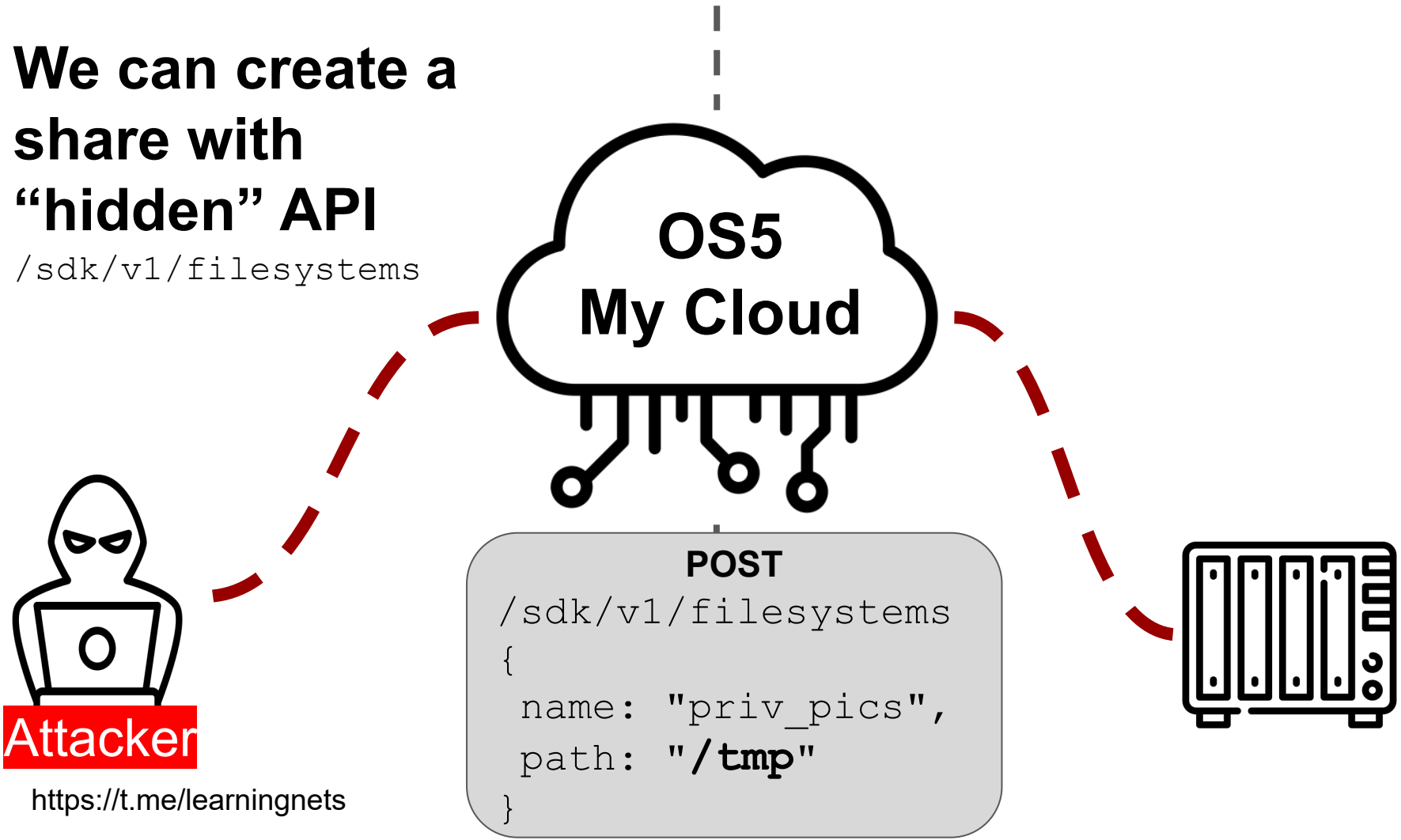


The Obvious Question



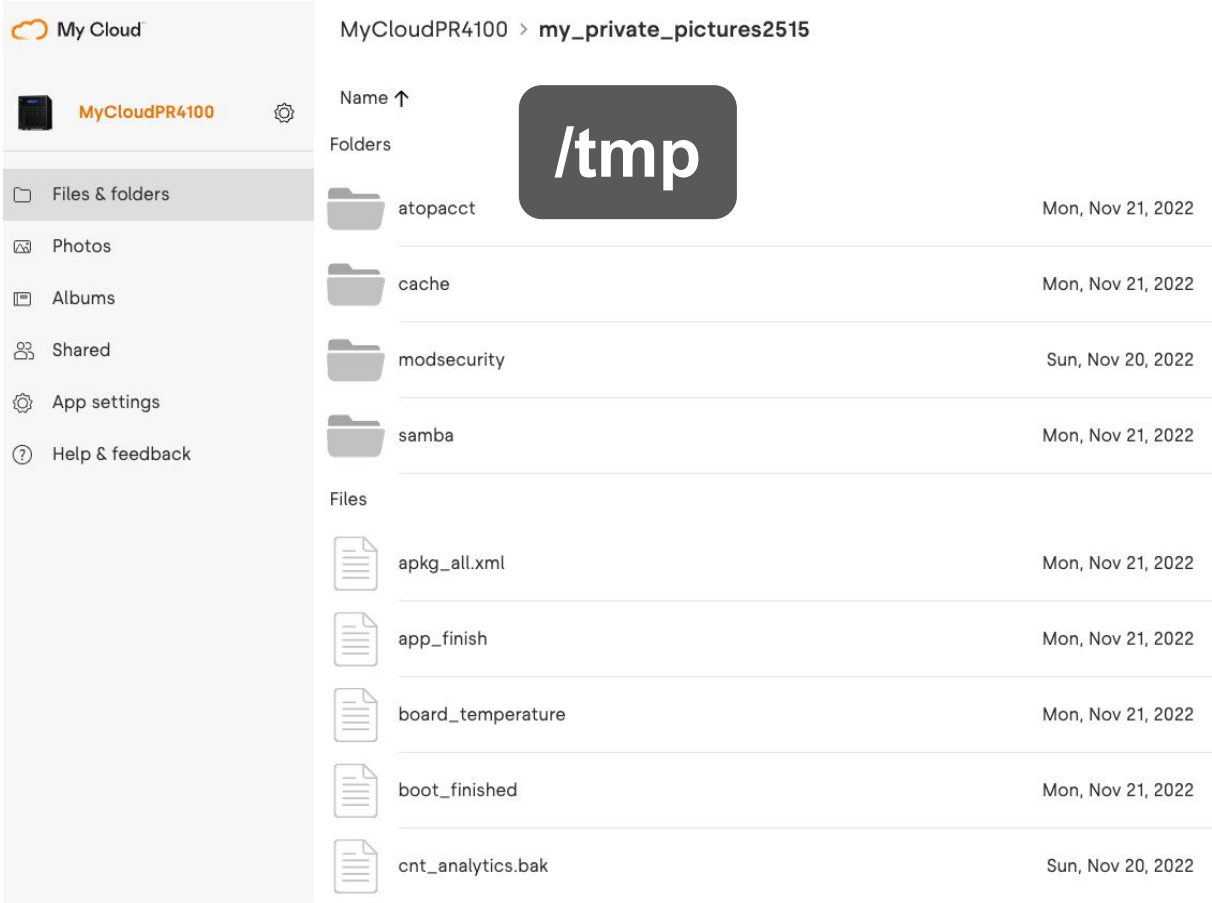
We can create a share with “hidden” API

`/sdk/v1/filesystems`

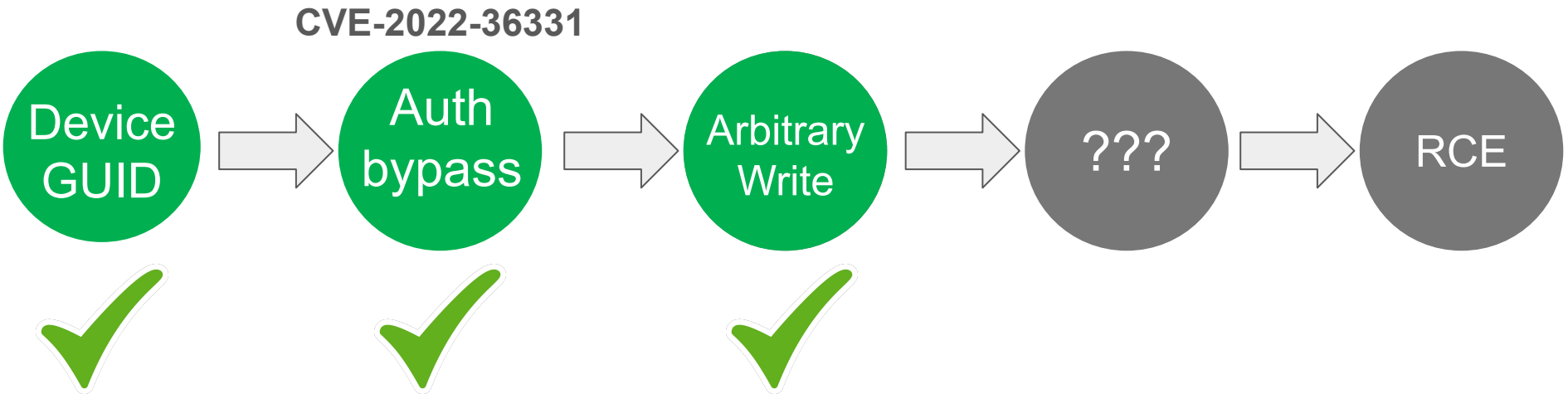


New share mapped to `/tmp`

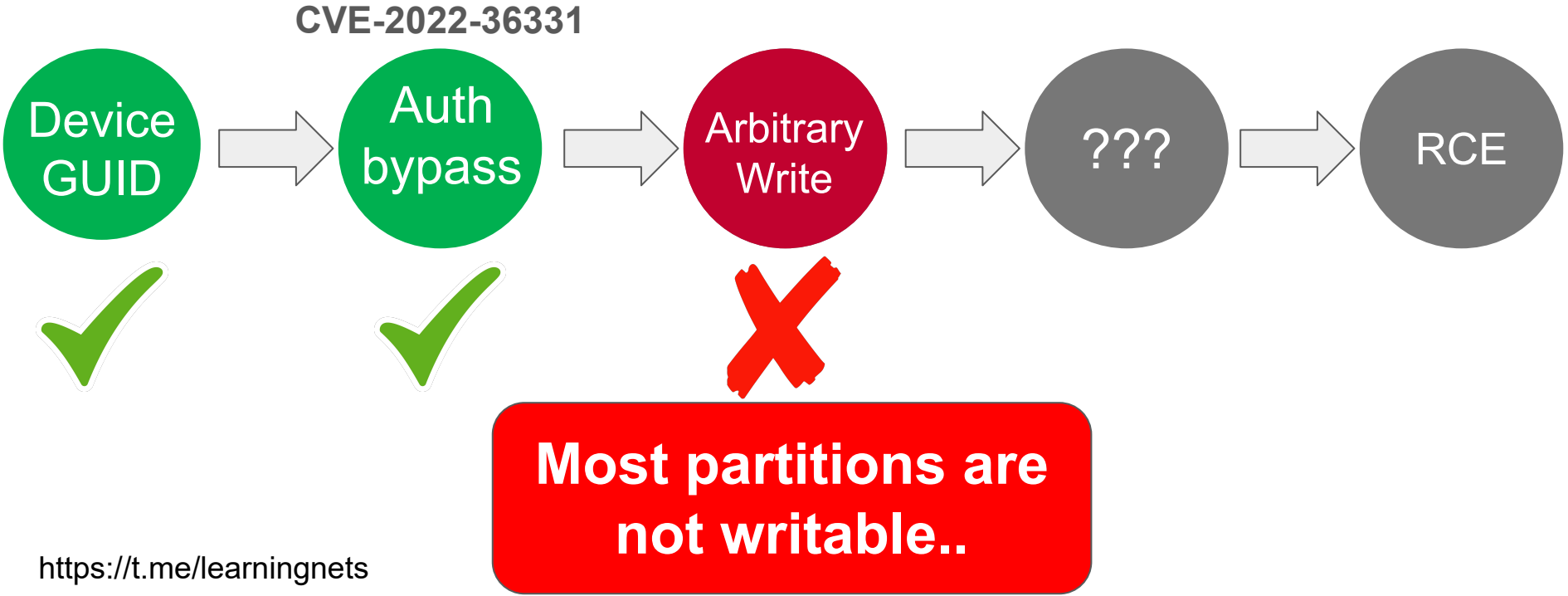
Now we can interact with files under `/tmp` - read/write



Leveraging Cloud Access to Achieve RCE

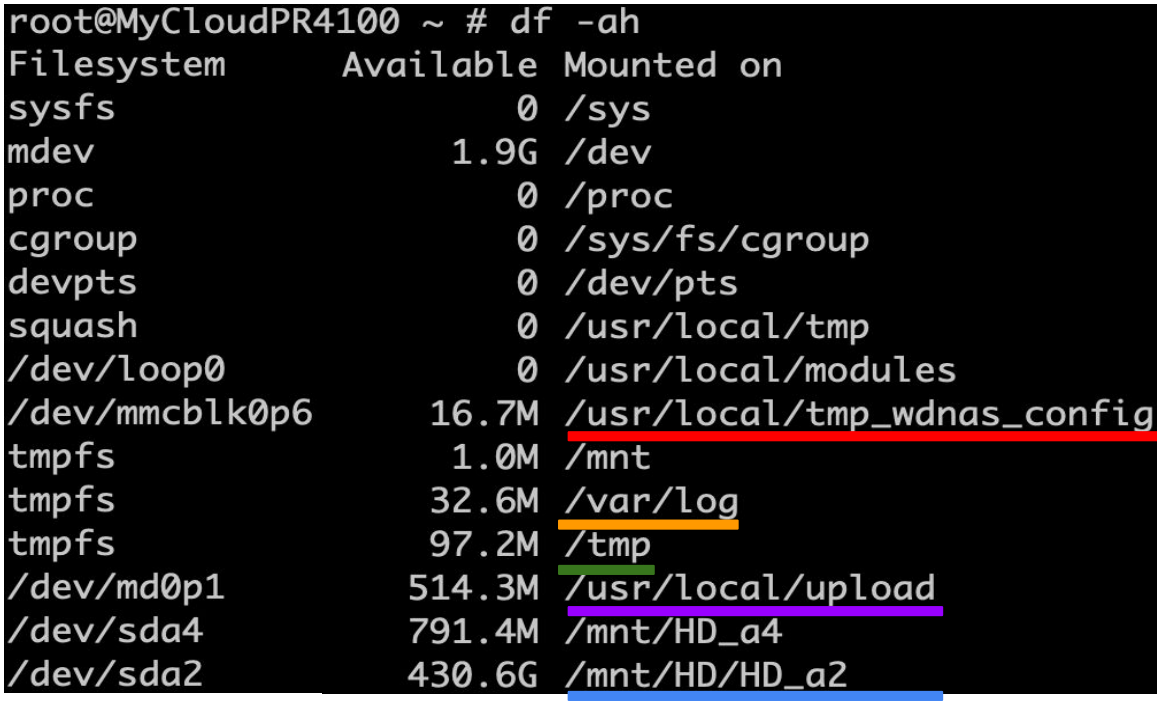


Leveraging Cloud Access to Achieve RCE

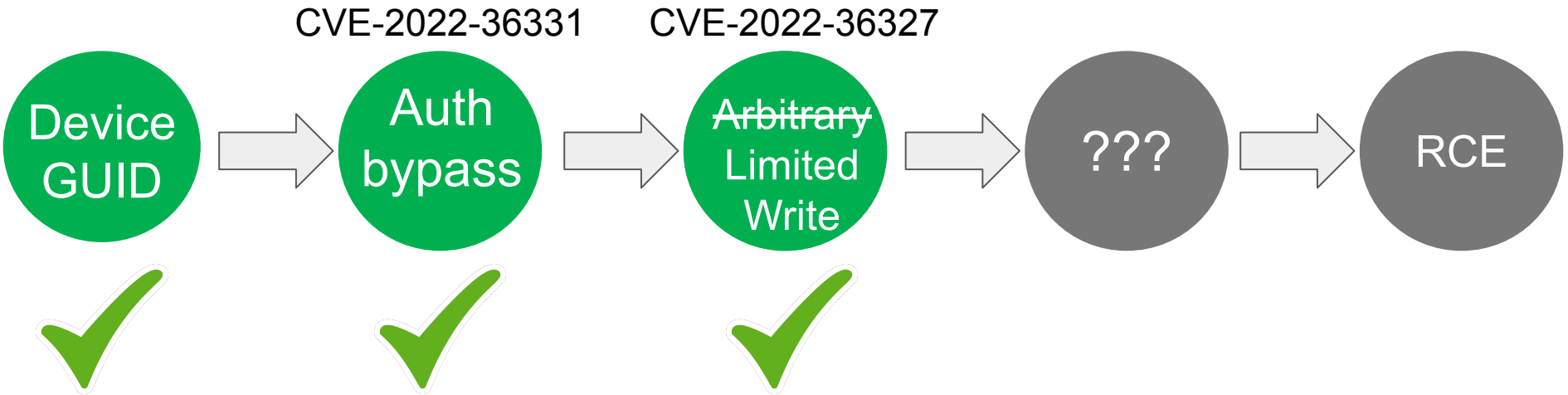


We Could Only Write To..

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

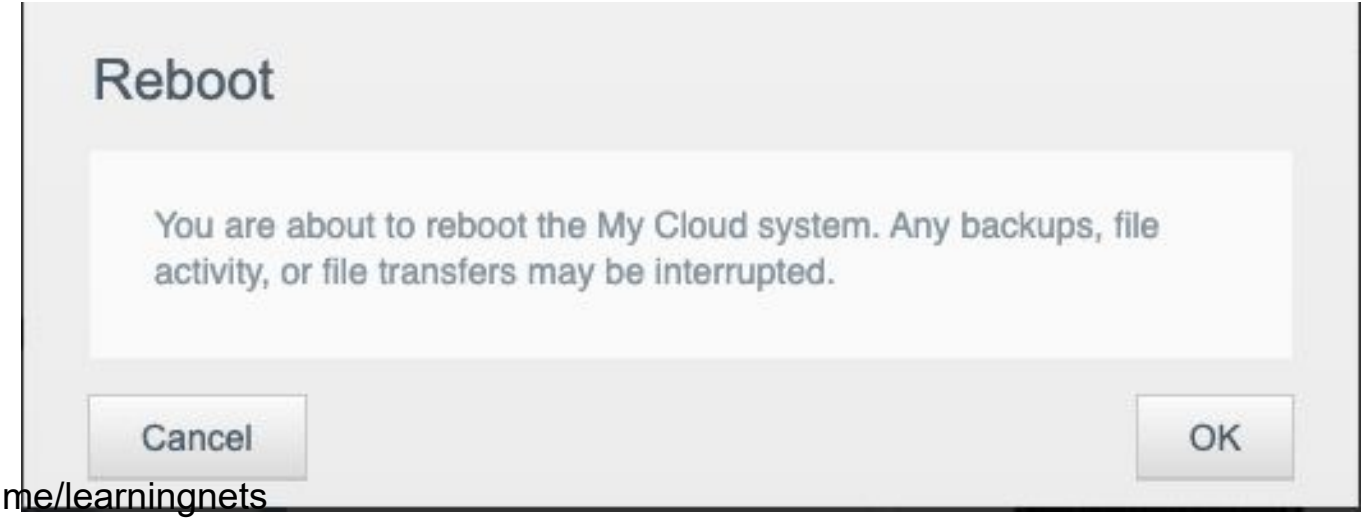


Leveraging Cloud Access to Achieve RCE



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?

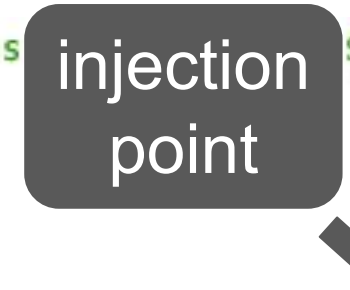
`/tmp/upload_fw_success` is read

```
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);
```



How to RCE Using /tmp?

- `do_reboot` is vulnerable to command injection

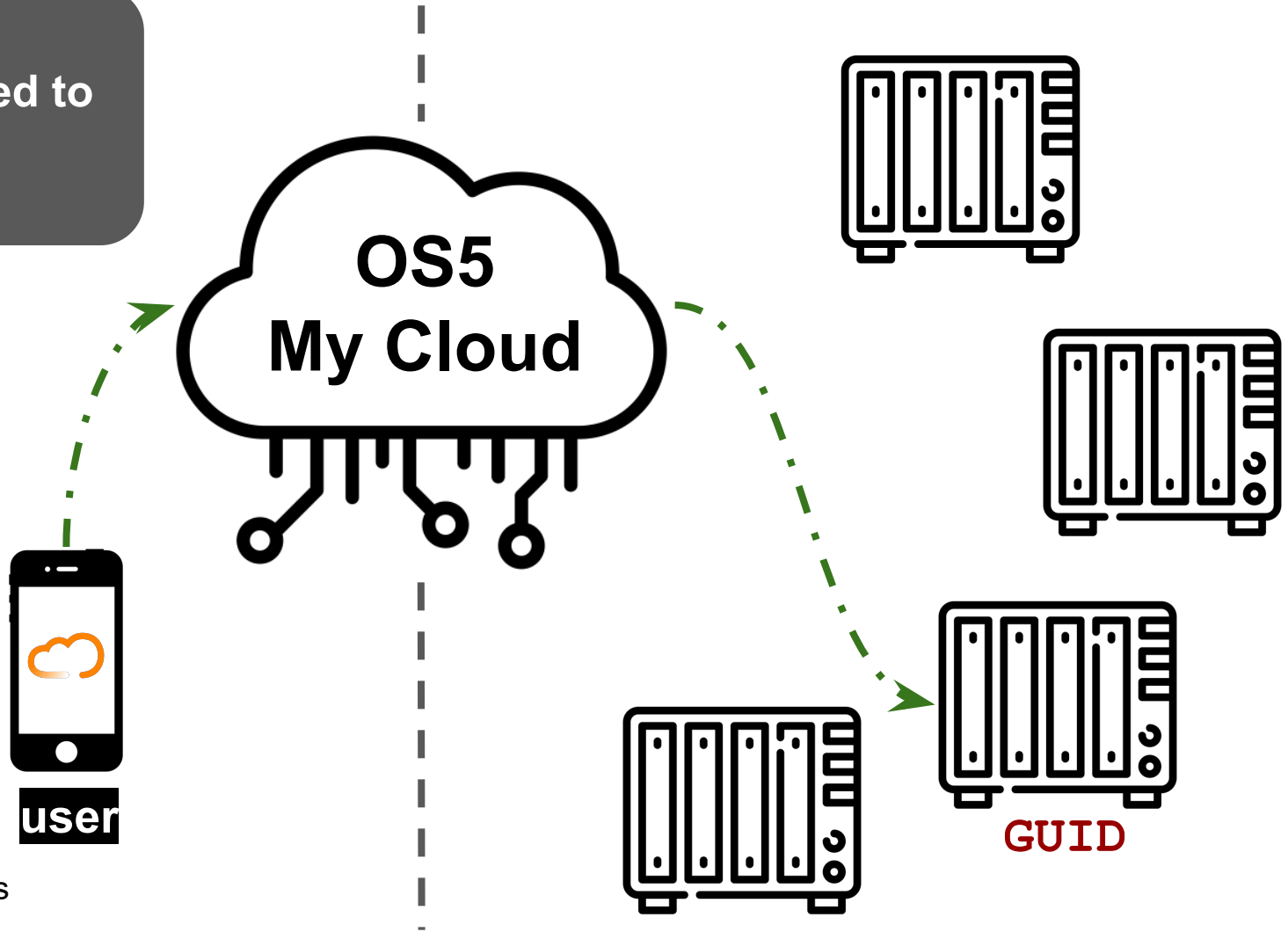
```
pFVar5 = popen("cat /tmp/upload_fw_success | awk \'{print $3}\'","r");
if (pFVar5 != (FILE *)0x0) {
    fgets(pFileContent,0x3f,1,pFVar5);
    pclose(pFVar5);
    strcat(pCommand,pFileContent);
}
sprintf((char *)pCommand,"logwdfw --gza_fw_install --corid \"%s\" --status rebooting",
        pFileContent);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

Executed

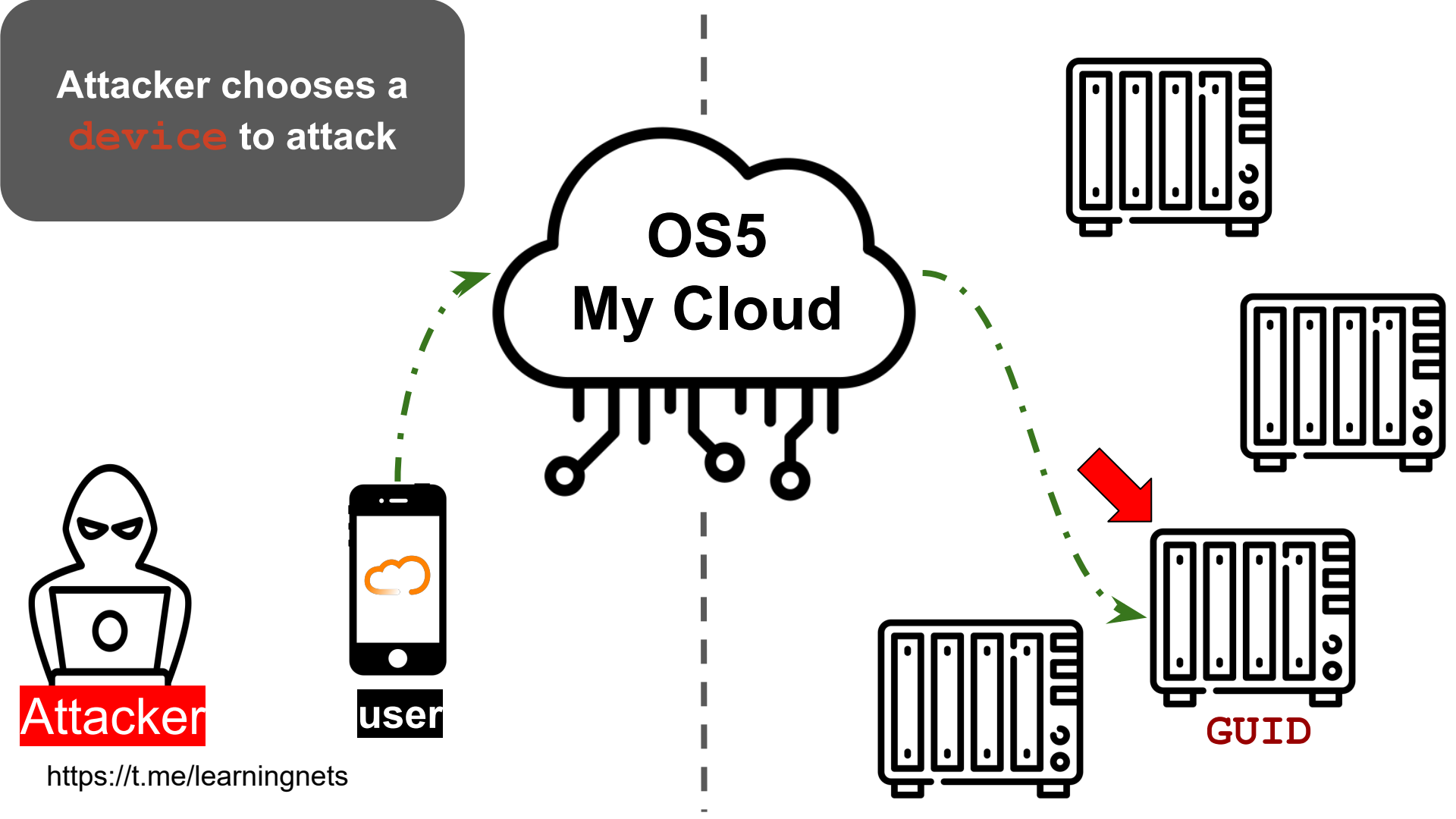
Leveraging Cloud Access to Achieve RCE

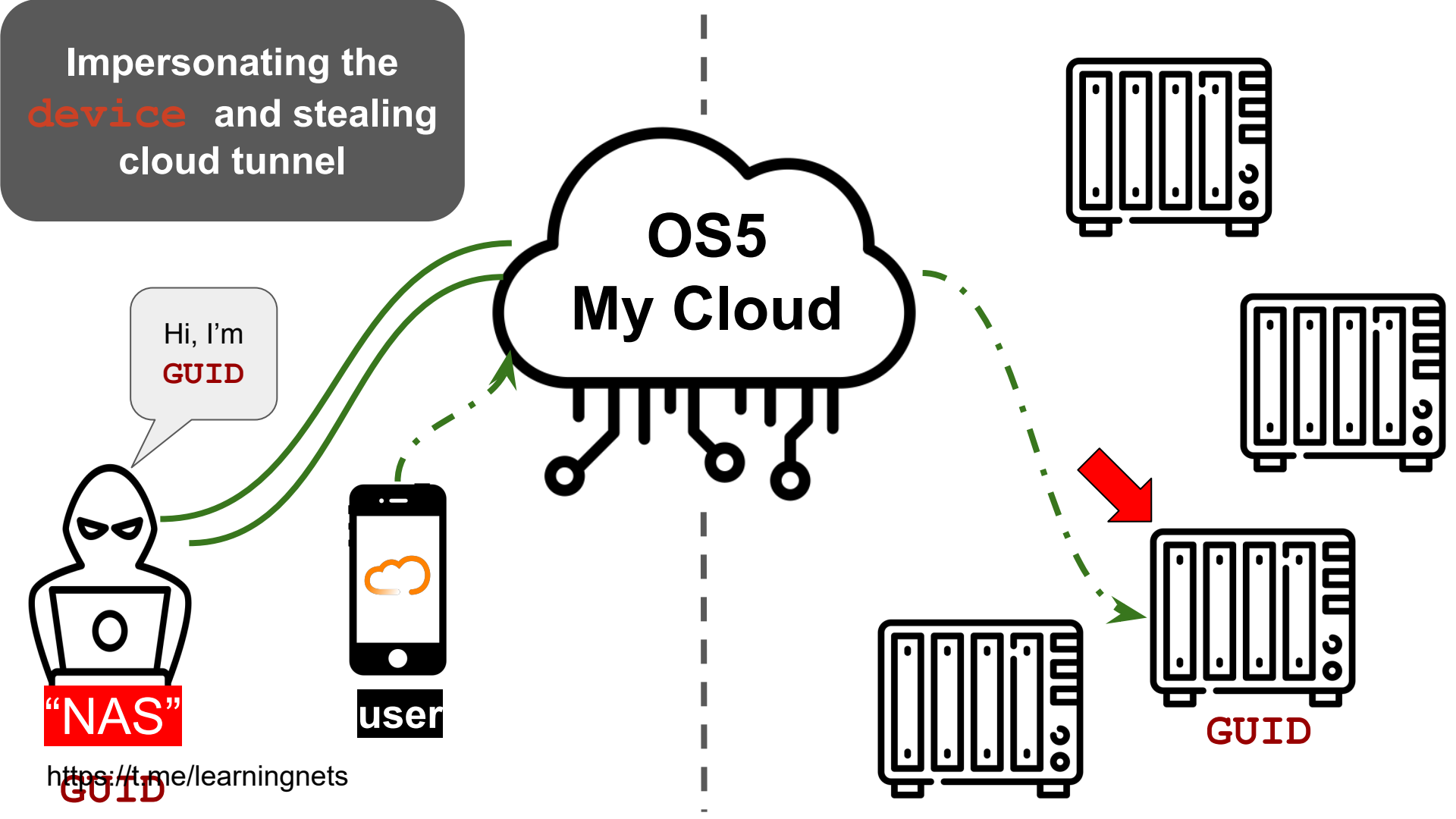
The Full Exploit Chain

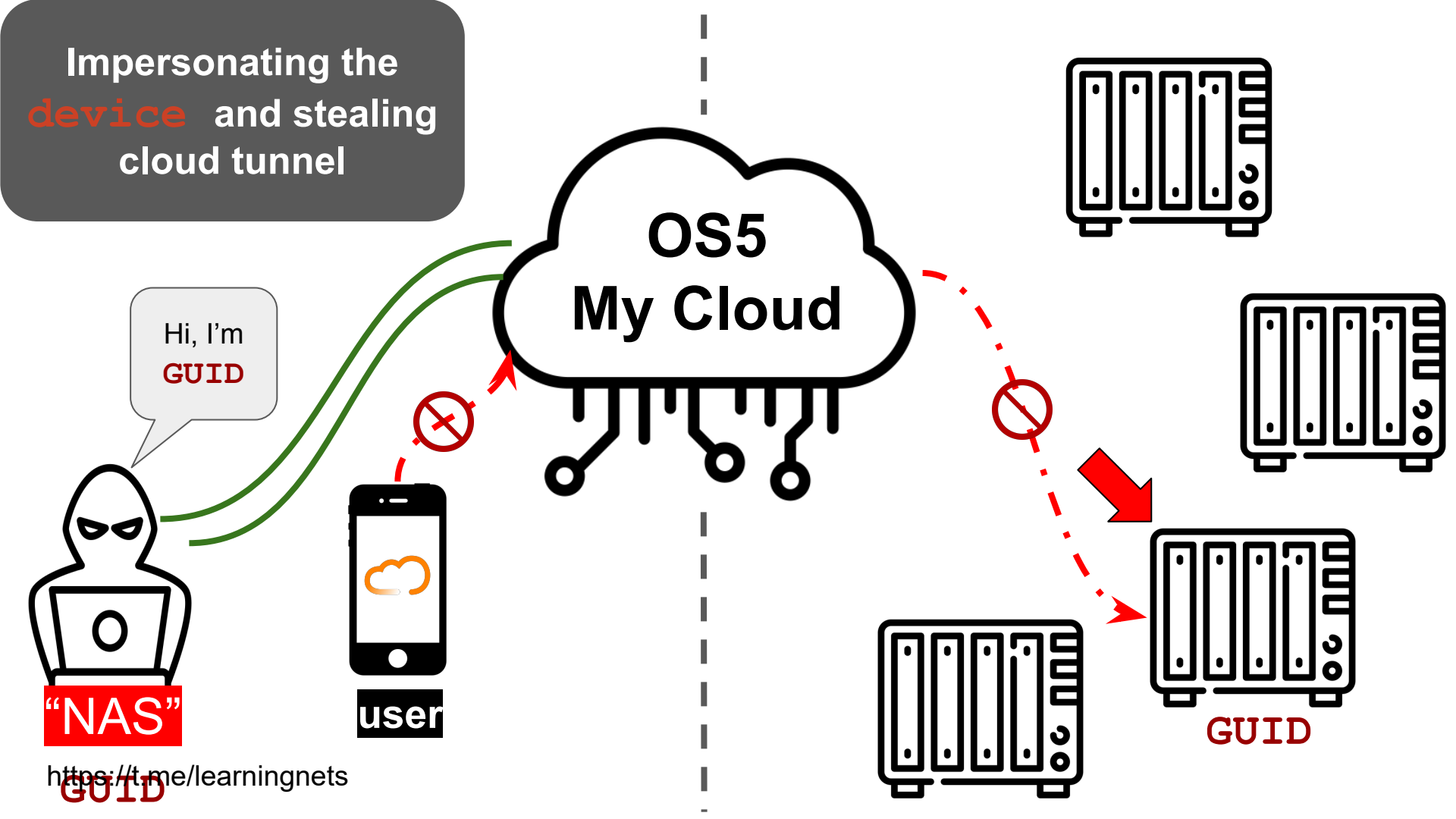
User is connected to their NAS



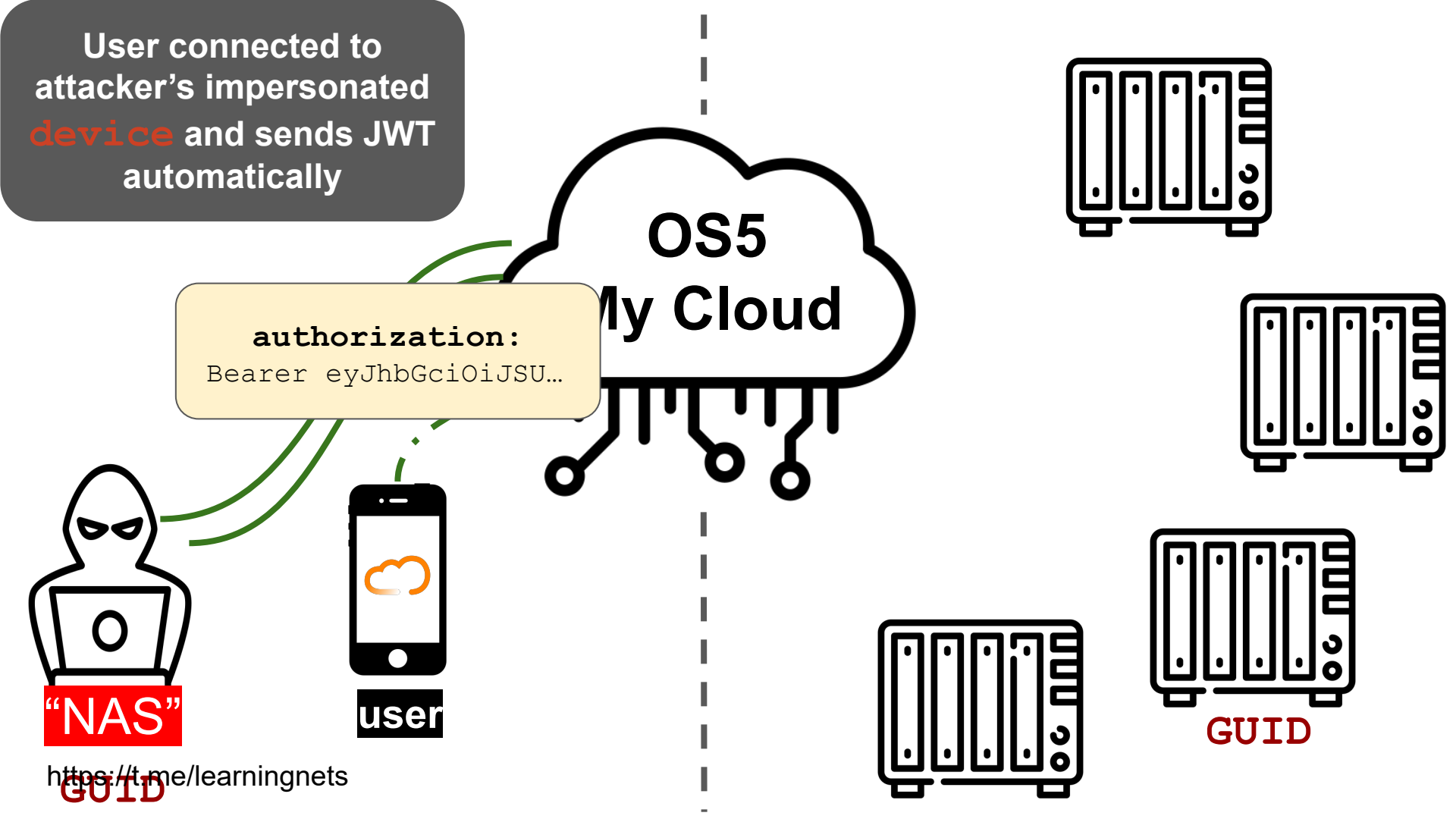
Attacker chooses a `device` to attack




Impersonating the device and stealing cloud tunnel
OS5
My Cloud
Hi, I'm
GUID
"NAS"
user
GUID
GUID
https://t.me/learningnets


Impersonating the device and stealing cloud tunnel
OS5 My Cloud
Hi, I'm GUID
"NAS"
user
GUID
GUID
https://t.me/learningnets

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



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

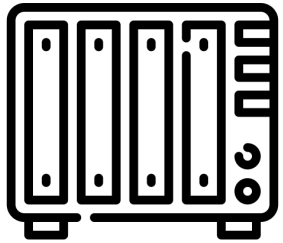
OS5
My Cloud

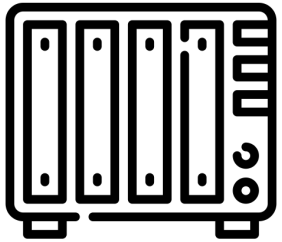
Attacker

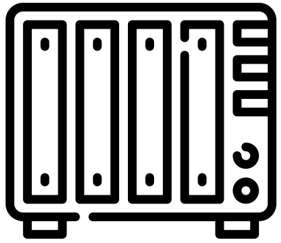
authorization:
`Bearer eyJhbGciOiJSU…`

GUID

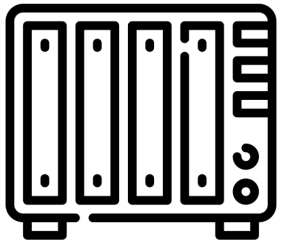
Real device establishes cloud-tunnel again (auto-reconnect)







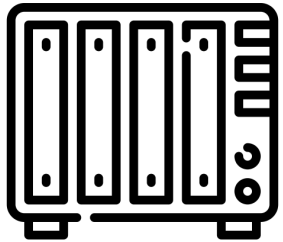
GUID

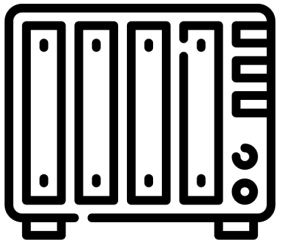


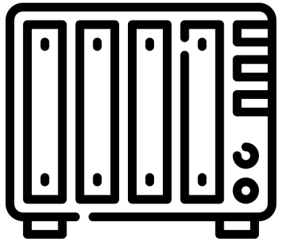
Attacker

authorization:
`Bearer eyJhbGciOiJSU…`

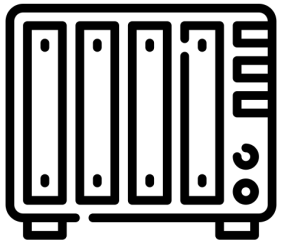
Real device establishes cloud-tunnel again (auto-reconnect)







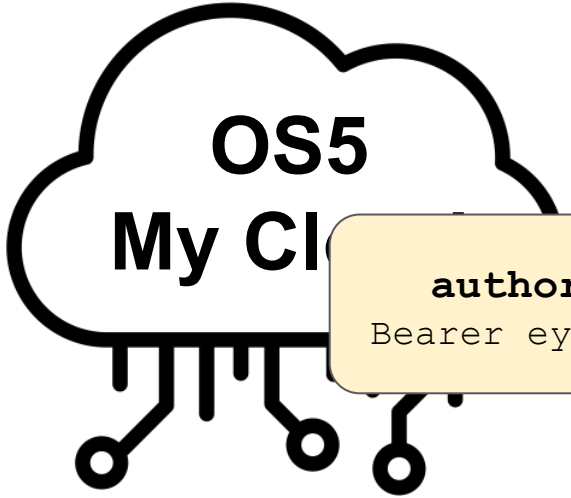
GUID



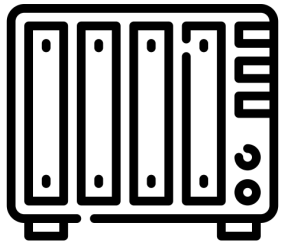
Attacker

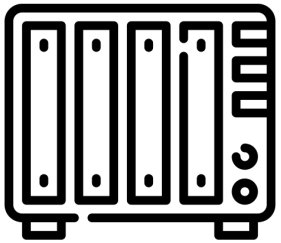
authorization:
`Bearer eyJhbGciOiJSU…`

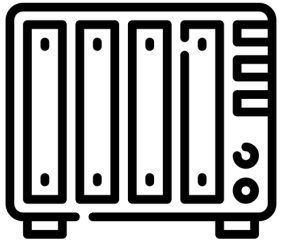
Attacker connects to the `device` using admin's stolen JWT



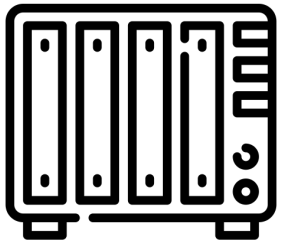
authorization:
Bearer eyJhbGciOiJSU…





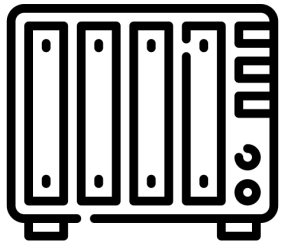


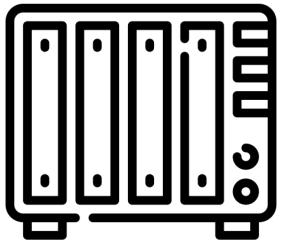
GUID

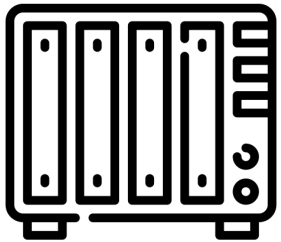


Attacker

Create a new share under `/tmp`



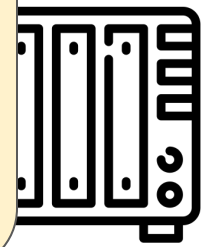




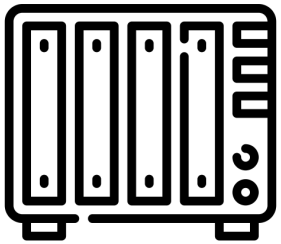
GUID

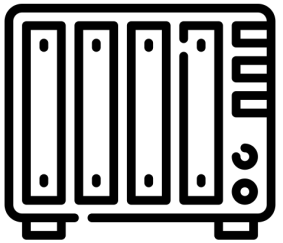
Attacker

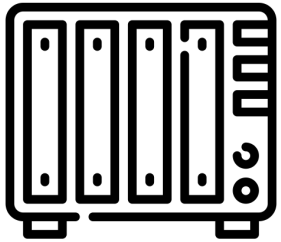
```
POST
/sdk/v1/filesystems
{
 name: "priv_pics",
 path: "/tmp"
}
```



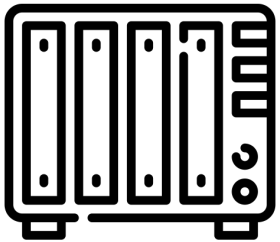
Write a reverse shell to
`/tmp/upload_fw_success`
to be injected during reboot



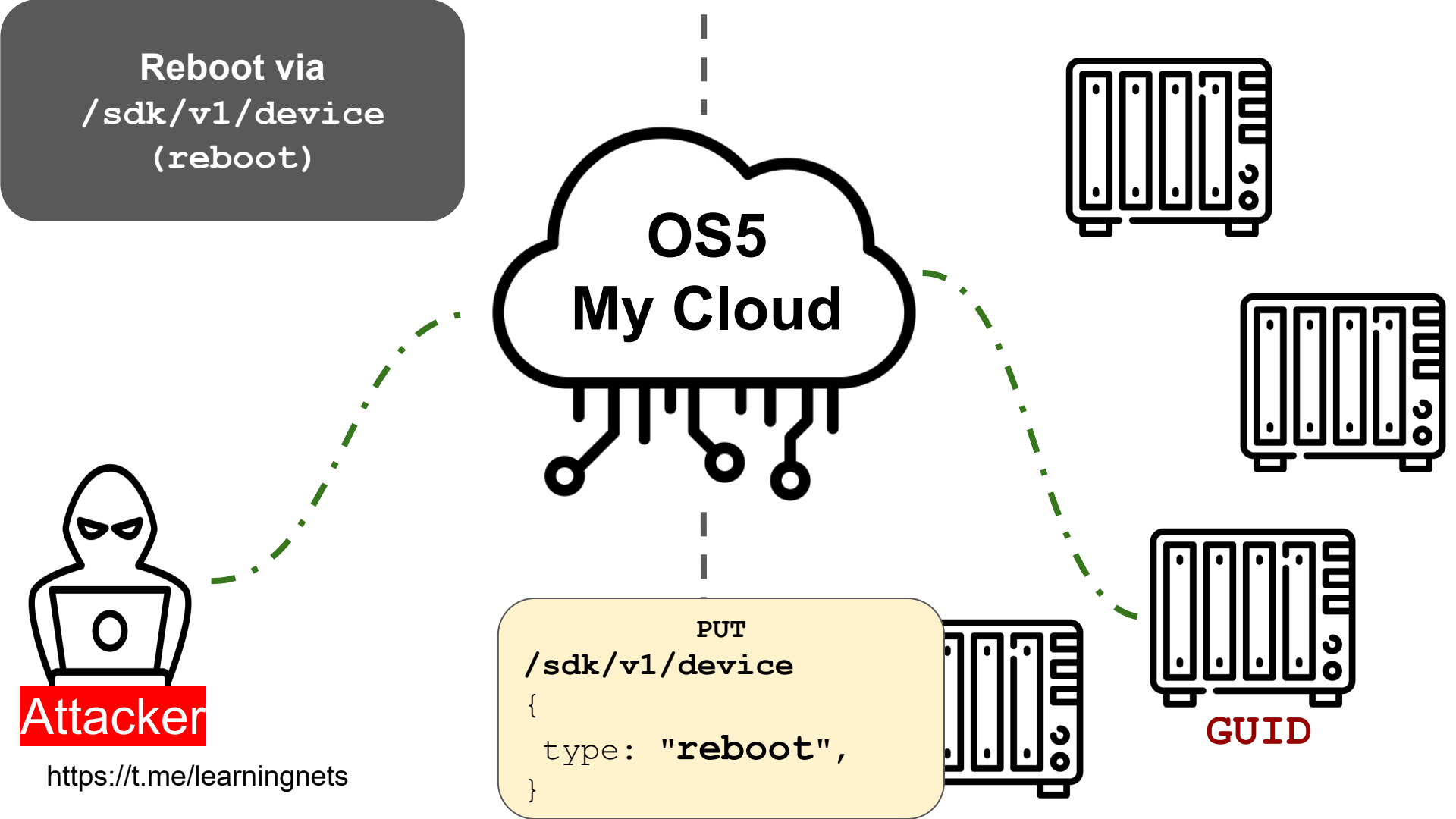




GUID

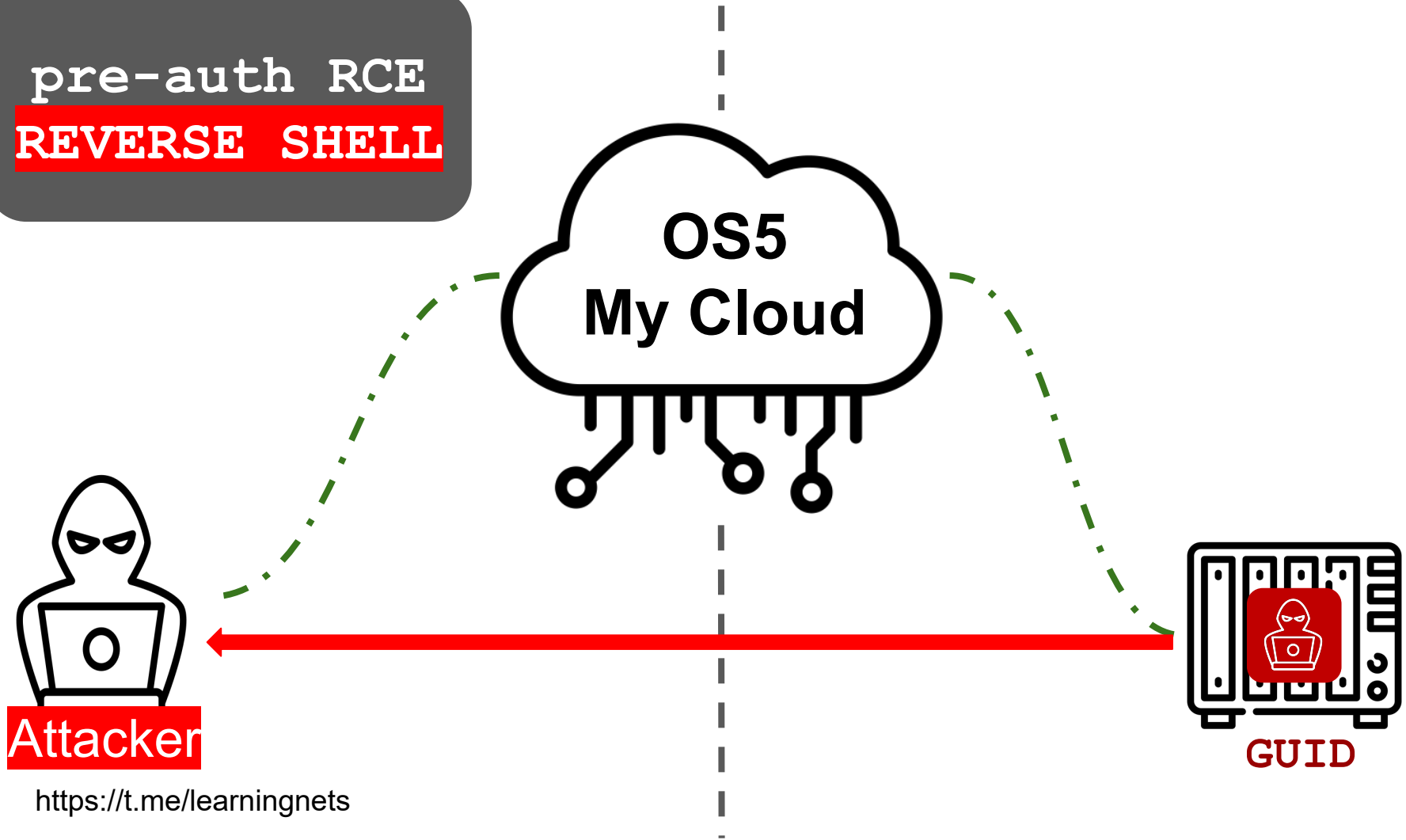


Attacker



pre-auth RCE

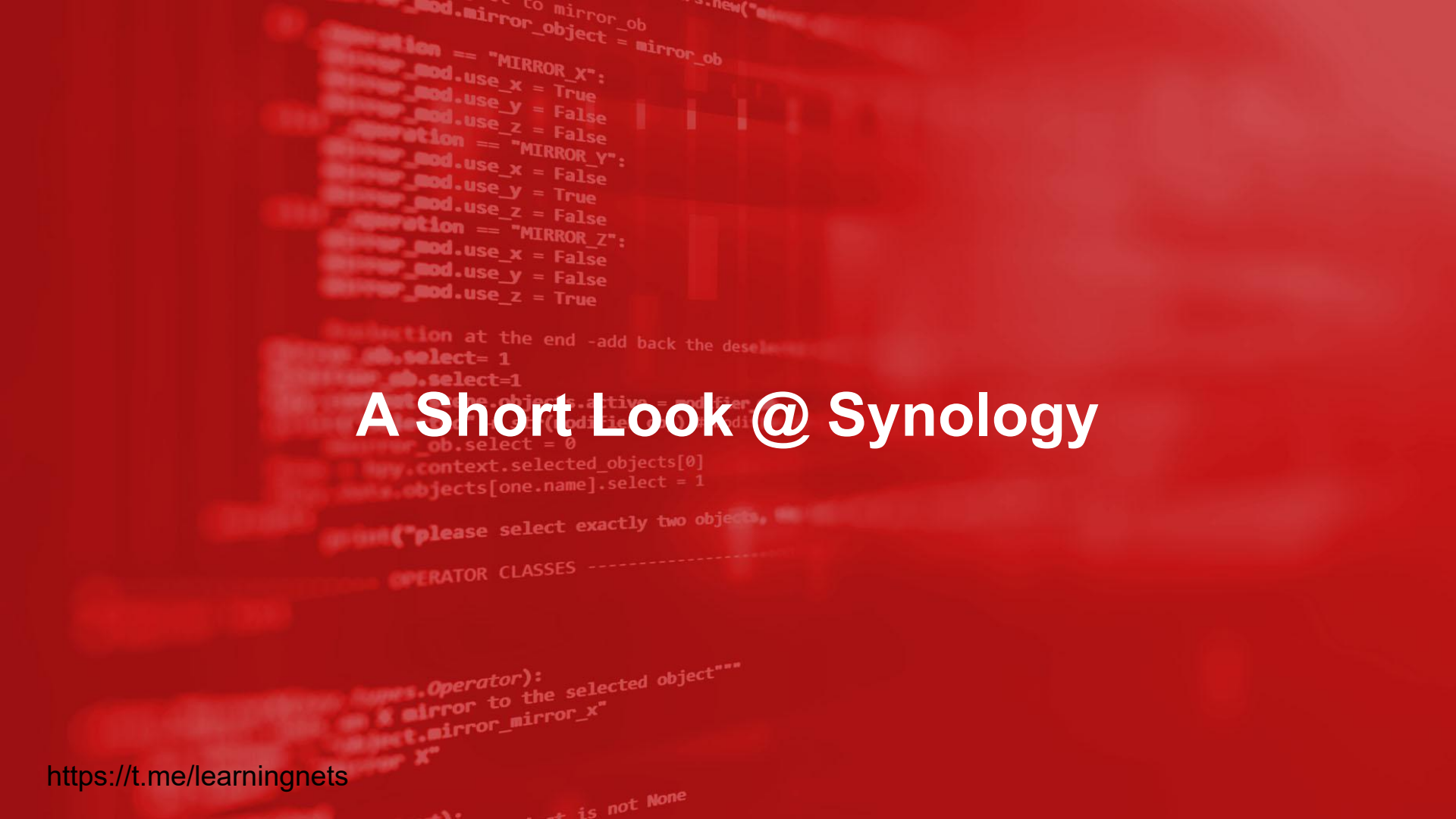
REVERSE SHELL



```
➜  ~ nc -lvvk 1234


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

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

Attacker

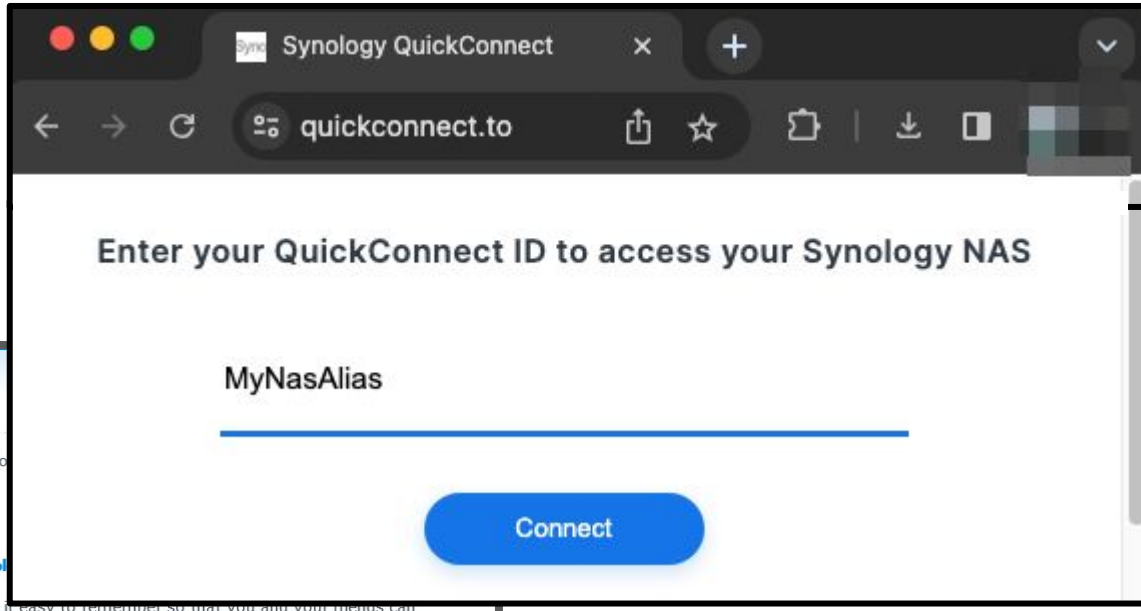
GUID

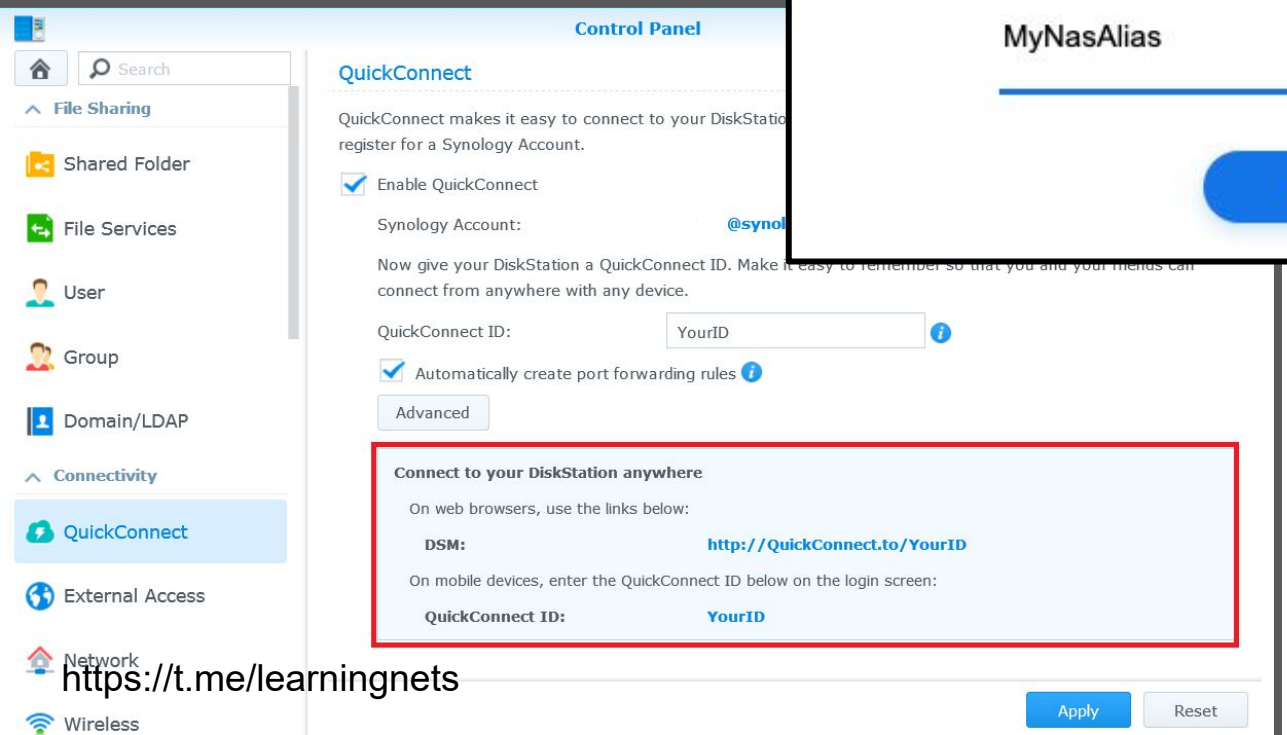


Synology DS920+

- 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`**

alias.quickconnect.to

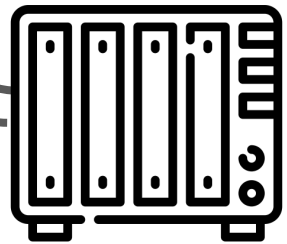




https://MyNas.quickconnect.to

Synology
QuickConnect

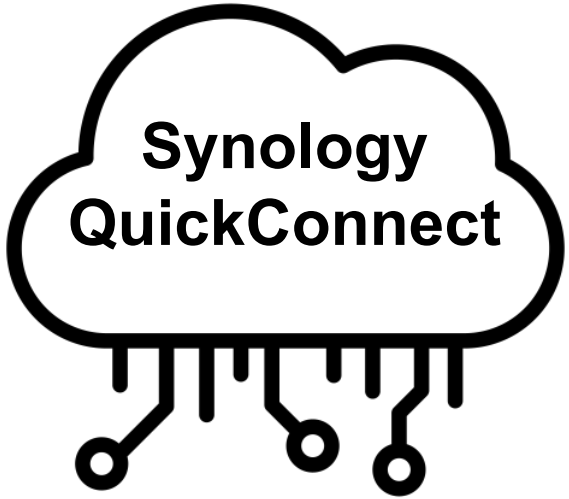
User



MyNas

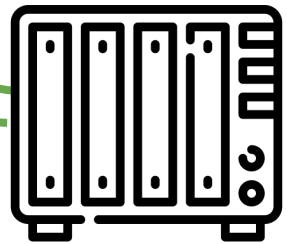
GET
https://MyNas.quickconnect.to

https://MyNas.quickconnect.to



Tunnel using OpenVPN

User



MyNas

GET
https://MyNas.quickconnect.to

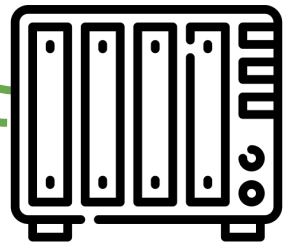
https://MyNas.quickconnect.to

Yes, we can collect all subdomain..

Synology QuickConnect

Tunnel using OpenVPN

User

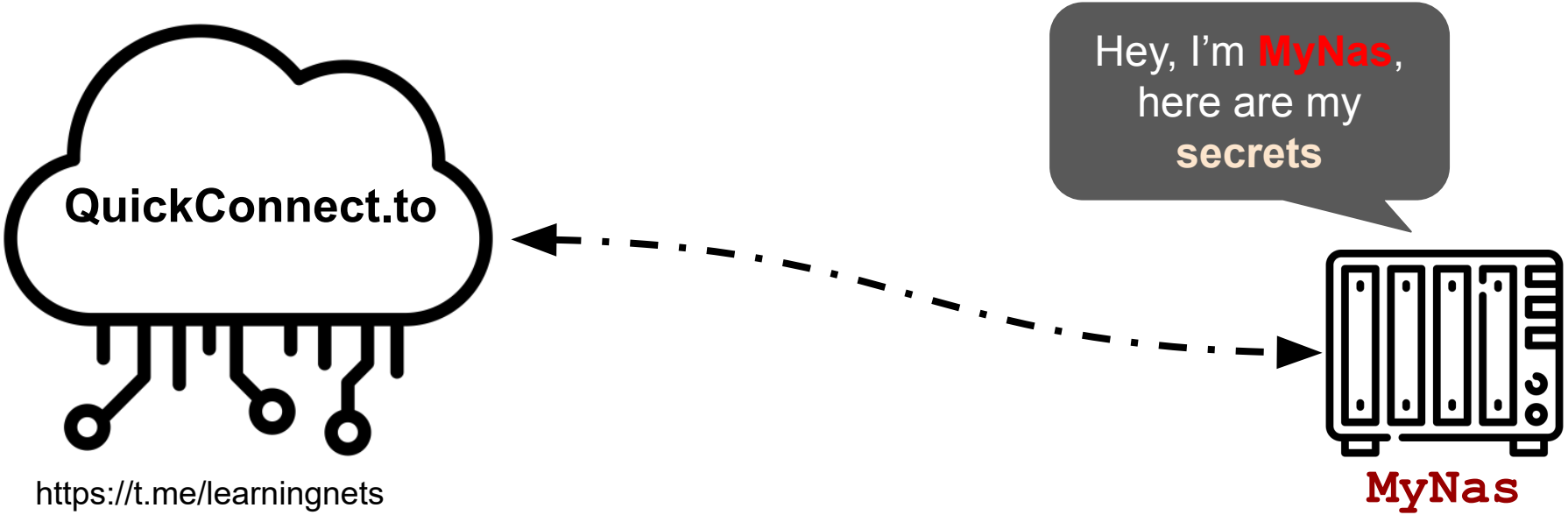


MyNas

GET
https://MyNas.quickconnect.to

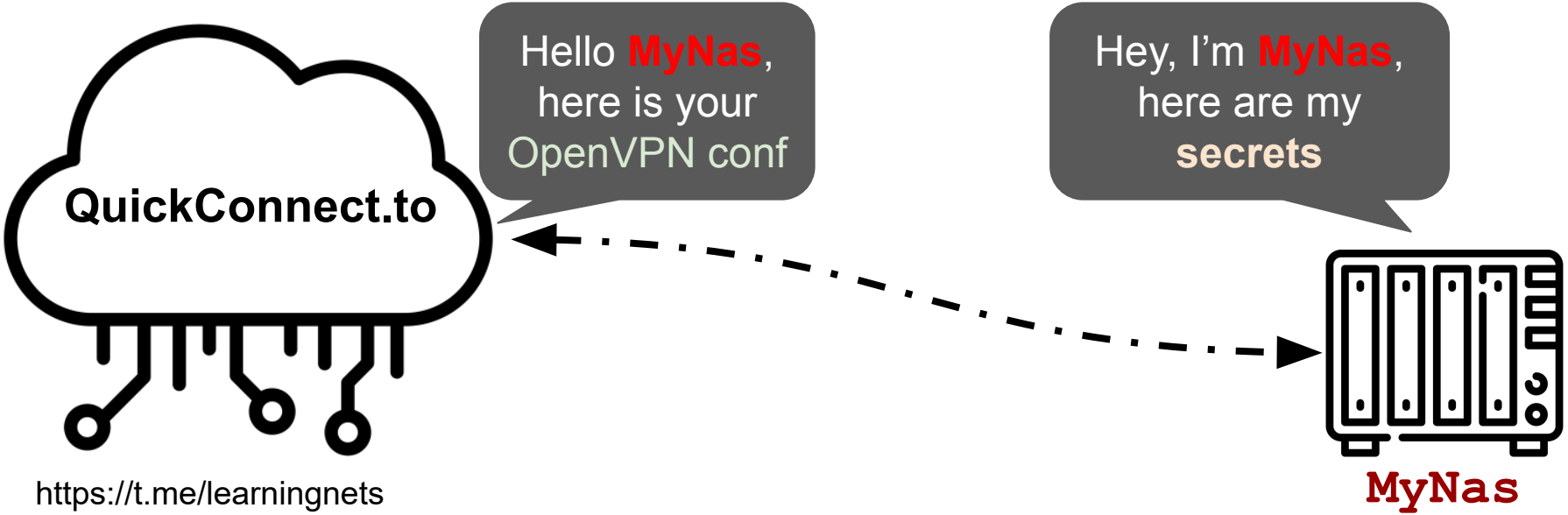
Device Authentication

- 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

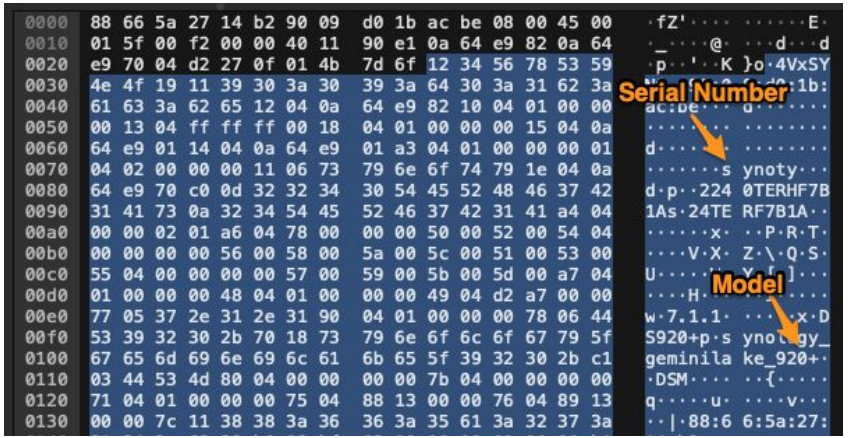


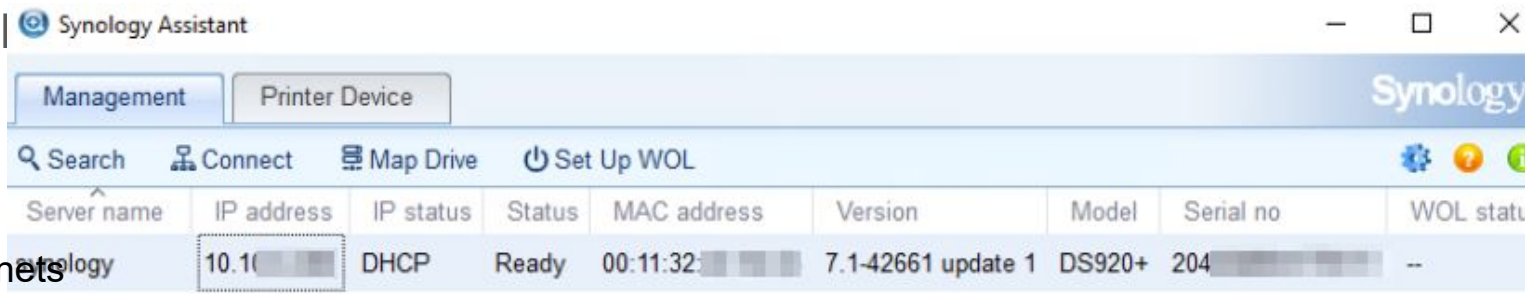
Synology Device Identifiers

- 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



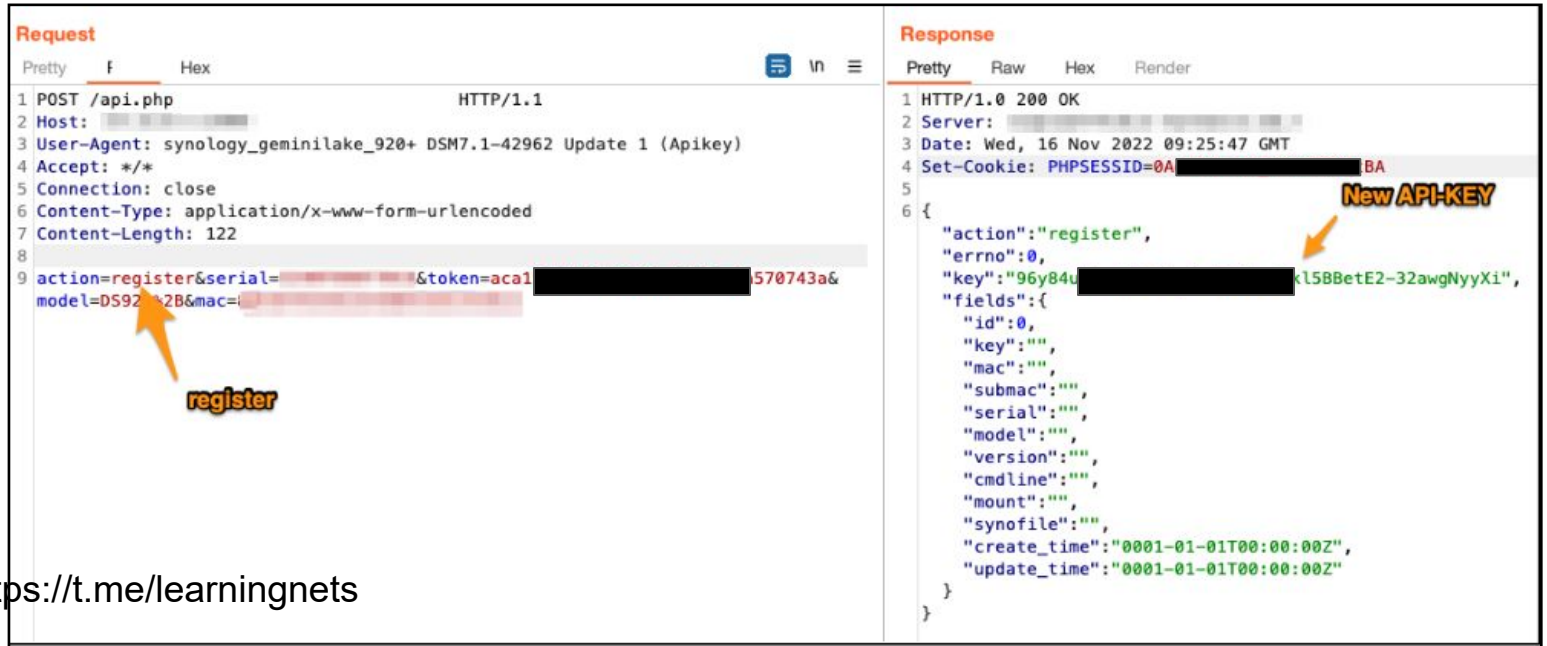


Synology Device Identifiers

- 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



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

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)

Synology Device Identifiers

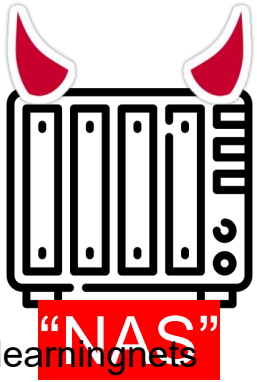
- So what secret do we need to impersonate Synology device?
  - **MAC Address - `90:09:d0:01:23:45`**
  - **Serial Number - `2230FG45A0`**
  - **Device Model - `DS920+`**
  - **DS Token** - Deterministically generated from the serial number ***`md5_rec(2230FG45A0)`***
  - **API-KEY** - Device's QuickConnect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

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 :)



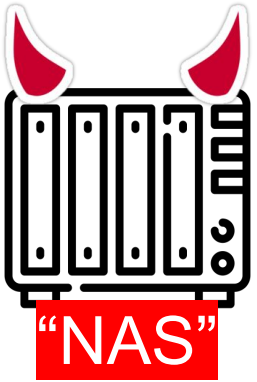

```
[ ] Checking if tokens are available
[ ] Tokens are available! Trying to extract them
[ ] Tokens are available! Trying to extract PATH_TOKEN_TY
PE_COOKIE
[ ] Tokens are available! Trying to extract PATH_TOKEN_TY
PE_SYNOID
[ ] Adding new admin user to the system
[ ] Adding new user: clarotypwn, password: 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)

> 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

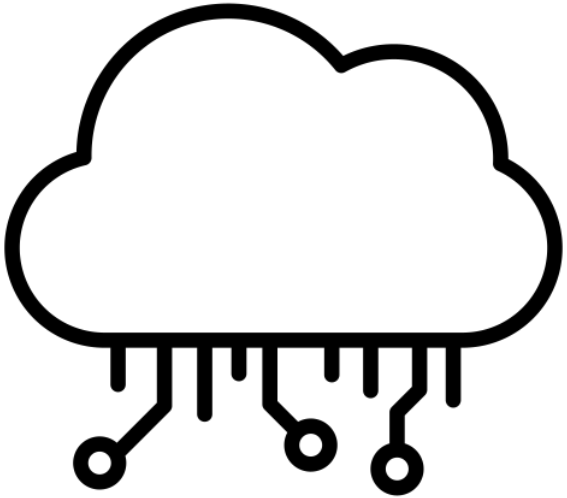
Summary

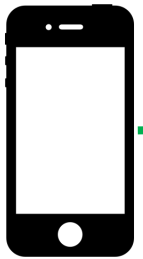
This is what the user wants..

Attacker

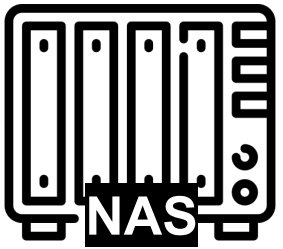


“NAS”





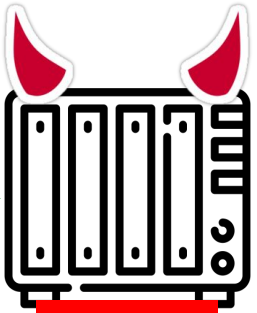
User



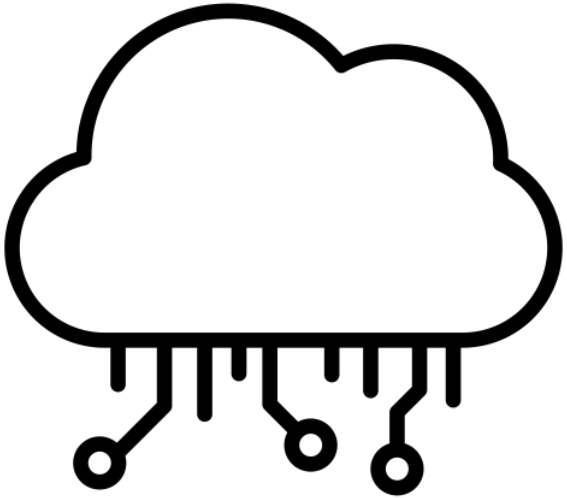
NAS

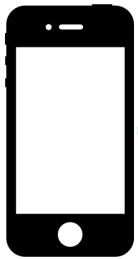
But this is what the user gets..

Attacker

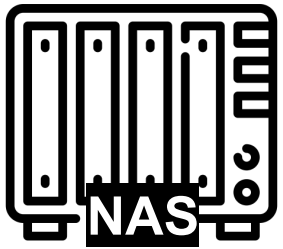


“NAS”

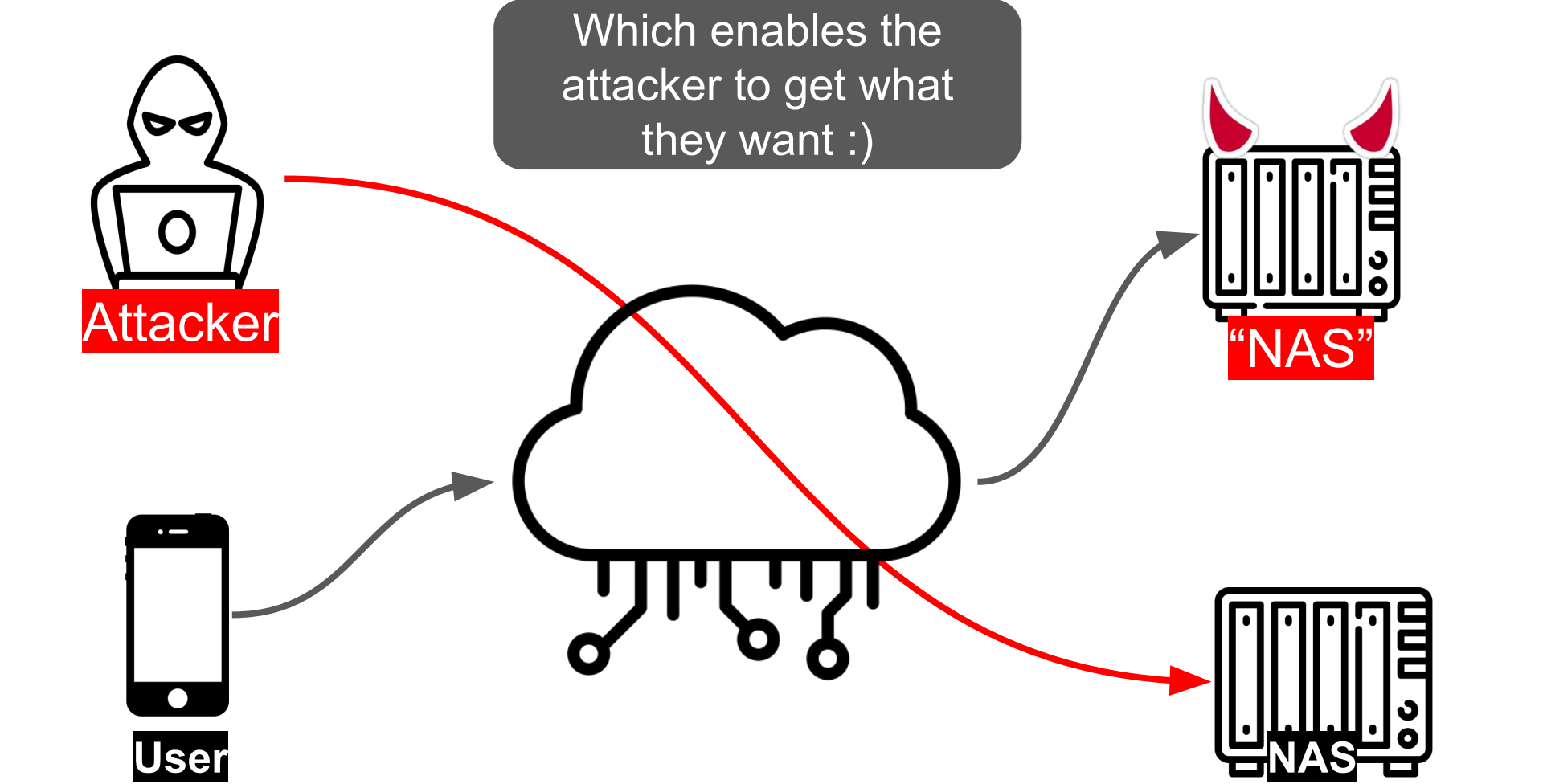




User



NAS



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