Next-Gen Mirai

Balthasar Martin <balthasar@srlabs.de>
Fabian Bräunlein <fabian@srlabs.de>

Security Research Labs

Mirai and IoT Reaper botnets exploited open Telnet and other known vulnerabilities

Mirai botnet

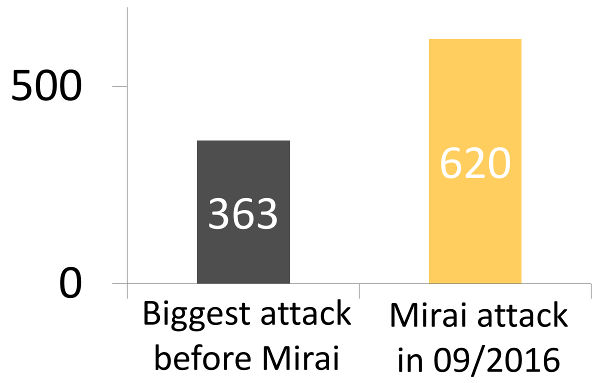
- Open Telnet with default credentials
- 24k devices[1] against Krebs on Security
- Up to 100k[2] devices in attack on Dyn

Probing random IP addresses for exposed devices

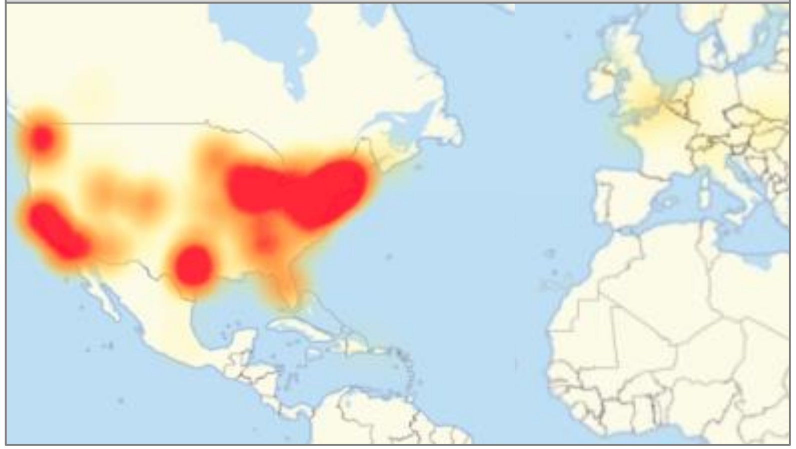
Reaper botnet

- Known vulnerabilities in web interfaces
- 20k devices[3], but way more vulnerable

DDoS against Akamai (Gbps)



Regions affected by attack on Dyn



Not actively used yet

?

[1] https://krebsonsecurity.com/2016/11/akamai-on-the-record-krebsonsecurity-attack/
[2] https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/
[3] https://www.arbornetworks.com/blog/asert/reaper-madness/

Most users thankfully do not expose their home devices to the Internet

- We got an IP camera that can be controlled via App
- Sricam is one of many brands based on Gwell firmwares
- Various vendors sell these devices under their own brands
- Available apps include: Sricam, APcam, Yoosee, 2CU, …

- [x] Video and bidirectional sound
- [x] Remote access from everywhere
- [x] Easy firmware updates

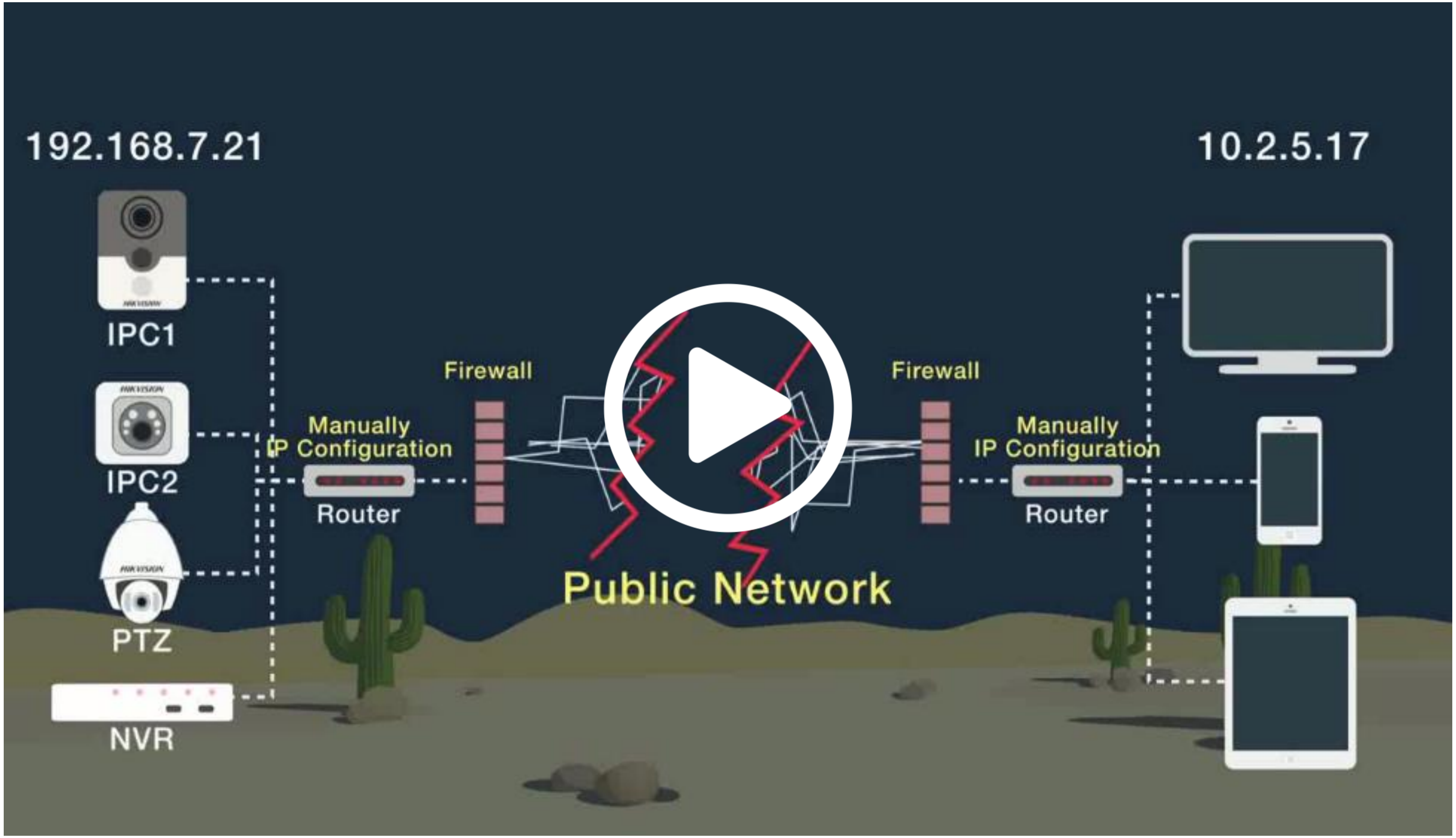
- [ ] Open telnet
- [ ] (Easy) command injection
- [ ] Web interface

- Most users will not expose their devices to the internet anyways

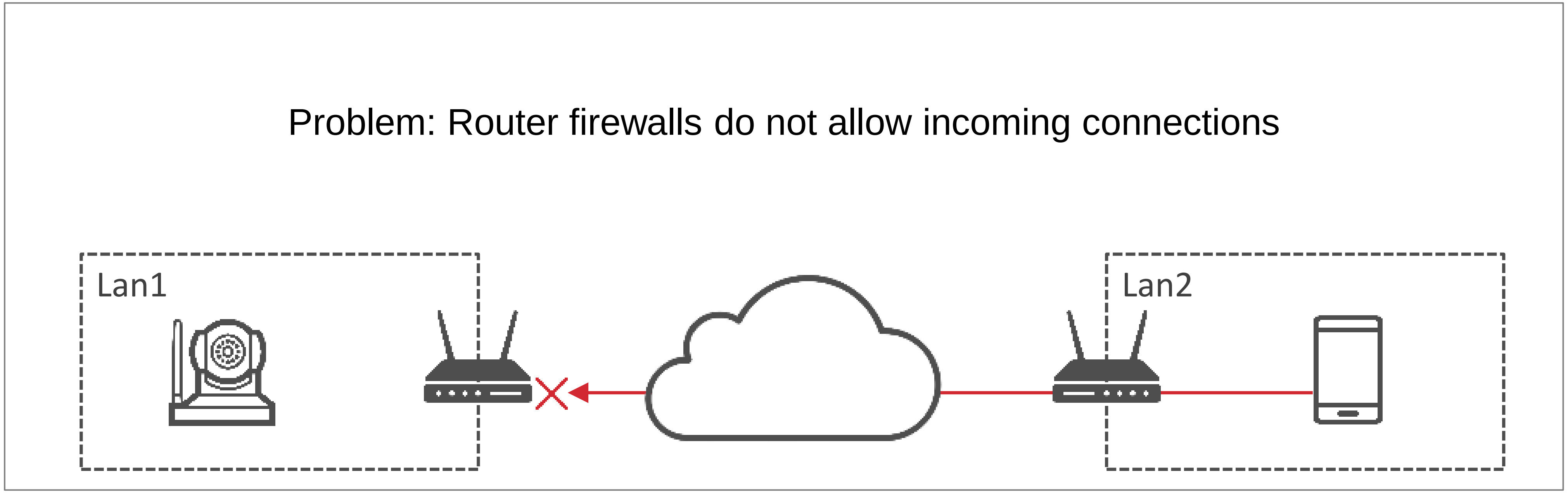
We are able to send packets to millions of devices in private networks and control 800,000 of them remotely – How this was done is the topic of this talk

Penetrating private networks is sold as a feature

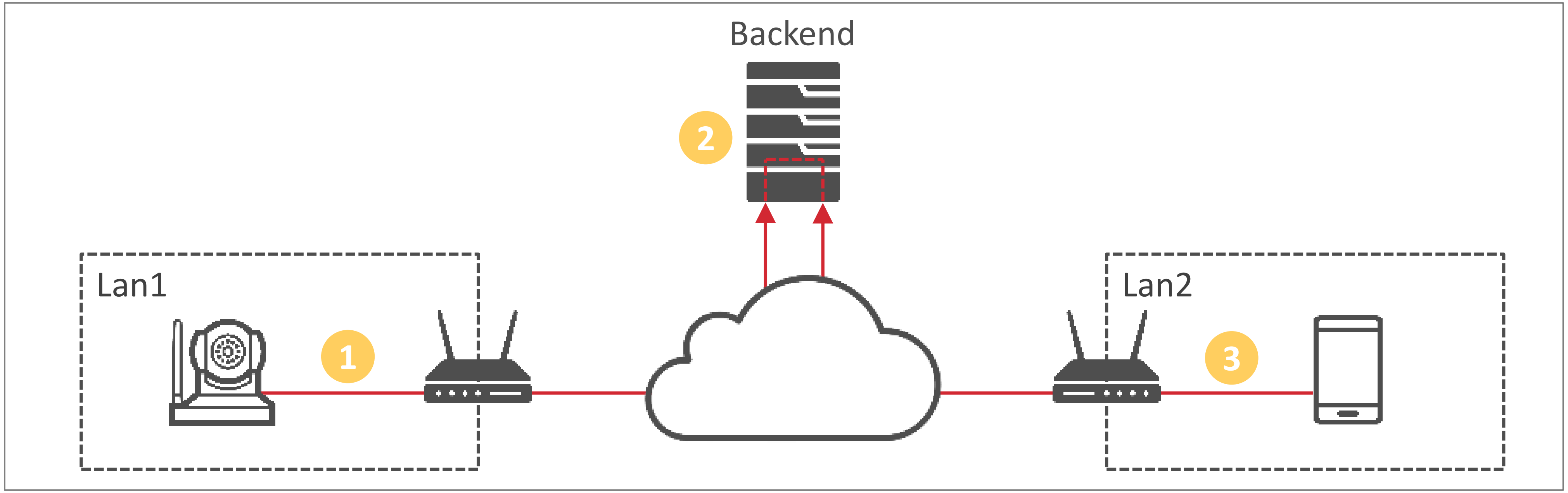
Vendor marketing video:



Proprietary cloud protocols bypass firewalls and allow for remote connections into private networks



Proprietary cloud protocols bypass firewalls and allow for remote connections into private networks



1. IP camera sends UDP packets to keep the NAT-table entry alive
2. Backend server can reach the device when needed
3. Control packets from app are forwarded by the backend*

Let's take a look at:

- videoipcamera.com / videoipcamera.cn
- cloud-links.net / cloudlinks.cn

*for transmitting video feeds, the backend negotiates a direct connection to the device

For building a botnet, we need connection, authentication and remote code execution

Connection

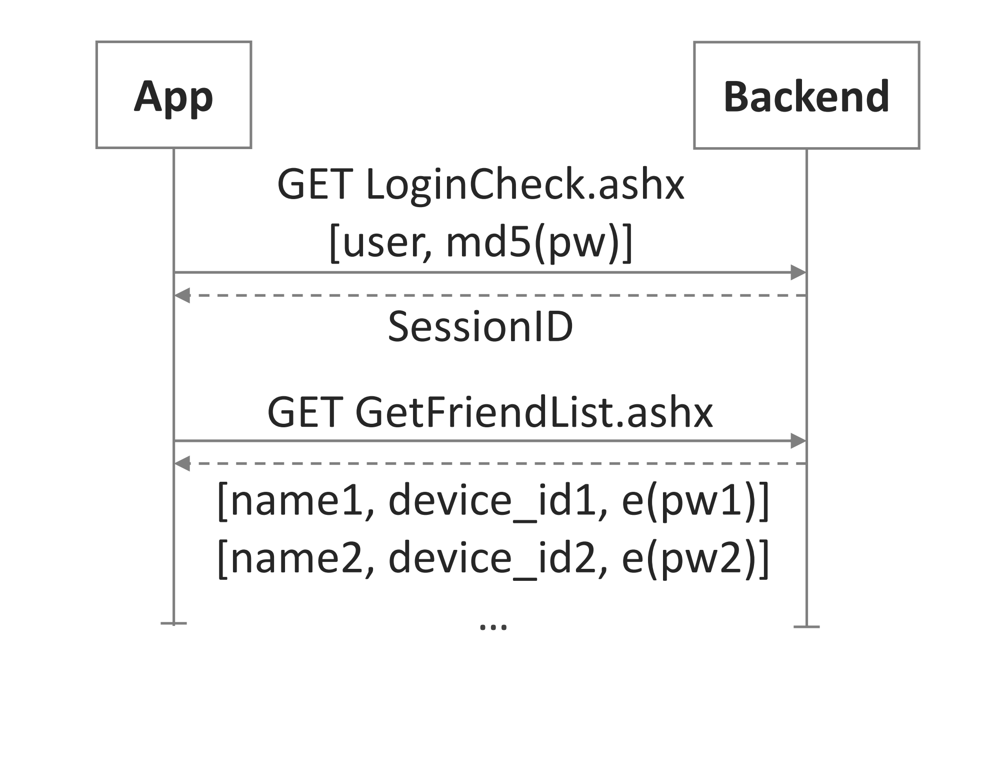
Authentication (-bypass)

Remote code execution

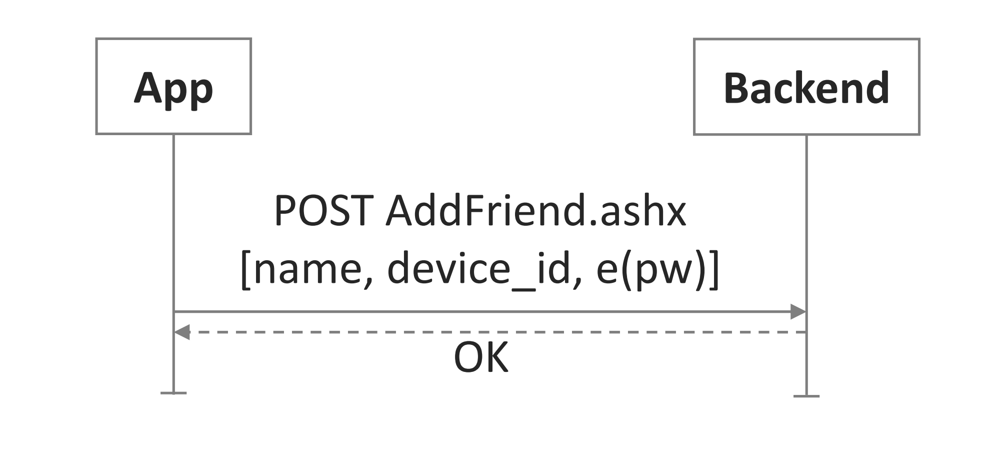
The backend acts as a contact storage

HTTP requests containing contact details

Logging in



Adding a device to an account

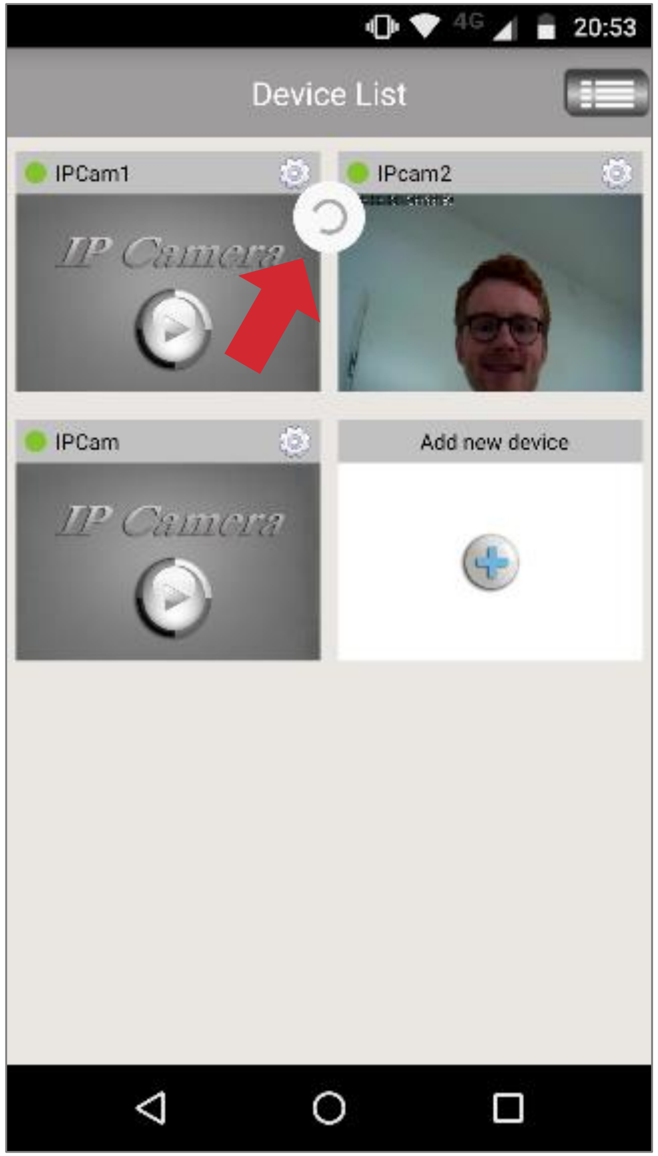


In a secure world…

… this would be the only way to check device credentials

… requests would be monitored and rate limited

In reality, all valid device IDs can be easily retrieved from the backend



UDP packet to check which devices are online

Request

```
28 00 04 00 00 00 00 00 b8 65 6d b7 66 d4 a1 ae
57 cd 73 ca 03 00 00 00 06 00 00 00 00 00 00 00
0f 00 00 00 00 00 00 00 XX XX 0a 00 XX XX 0c 00
XX XX 09 00
```

Response

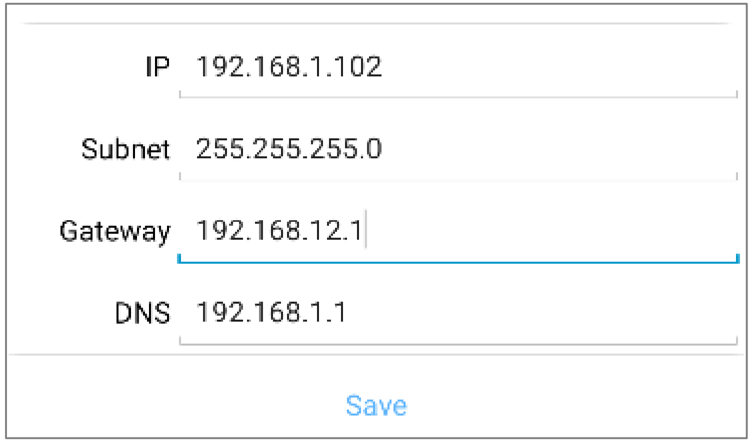
```
29 00 00 00 03 00 00 00 06 00 00 00 00 00 00 00
0f 00 00 00 00 00 00 00 XX XX 0a 00 00 00 00 00
01 00 00 00 00 00 00 07 XX XX 0c 00 00 00 00 00
00 00 00 00 00 00 00 07 XX XX 09 00 00 00 00 00
01 00 00 00 00 00 00 07
```

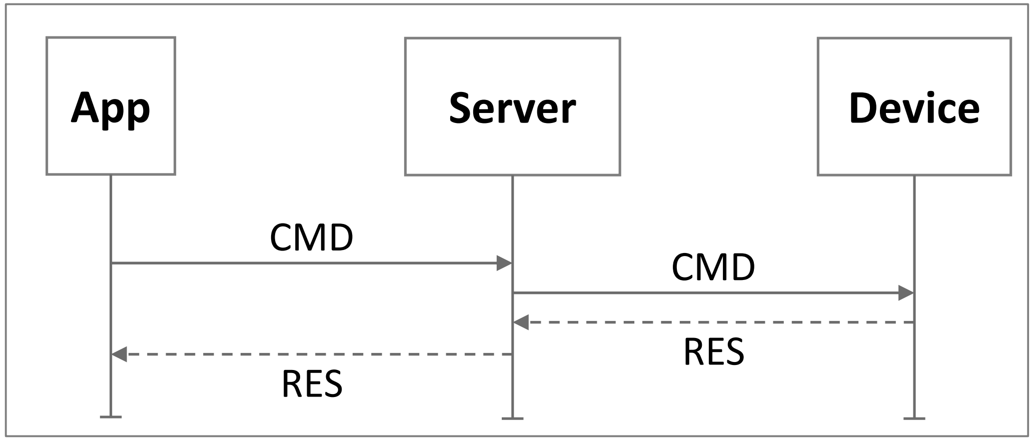
Header | No. devices | Device IDs | Online status

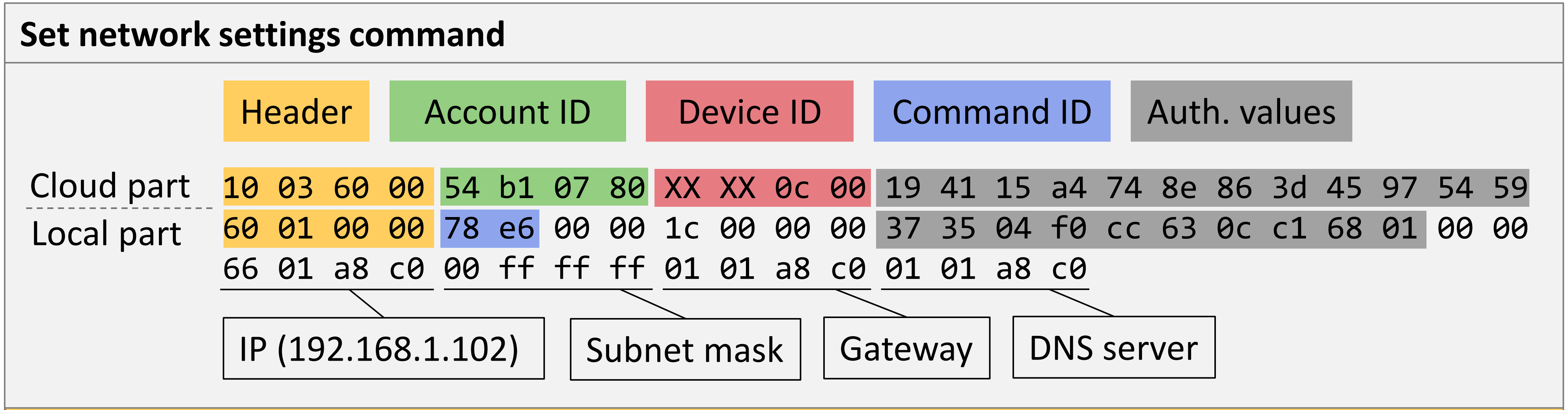
- Does not require authentication
- 62 device IDs in one UDP packet
- No rate limiting
- **Check all possible IDs in 1 hour**

Backend	Dev. ID length	Collected IDs
videoipcam	6 digits	**140,741**
cloudlinks	7 digits	**3,277,280**

The backend forwards command packets based on the device ID







- **Some types of commands are forwarded to the device just based on device ID**
- **Potential for pre-auth RCE → exploiting all devices in just hours**

We have found a large number of devices – now we need to authenticate

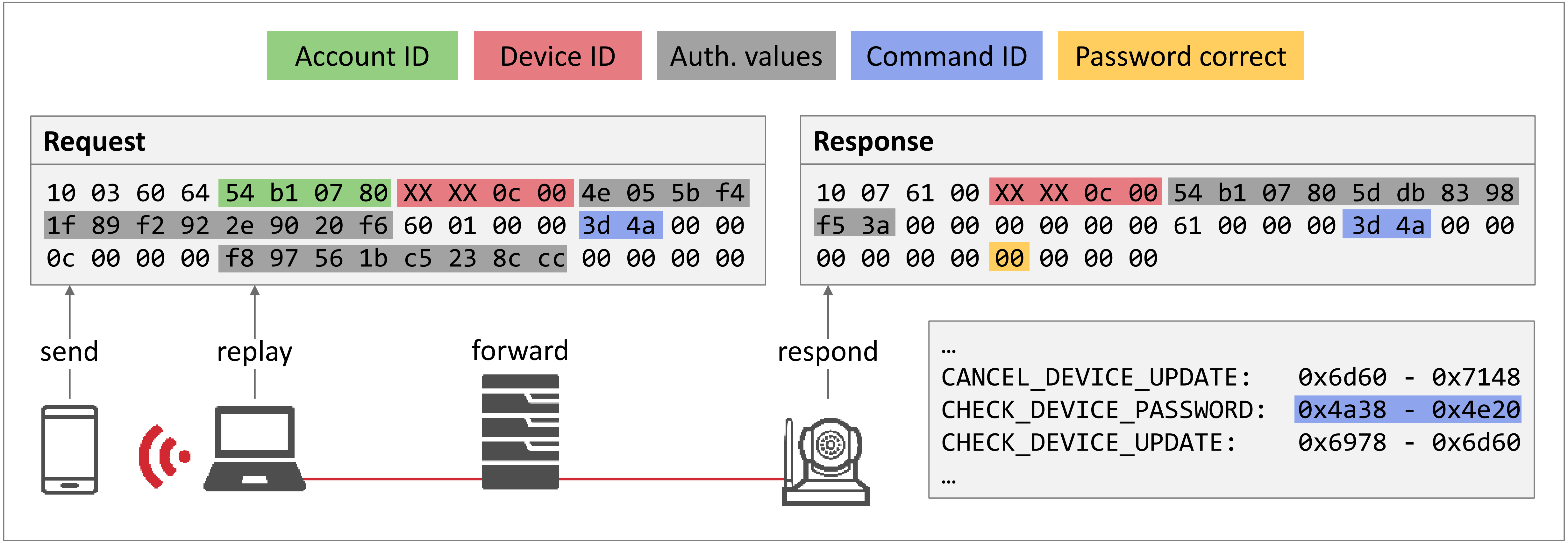
Connection

- Low entropy device IDs allow for efficient enumeration
- Packets are forwarded to devices just based on device ID

Authentication (-bypass)

Remote code execution

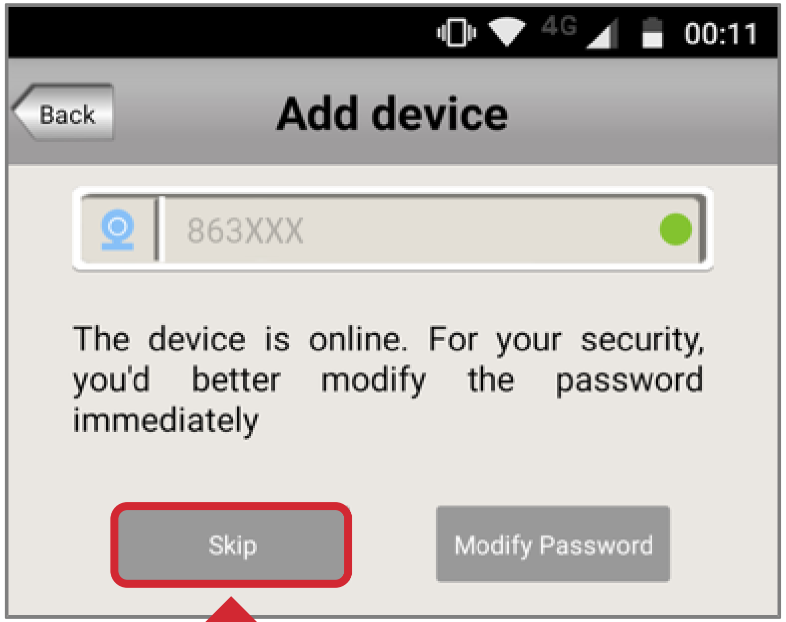
Device passwords can be efficiently enumerated



- When accessing device settings via app, a check-password UDP packet is sent
- It can be captured and replayed with a different device ID to check it for the same password
- The device does not have to be added to the account and no rate limiting is employed

Enumerating weak and default passwords yields access to large numbers of devices

- Devices are using different default passwords: 888888, 123, ...
- Users will choose bad passwords anyway: 123456, ABCDEF, ...
- On videoipcamera, we encountered no rate limiting
- For cloudlinks, the app presented us a client side CAPTCHA
- We did not test the limits and checked 140,000 devices in 6 hours



Incredibly tempting button

Backend	Password	No. devices
Videoipcam	888888	**63,029**
Videoipcam	123456	**1,454**
Cloudlinks	123	**703,000***
Cloudlinks	123456	**46,600***
Total		**814,083***

- View camera feeds, turn devices, hear and send audio
- Get WiFi credentials, near network names, mail credentials
- Access and change device settings

*estimates based on a random 1,000 devices sample

Demo: Enumerating device IDs and passwords

We can access a large number of devices – now we need to execute commands on them

Connection
- Low entropy device IDs allow for efficient enumeration
- Packets are forwarded to devices just based on device ID

Authentication (-bypass)
- Passwords can be enumerated without rate limiting
- Default passwords yield high numbers of devices

Remote code execution

The filesystem in the firmware can be manipulated to add a backdoor

```
$ binwalk npcupg_14.00.00.52.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
32            0x20            JFFS2 filesystem, little endian
2943372       0x2CE98C        ELF, 32-bit LSB executable, ARM, version 1 (SYSV)
```

FW header

JFFS2 filesystem

```
├── dhcp.script
├── gwellipc
├── minihttpd.conf
├── npc
├── upgfile_ok
├── version.txt
└── [...]
```

32-bit ELF binary

```
$ xxd -l 64 npcupg_14.00.00.52.bin

00000000: 0000 0000 6ce9 2c00 211b 0000 397c abbf  ....l.,.!...9|..
00000010: 372a 856a a618 2c6b 0cbc f1a8 3400 000e  7*.j..,k....4...
00000020: 8519 01e0 3300 0000 9611 8be8 0100 0000  ....3...........
00000030: 0000 0000 0200 0000 3e6d 0644 0b08 0000  ........>m.D....
```

On boot, dhcp.script is executed → add malware or open telnet

When installing a modified firmware, “MD5 err!” is printed on serial output

Patching the main camera binary allows for printing the expected firmware checksum

Serial output when installing a firmware

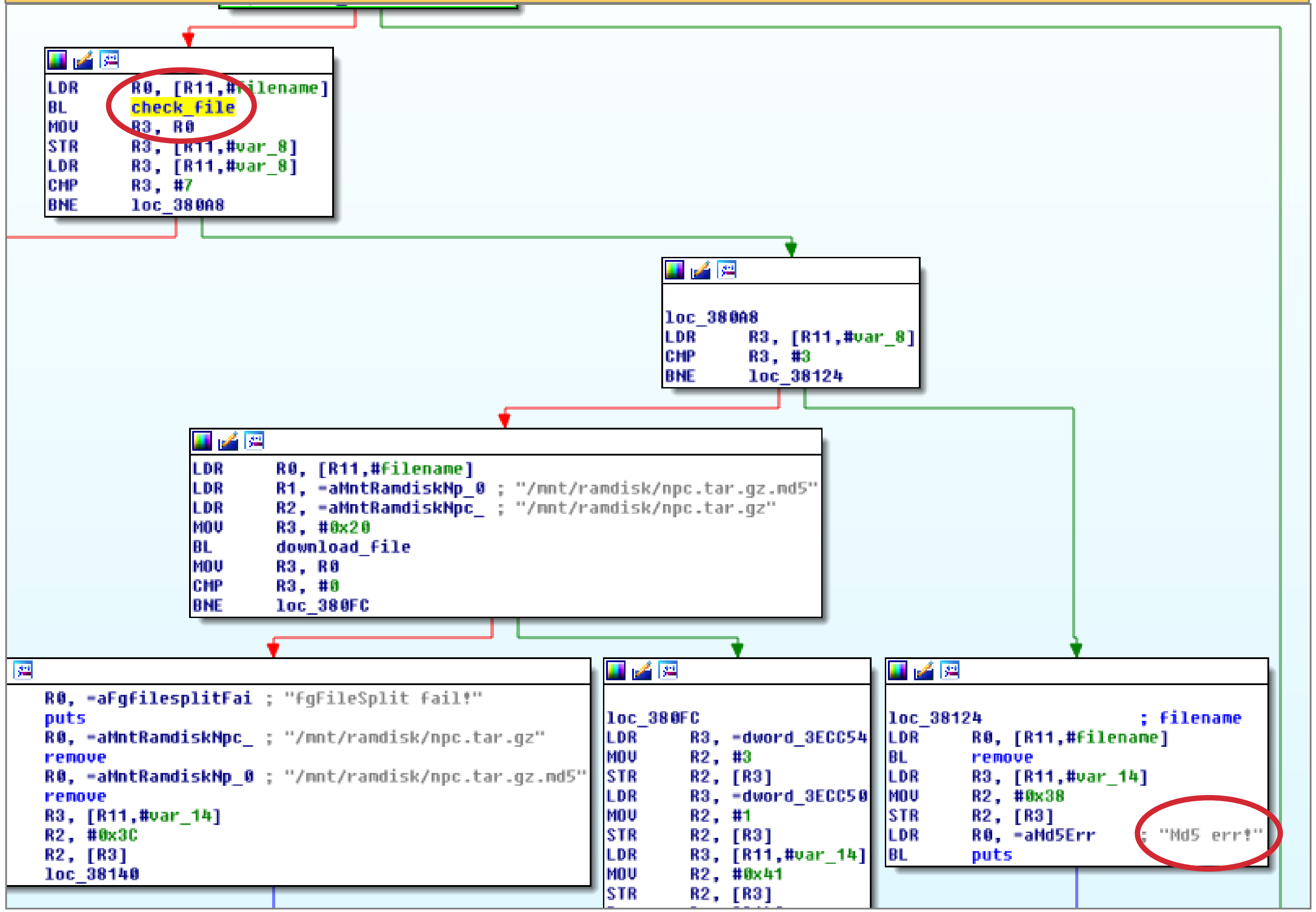
- **Modified file system**

```
Start Seq = 00000d4b
Md5 err!
```

- **Original file system**

```
Start Seq = 00000a99
57 124 171 191 55 42 133 106 166
24 44 107 12 188 241 168
Newst version !
fgCheckUpgFile over!
```

Byte-wise comparison of expected and given hash

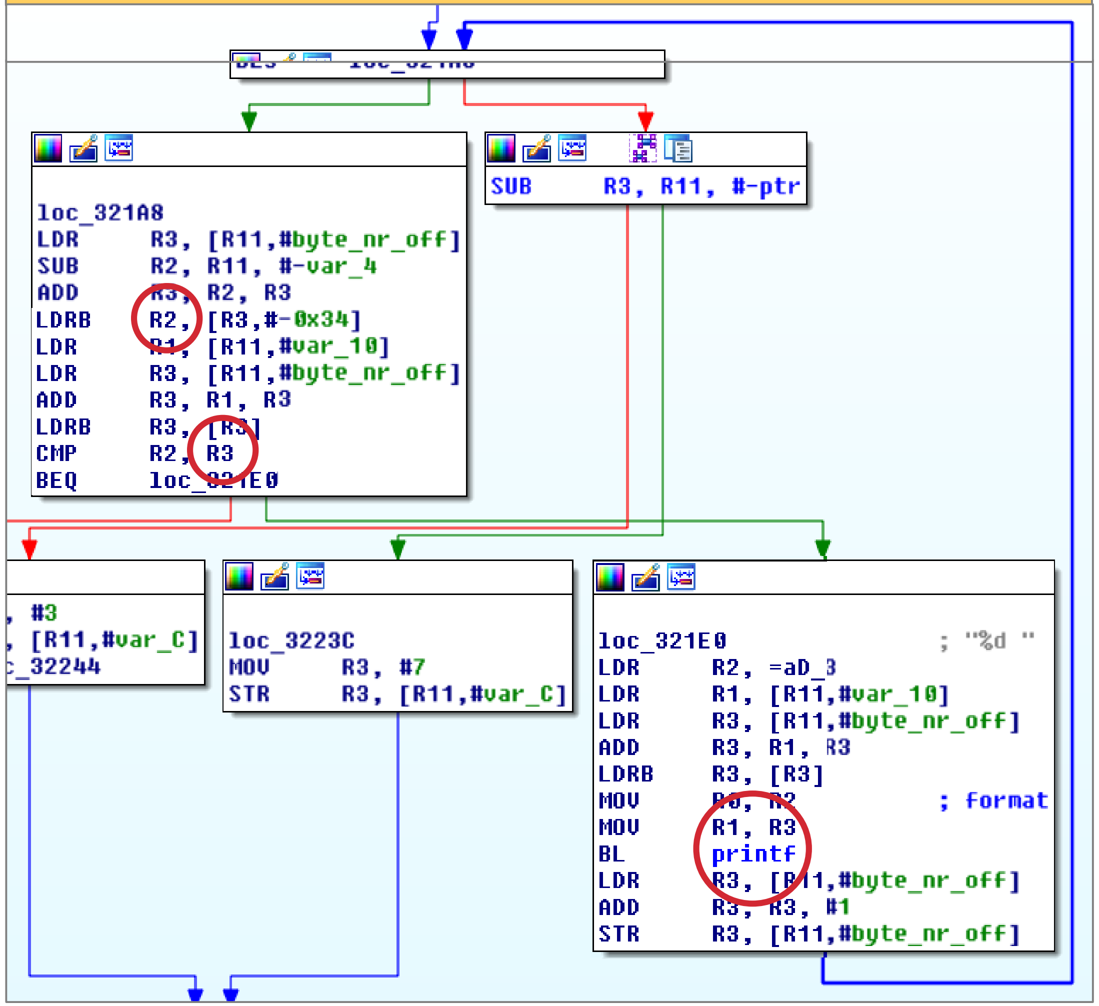


Patching the main camera binary allows for printing the expected firmware checksum

Serial output when installing a firmware

- **Modified file system**
  ```
  Start Seq = 00000d4b
  Md5 err!
  ```
- **Original file system**
  ```
  Start Seq = 00000a99
  57 124 171 191 55 42 133 106 166
  24 44 107 12 188 241 168
  Newst version !
  fgCheckUpgFile over!
  ```

Byte-wise comparison of expected and given hash



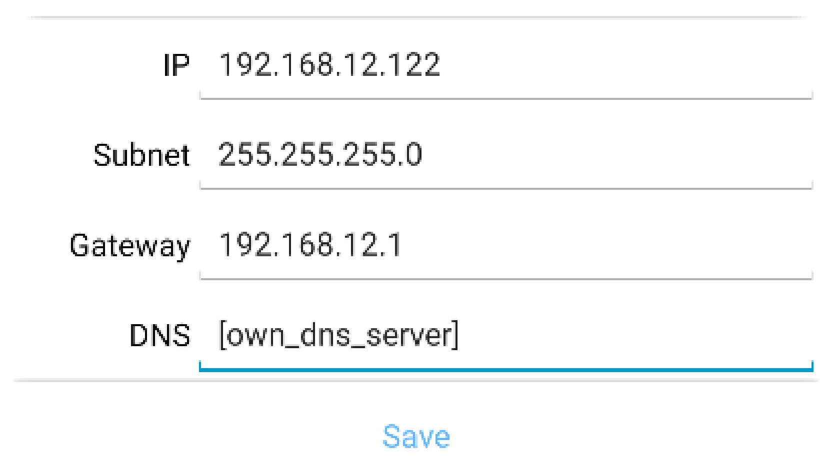
Patch main binary to print expected hash

```
kill -9 [process_number]

printf '\x50' | dd bs=1 seek=172469 of=/npc/npc …
printf '\x02' | dd bs=1 seek=172488 of=/npc/npc …
printf '\x05' | dd bs=1 seek=172536 of=/npc/npc …
```

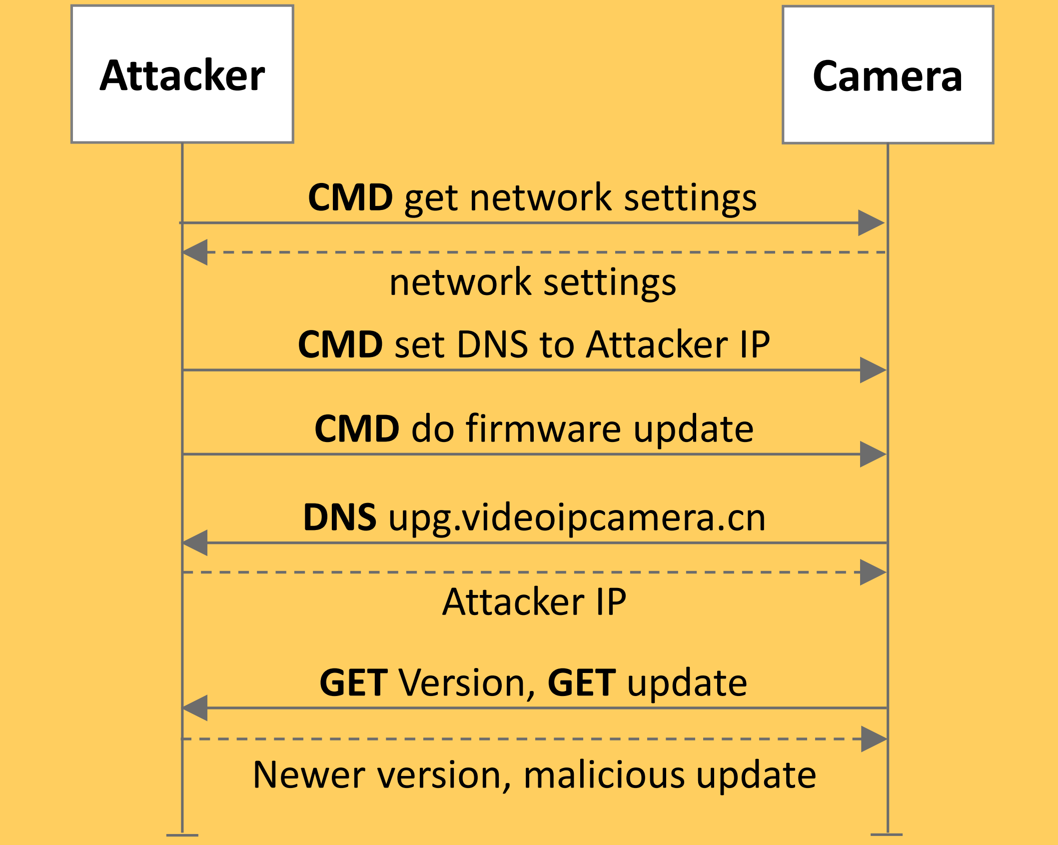
Mass-scale remote installation of malicious firmwares possible by redirecting camera to attacker‘s update server

Remember the network settings packet?



- Two different kinds of firmwares:
  - 14.00.00.XX
  - 21.00.00.XX
- Current version in update request
- Fully automatable procedure

Initiate firmware update and deliver malware



Demo: Installing a malicious firmware remotely via terminal

Infrastructure and protocol design entail a high abuse potential

Connection	▪ Low entropy device IDs allow for efficient enumeration ▪ Packets are forwarded to devices just based on device ID
Authentication (-bypass)	▪ Passwords can be enumerated without rate limiting ▪ Default passwords yield high numbers of devices
Remote code execution	▪ Malicious firmware updates can be installed remotely ▪ The process can be automated for botnet creation

Many vendors employ similar cloud solutions

Cloud technology	Backends	Camera vendors	Apps
Cloudlinks	videoipcamera.com videoipcamera.cn cloud-links.net cloudlinks.cn	Sricam HKVstar / Unifore HiKam Digoo All with npc FW[1]	Sricam YooSee 2CU APcam All with p2p-core[2]
ezviz	hik-connect.com ezvizlife.com	Hikvision EZVIZ	Hikconnect iVMS-4500 EZVIZ
easy4ip	easy4ip.com ?	Dahua Various grey-market rebrands	Easy4ip gDMSS / iDMSS

All other vendors we looked at had cloud solutions for remote access as well:

- Axis → Axis companion / MyAxis
- D-Link → mydlink cloud
- …

[1] http://www.gwell.cc/e/action/ListInfo/?classid=102
[2] http://cloudlinks.cn/sdk/android/docs/index.html

Premium vendors make similar mistakes

	Market position	Cloud service problems	Latest authentication bypass
Hikvision	▪ Biggest video surveillance company by market share [1]	▪ Firmware update enabled Hikconnect with password ABCDEF ▪ Device IDs and passwords can be checked per POST without rate limiting ▪ There are 2,760,000* valid device IDs ▪ 50,000* have the password ABCDEF *estimate based on 100.000 random samples	▪ March 2017[2] ▪ CGI checks only for the username portion of "auth" parameter ▪ Access the camera as admin user
Dahua	▪ Second biggest video surveillance company by market share [1]	▪ Lorex sells Dahua devices with FLIR cloud ▪ FLIR establishes tunnel to camera just based on device ID[3]	▪ March 2017[4] ▪ Directly download list of users and passwords ▪ Exploitable via cloud tunnel

Other interesting research:

- Zoltan Balazs: The real risks of the IoT security-nightmare
- Amit Serper: Zero-day exploits in IP cameras

[1] https://ipvm.com/reports/video-surveillance-companies-top10-market-share
[2] http://seclists.org/fulldisclosure/2017/Sep/23
[3] https://depthsecurity.com/blog/unauthorized-flir-cloud-access
[4] http://seclists.org/fulldisclosure/2017/Mar/7

Users can only avoid cloud and p2p functionalities

- **Deactivate p2p if possible** → There may be no option for this – or the option has no effect[1]
- **Seperate the camera from the internet and access via VPN** → Only for technical users
- **Contact your vendor** → We tried that and it was not very productive

↓

Users depend on the vendors to build secure systems

[1] https://krebsonsecurity.com/2016/02/this-is-why-people-fear-the-internet-of-things/

Vendors need to apply well known security principles to their proprietary solutions

	In summary	What we need
Missing/ weak authentication	▪ Low-entropy device IDs ▪ Widely shared default passwords with skippable change prompt ▪ Packets forwarded just based on device ID	▪ High-entropy device IDs ▪ Unique, strong default passwords, unskippable security prompts ▪ Authentication check before forwarding packets to camera
Insufficient rate limiting	▪ Multiple authentication endpoints (UDP and HTTP) without any rate limiting or monitoring	▪ Basic rate limiting and monitoring for all endpoints
Coarse access control	▪ Successful authentication allows for vast reconfiguration, from anywhere	▪ Limit info leakage and reconfiguration possibilities, especially from the Internet
Missing/ improper use of crypto	▪ No transport layer security ▪ Firmware "signature" with MD5 and DES ▪ Symmetric encryption of secrets with keys hardcoded in the app	▪ Proper encryption of all traffic ▪ Asymmetric firmware signatures

Thank you!

Cloud services make it possible to reach large numbers of IP cameras in private networks. As there will always be vendors with insecure protocols and devices, we need to be prepared for DDoS attacks.

Many thanks to **Marvin Bornstein** and our friends at SRLabs
– **Karsten Nohl, Luca Melette, Mark Carney,** and **Stephan Zeisberg** –
for making this research possible!

Questions?

Balthasar Martin <balthasar@srlabs.de>
Fabian Bräunlein <fabian@srlabs.de>