Next-Gen Mirai

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

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

---

**Regions affected by attack on Dyn**

<table>
<thead>
<tr>
<th>DDoS against Akamai (Gbps)</th>
<th>Regions affected by attack on Dyn</th>
<th>Not actively used yet</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://krebsonsecurity.com/2016/11/akamai-on-the-record-krebsonsecurity-attack/" alt="Bar chart" /></td>
<td><img src="https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/" alt="Map" /></td>
<td><img src="https://www.arbornetworks.com/blog/asert/reaper-madness/" alt="Question mark" /></td>
</tr>
</tbody>
</table>

- **Biggest attack before Mirai**
- **Mirai attack in 09/2016**

---

[^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, ...

- Video and bidirectional sound
- Remote access from everywhere
- 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.

Problem: Router firewalls do not allow incoming connections.
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

- **App**
  - GET LoginCheck.ashx [user, md5(pw)]
  - GET GetFriendList.ashx
    - [name1, device_id1, e(pw1)]
    - [name2, device_id2, e(pw2)]
    - ...

- **Backend**
  - SessionID

### Adding a device to an account

- **App**
  - POST AddFriend.ashx [name, device_id, e(pw)]

- **Backend**
  - OK

**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

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Request Packet" /></td>
<td><img src="image2.png" alt="Response Packet" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Header</th>
<th>No. devices</th>
<th>Device IDs</th>
<th>Online status</th>
</tr>
</thead>
</table>

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

### Backend

<table>
<thead>
<tr>
<th>Backend</th>
<th>Dev. ID length</th>
<th>Collected IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>videoipcam</td>
<td>6 digits</td>
<td>140,741</td>
</tr>
<tr>
<td>cloudlinks</td>
<td>7 digits</td>
<td>3,277,280</td>
</tr>
</tbody>
</table>
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
- Authentication (-bypass)
- Remote code execution

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

<table>
<thead>
<tr>
<th>Backend</th>
<th>Password</th>
<th>No. devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Videoipcam</td>
<td>888888</td>
<td>63,029</td>
</tr>
<tr>
<td>Videoipcam</td>
<td>123456</td>
<td>1,454</td>
</tr>
<tr>
<td>Cloudlinks</td>
<td>123</td>
<td>703,000*</td>
</tr>
<tr>
<td>Cloudlinks</td>
<td>123456</td>
<td>46,600*</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>814,083</strong>*</td>
</tr>
</tbody>
</table>

*estimates based on a random 1,000 devices sample

- View camera feeds, turn devices, hear and send audio
- Get WiFi credentials, near network names, mail credentials
- Access and change device settings
Demo: Enumerating device IDs and passwords
We can access a large number of devices – now we need to execute commands on them

- Low entropy device IDs allow for efficient enumeration
- Packets are forwarded to devices just based on device ID
- Passwords can be enumerated without rate limiting
- Default passwords yield high numbers of devices
The filesystem in the firmware can be manipulated to add a backdoor.

```bash
$ binwalknpcupg_14.00.00.52.bin
```

<table>
<thead>
<tr>
<th>DECIMAL</th>
<th>HEXADECIMAL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0x20</td>
<td>JFFS2 filesystem, little endian</td>
</tr>
<tr>
<td>2943372</td>
<td>0x2CE98C</td>
<td>ELF, 32-bit LSB executable, ARM, version 1 (SYSV)</td>
</tr>
</tbody>
</table>

FW header

- JFFS2 filesystem
  - dhcp.script
  - gwellipc
  - minihttpd.conf
  - npc
  - upgfile_ok
  - version.txt
  - [...] (..)

32-bit ELF binary

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

- 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
  - Newest 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!

### 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 ...

### Byte-wise comparison of expected and given hash
Mass-scale remote installation of malicious firmwares possible by redirecting camera to attacker’s update server

Remember the network settings packet?

<table>
<thead>
<tr>
<th>IP</th>
<th>192.168.12.122</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet</td>
<td>255.255.255.0</td>
</tr>
<tr>
<td>Gateway</td>
<td>192.168.12.1</td>
</tr>
<tr>
<td>DNS</td>
<td>[own_dns_server]</td>
</tr>
</tbody>
</table>

Initiate firmware update and deliver malware

Attacker

- **CMD get network settings**
- **CMD set DNS to Attacker IP**
- **CMD do firmware update**
  - **DNS upg.videoipcamera.cn**
  - **Attacker IP**
  - **GET Version, GET update**
  - **Newer version, malicious update**

Camera

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

<table>
<thead>
<tr>
<th>Cloud technology</th>
<th>Backends</th>
<th>Camera vendors</th>
<th>Apps</th>
</tr>
</thead>
</table>
| 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.com      | ▪ 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](http://www.gwell.cc/e/action/ListInfo/?classid=102)
Premium vendors make similar mistakes

<table>
<thead>
<tr>
<th>Market position</th>
<th>Cloud service problems</th>
<th>Latest authentication bypass</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hikvision</strong></td>
<td>• Biggest video</td>
<td>• March 2017[2]</td>
</tr>
<tr>
<td></td>
<td>surveillance company</td>
<td>• CGI checks only for the</td>
</tr>
<tr>
<td></td>
<td>by market share[1]</td>
<td>username portion of “auth”</td>
</tr>
<tr>
<td></td>
<td>• Firmware update</td>
<td>parameter</td>
</tr>
<tr>
<td></td>
<td>enabled Hikconnect</td>
<td>• Access the camera as admin</td>
</tr>
<tr>
<td></td>
<td>with password ABCDEF</td>
<td>user</td>
</tr>
<tr>
<td></td>
<td>• Device IDs and</td>
<td>• March 2017[4]</td>
</tr>
<tr>
<td></td>
<td>passwords can be</td>
<td>• Directly download list of</td>
</tr>
<tr>
<td></td>
<td>checked per POST</td>
<td>users and passwords</td>
</tr>
<tr>
<td></td>
<td>without rate limiting</td>
<td>Exploitable via cloud</td>
</tr>
<tr>
<td></td>
<td>• There are 2,760,000*</td>
<td>tunnel</td>
</tr>
<tr>
<td></td>
<td>valid device IDs</td>
<td>• lorex sells Dahua</td>
</tr>
<tr>
<td></td>
<td>• 50,000* have the</td>
<td>devices with FLIR cloud</td>
</tr>
<tr>
<td></td>
<td>password ABCDEF</td>
<td>FLIR establishes tunnel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to camera just based on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>device ID[3]</td>
</tr>
</tbody>
</table>

| **Dahua**       | • Second biggest       | • Lorex sells Dahua devices  |
|                 | video surveillance     |   with FLIR cloud           |
|                 | company by market share|   FLIR establishes tunnel   |
|                 | [1]                    |   to camera just based on   |
|                 |                       |   device ID[3]              |

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
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\(^1\)
- **Separate 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

---

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

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

Insufficient rate limiting

Coarse access control

Missing/improper use of crypto
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!