The printer goes brrrrr

SYNACKTIV

Introduction

Who are we

Synacktiv

Offensive security company

Offices in Paris, Lyon, Toulouse and Rennes

~ 120 Ninjas

We are hiring !!!

Mehdi TALBI

SECURITY EXPERT
@abu_y0ussef

Rémi JULLIAN

SECURITY EXPERT
@netsecurity1

Thomas JEUNET

SECURITY EXPERT
@cleptho

SYNACKTIV

Pwn2own

Pwn2Own Contest

International Contest organized by ZDI (Trend Micro)

Pwn2Own Austin 2021

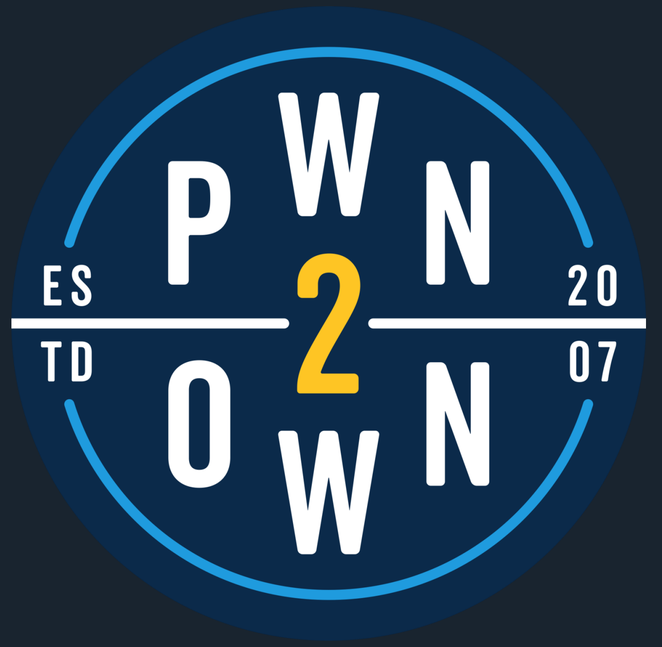
58 total entries

Won by Synacktiv

19 different devices

Routers, NAS, Printers, …

22 teams



Pwn2own

The Printers

Pwn2own

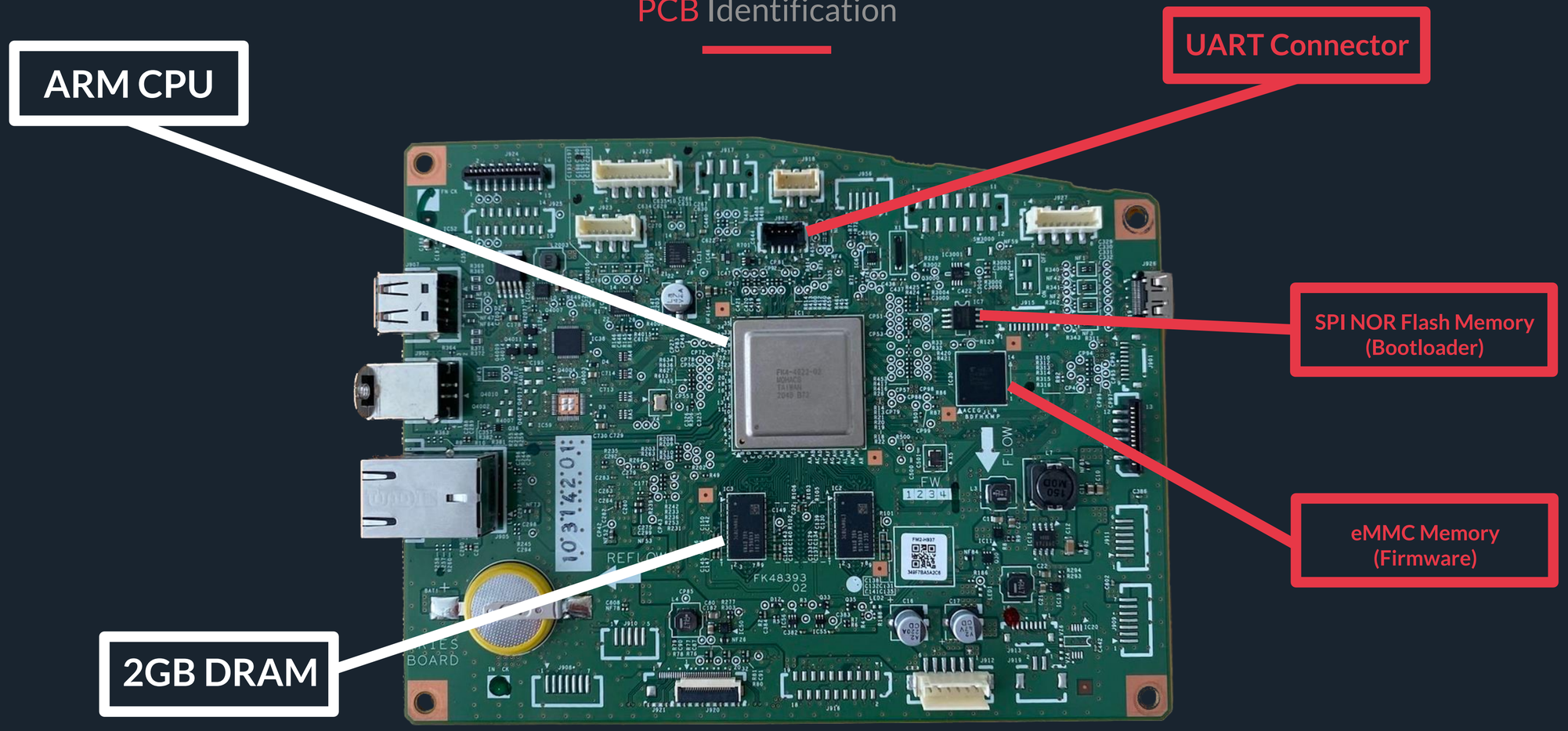
Leaderboard

Contestant	CASH	POINTS
SYNACKTIV	$197,500	20
DEVCORE	$180,000	18
STARLABS	112,500	12
SAM THOMAS	$90,000	9
THEORI	$80,000	8
BIEN PHAM	$62,500	6.5
NCC GROUP	$60,000	5
TRICHIMTRICH	$40,000	5
MARTIN RAKHAMANOV	$40,000	4
FLASHBACK	$33,750	3.75

Bootloader

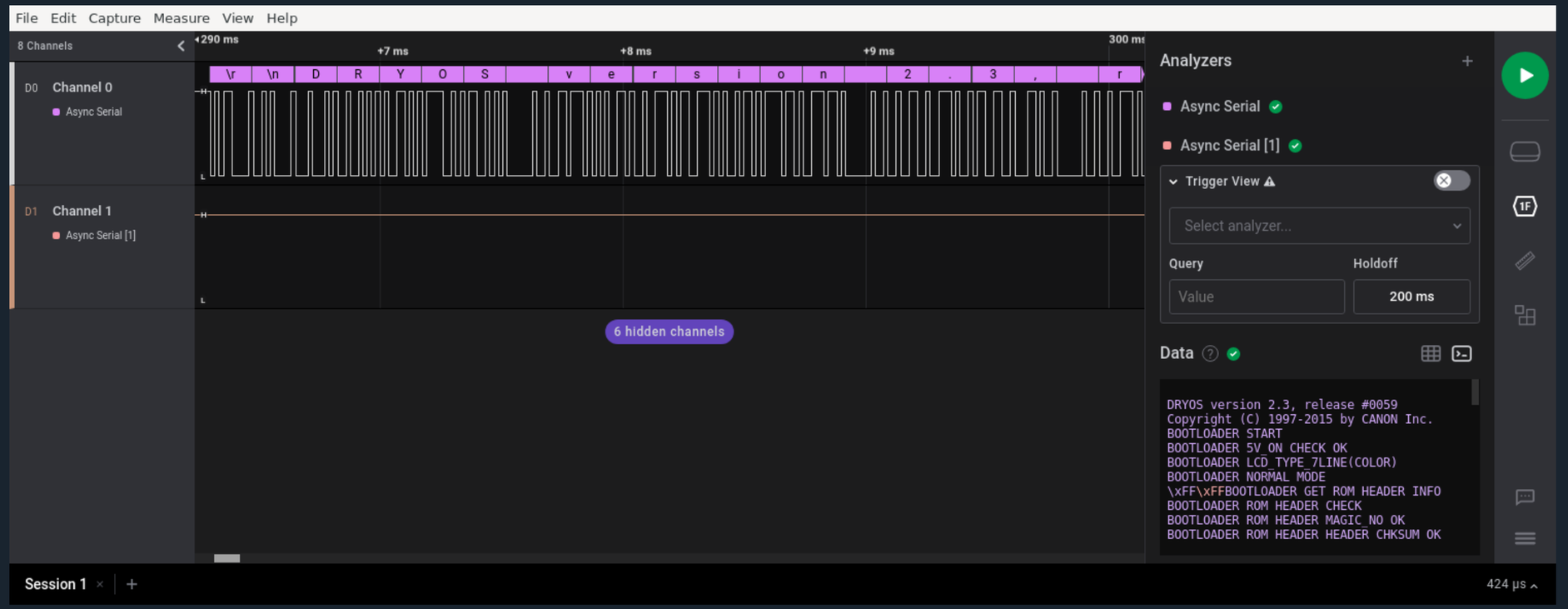
Bootloader

PCB Identification



Bootloader

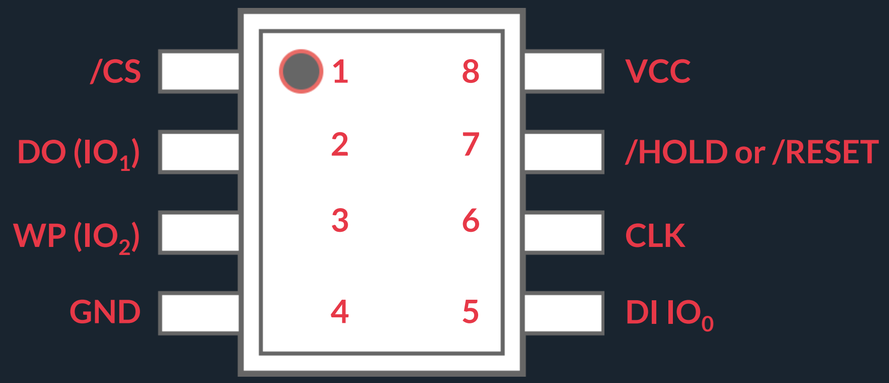
UART Start Sequence



SYNACKTIV

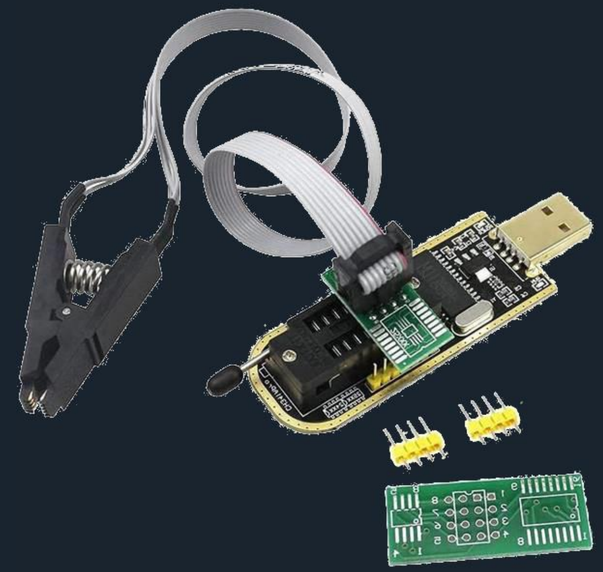
Bootloader

Dumping The Bootloader



W25Q16JV (datasheet)

Flashrom + SOP8 clip + CH341

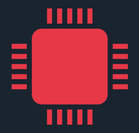


Bootloader

Analysis

The bootloader is able to “download” a firmware to the eMMC

eMMC at 0x1500000



RAM at 0x40b00000

```
if (emmc_direct_read(0x40B00000, 0x1500000u, 0x40u) != 0x40)
{
    log("BOOTABLE HEADER READ ERROR\n");
    return -1;
}
```

Bootloader

Deobfuscation Routine

```
uint8_t* NCFW_deobfuscate(uint8_t *data, unsigned int size, char offset)
{
    uint32_t i;
    uint32_t tmp;

    for (i = 0; i < size; ++i) {
        tmp = (uint8_t)(data[i] - (offset + i) - 1);
        data[i] = ~((2 * tmp) | (tmp >> 7));
    }
    return data;
}
```

Firmware

Getting The Firmware

The hard way

Dump the eMMC

The easy way

Setup a HTTP Proxy
Intercept URL updates

Alternative easy way

Download ‘MF63Cdw/MF641Cw Firmware Update Tool’
Extract Firmware

Package Format

SYNACKTIV

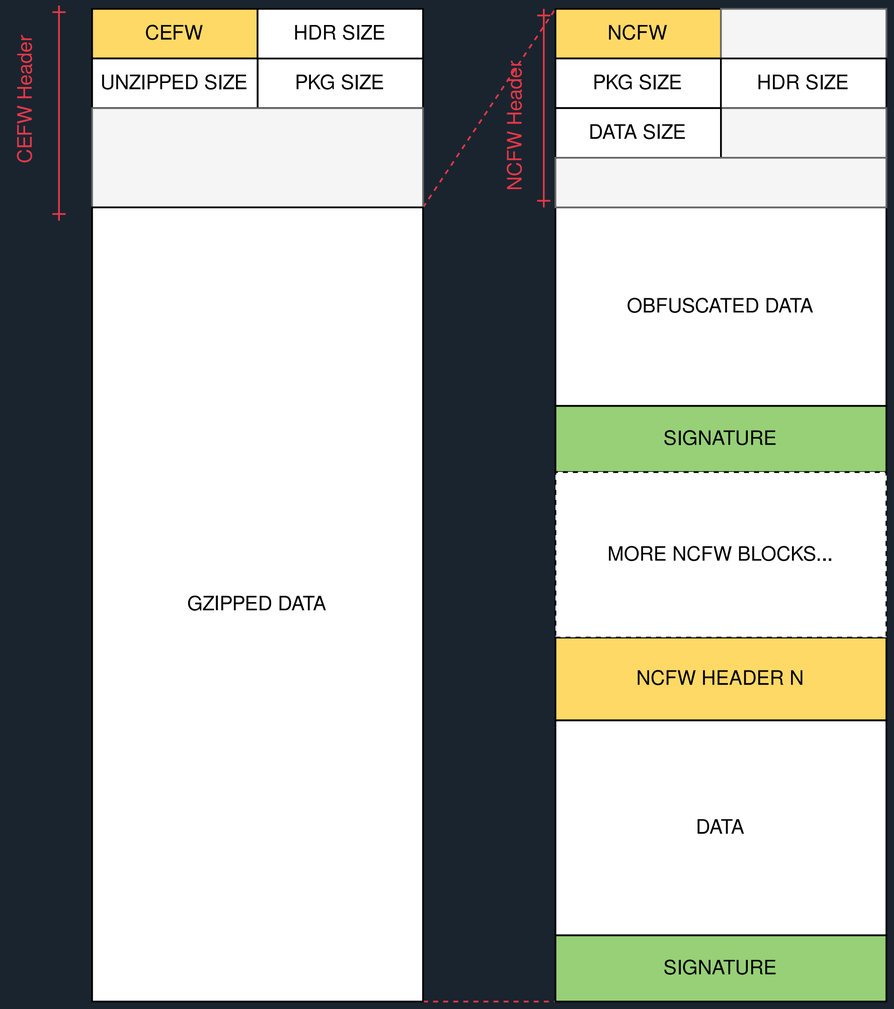
Package Format

CEFW Block

Gzipped Content
Only present in packages downloaded from Canon website
Multiple NCFW blocks (once uncompressed)

NCFW Block

Obfuscated data with routine identified in the bootloader
Multiple NCA blocks (once deobfuscated)



Package Format

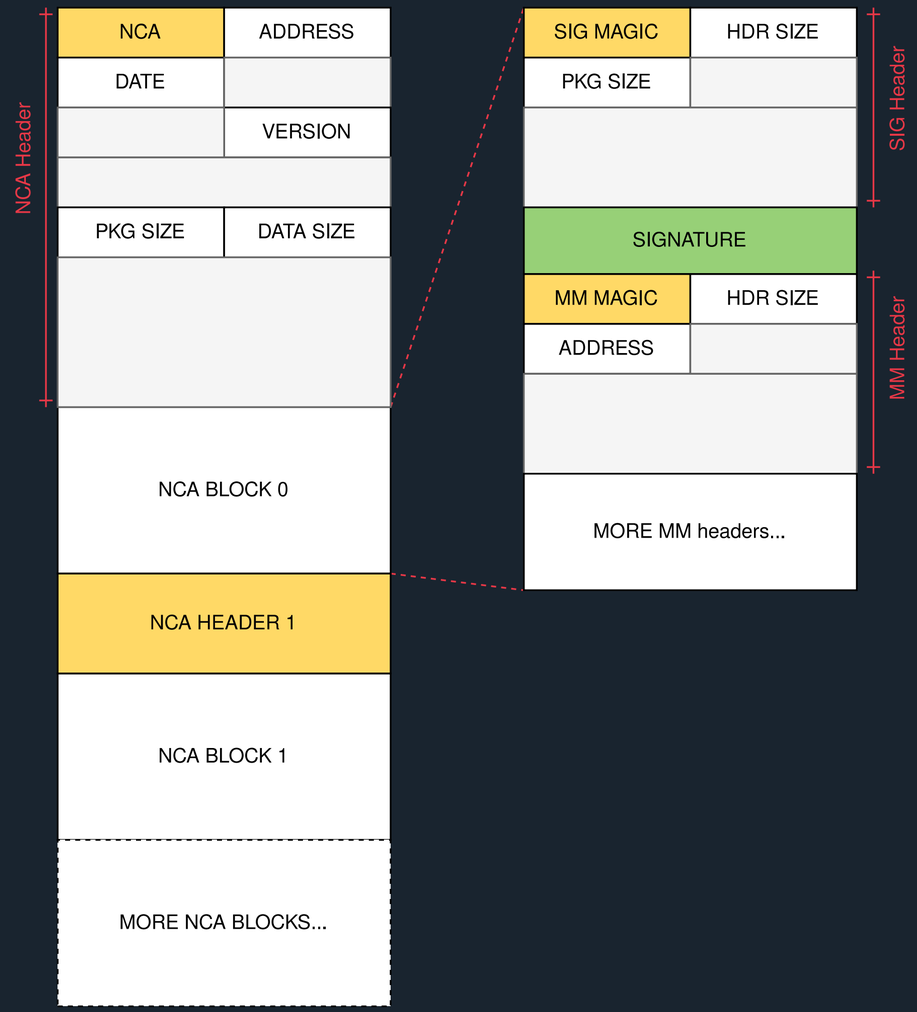
NCA Block

Block of data written on the eMMC

- eMMC address
- RAM loading address
- Version & Release date
- Etc.

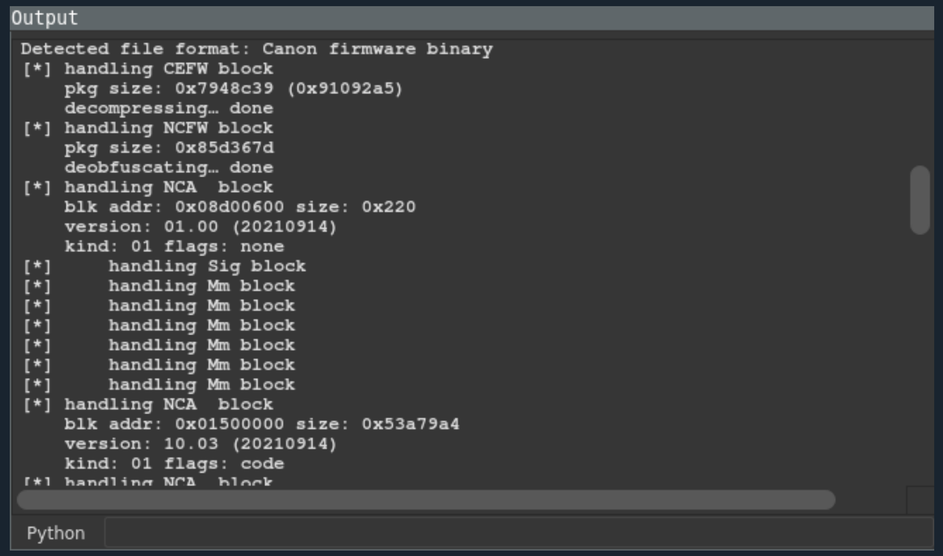
NCA Block 0

SIG Block + Multiple MM headers
(one per further NCA blocks)



IDA Loader

Parse Canon package format



Code on Synacktiv's Github repository

Firmware analysis

Firmware

Primary Analysis

- ARM instruction set
- DryOs Operating System
- > 100k functions!!
- Scripts to rename functions
  - Based on logging API
  - More than 2700 functions renamed

```
logf(2802, 3, "[CPC] %s ERROR [Fail getOperationParam]\n", "pjcc_act_checkUserPassword2");
logf(3604, 3, "[CADM] %s: cadmMessage.message.pEventMessage is NULL", "cadm_sendEventMessage");
logf(3520, 6, "[USBD] %s EPNo = 0x%X EPNoSS = 0x%X\n", "ScanBULK Out", (unsigned __int8)v14[0], v1)
```

Firmware

DryOS Overview

DryOs

- Canon custom Real Time Operating System
- Used for printers, DSL cameras, etc.
- Older release identified on a Canon MX920 series
- Based on µITRON

µITRON

- Micro Industrial TRON
- Japanese RTOS
- Specification publicly available

```
sub_40C9F5BC("DRYOS version 2.3, release #0059");

/* … */

sub_414FDE88(" Dry-ITRON4.0 object name : isem, iflg, idtq, imbx,
impf, impl, icyc\n");
```

Firmware

DryOS Mitigations

- The whole system is linked into a single module
- No ASLR
- No stack-cookies
- No W^X protection
- No security assertions
- … and obviously, no modern protections (CFI, etc.)

Firmware

Dryshell

Debug shell

Available via the UART

413 unique commands

46 command families

- System utilities
- Network
- Debug
- Etc.

```
Dry> vers
DRYOS version 2.3, release #0059
Dry-MK 2.66
Dry-DM 1.21
Dry-FSM 0.10
Dry-EFAT 1.22
Dry-stdlib 1.57
Dry-PX 1.15
Dry-drylib 1.22
Dry-shell 1.19
Dry-command alpha 065
```

Firmware

Dryshell

Memory access (useful for exploitation)

- **xd:** Dump memory
- **xm:** Modify memory

eMMC Dumping

```
// Dry> emmc_dump 1500000 64
// read address = 0x01500000.

// dump size = 64.
// |+0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
// 01500000|AF AF 9C 9C 01 50 00 00 20 21 09 14 00 00 00 01
// 01500010|58 58 78 78 10 03 01 01 05 3A 79 A4 05 3A 79 64
// 01500020|00 00 00 00 40 B0 00 00 00 00 00 00 00 00 00 00
// 01500030|00 00 00 00 00 00 00 00 00 00 00 08 63 7A A0
// 01500040|
```

Hunting for vulnerabilities

Attack Surface

HTTP/HTTPS
Canon Web Server

MFNP
Canon Print & Scan Jobs –
8610/TCP, 8610/UDP

SNMP
161/UDP

LPD
Line Printer Daemon Protocol
515/TCP

Zeroconf
Multicast DNS (Bonjour Apple)
5353/UDP

SLP
Service Location Protocol
427/UDP

IPP/IPPS
Internet Printing Protocol
631/TCP, 10433/TCP

NetBIOS
137/UDP, 138/UDP

CADM
Canon Administration
Proprietary protocol
9007/TCP, 9013/TCP,
47545/UDP, 47545/TCP,
47547/TCP over SSL

JetDirect
PDL-based printing
9100/TCP

WSD
Web Services Dynamic Discovery
3702/UDP

Vulnerability

Overview

Heap-based overflow in the CADM service (CVE-2022-24672)

21 Jan 2022

Vulnerability reported (by ZDI) to vendor

18 Mars 2022

Coordinated public release of advisory

CADM Service

Overview

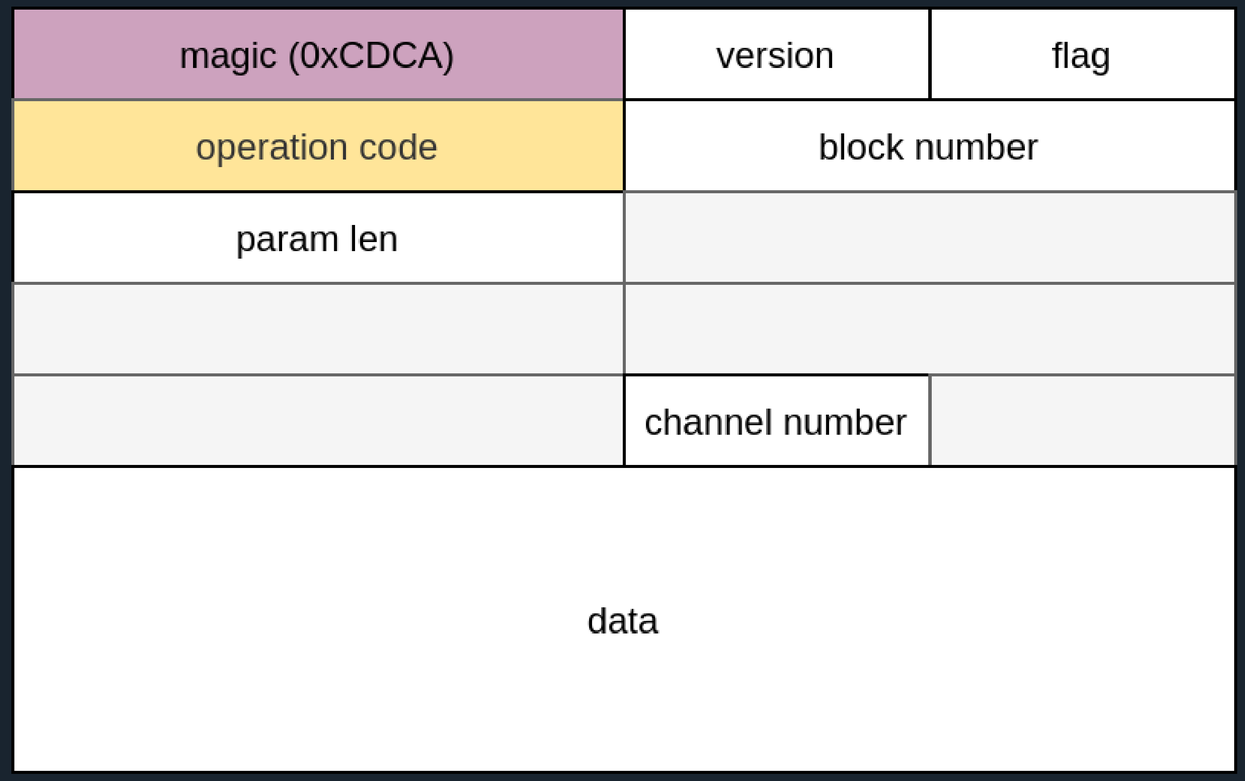
Canon ADMinistration?

41 supported operations

- Add new user
- Start job
- Shutdown device
- Etc.

CADM Service

Message Format



The Vulnerability

Vulnerable Code

```
uint32_t pjcc_dec_ope_checkUserPassword2(int *pkt, int a2, int *a3)
{
    /* ... */

    alloc = (pjcc_checkpassword_payload *)pjcc_zeroAlloc(428);
    pjcc_checkpass_obj = alloc;

    v7 = pjcc_dec_ubyte(pkt, alloc);
    v12 = pjcc_dec_ulong(pkt, (int)&pjcc_checkpass_obj->field_4);

    v14 = pjcc_dec_ubyte(pkt, &pjcc_checkpass_obj->buffer_len);
    v17 = pjcc_dec_buffer(pkt, pjcc_checkpass_obj->buffer_len, (char *)pjcc_checkpass_obj->buffer, v15);

    v19 = pjcc_dec_ubyte(pkt, &pjcc_checkpass_obj->salt_len);
    v22 = pjcc_dec_buffer(pkt, pjcc_checkpass_obj->salt_len, (char *)pjcc_checkpass_obj->salt, v20);

    v24 = pjcc_dec_ubyte(pkt, &pjcc_checkpass_obj->hash_len);

    res = pjcc_dec_buffer(pkt, pjcc_checkpass_obj->hash_len, (char *)pjcc_checkpass_obj->hash, v25);

    /* ... */
}
```

checkUserPassword2 Format

The Vulnerability

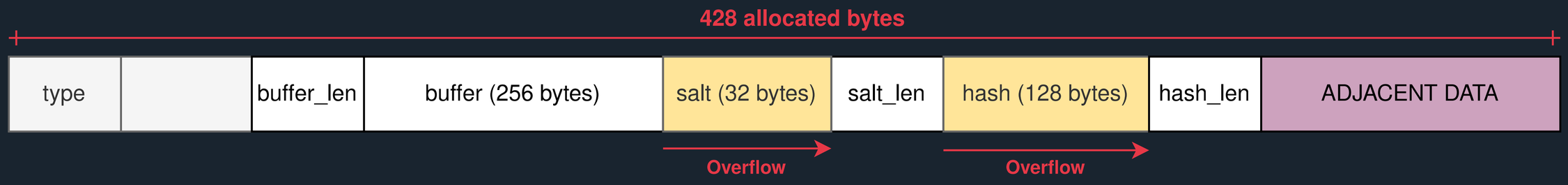
Heap overflow

Multiple copies without size checking

2 vulnerable buffers

Overflow with:

- Controlled size
- Controlled data



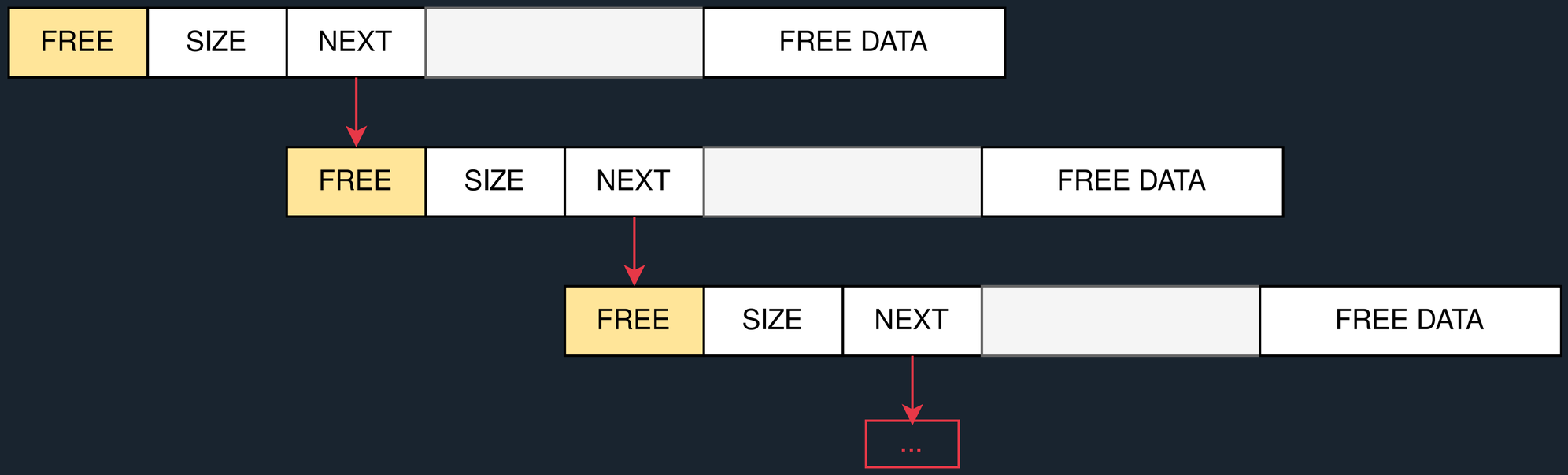
DryOs Allocator

Overview

“best-fit” allocator

Linked list of free chunks

40 bytes of metadata



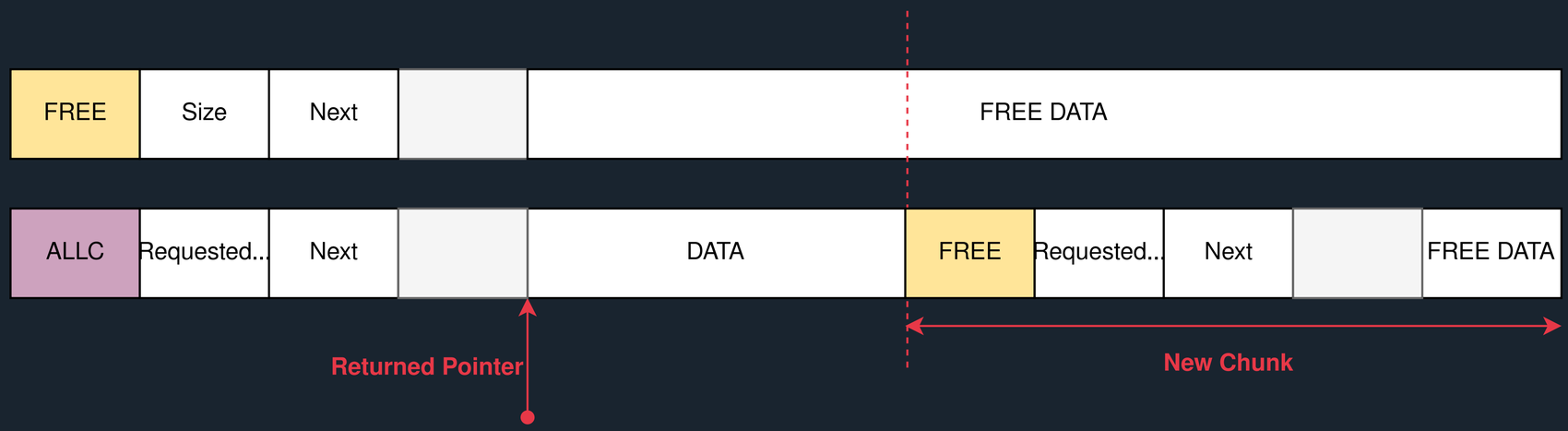
DryOs Allocator

Allocation

malloc returns the first free chunk that fulfills the requested size

Creates a new chunk with remaining space

IF chunk_size – request_size > metadata_size



Unlink the chunk from the freelist

DryOs Allocator

DeAllocation

Free chunk inserted back in the freelist

Free chunks ordered by their address

Chunk merged with adjacent free chunks

DryOs Allocator

Heap State

Custom DryShell command: !hd

- Iterates over the freelist
- Uses built-in ‘xd’ command to read memory

```
DryOs > !hd
magic = 0x0, size = 0x5ff930, next = 0x49c1dc88
magic = 0x46524545, size = 0x48, next = 0x49c1e7c0
magic = 0x46524545, size = 0x78, next = 0x49c30e50
magic = 0x46524545, size = 0x30, next = 0x49c30f10
magic = 0x46524545, size = 0x60, next = 0x49c35c98
magic = 0x46524545, size = 0x48, next = 0x49d0b578
magic = 0x46524545, size = 0x60, next = 0x49d14c70
magic = 0x46524545, size = 0x60, next = 0x49d15a18
magic = 0x46524545, size = 0x240, next = 0x49d22268
magic = 0x46524545, size = 0x2848, next = 0x49d24b68
magic = 0x46524545, size = 0x9198, next = 0x49d2ddd8
magic = 0x46524545, size = 0x292140, next = 0x0
```

Exploitation

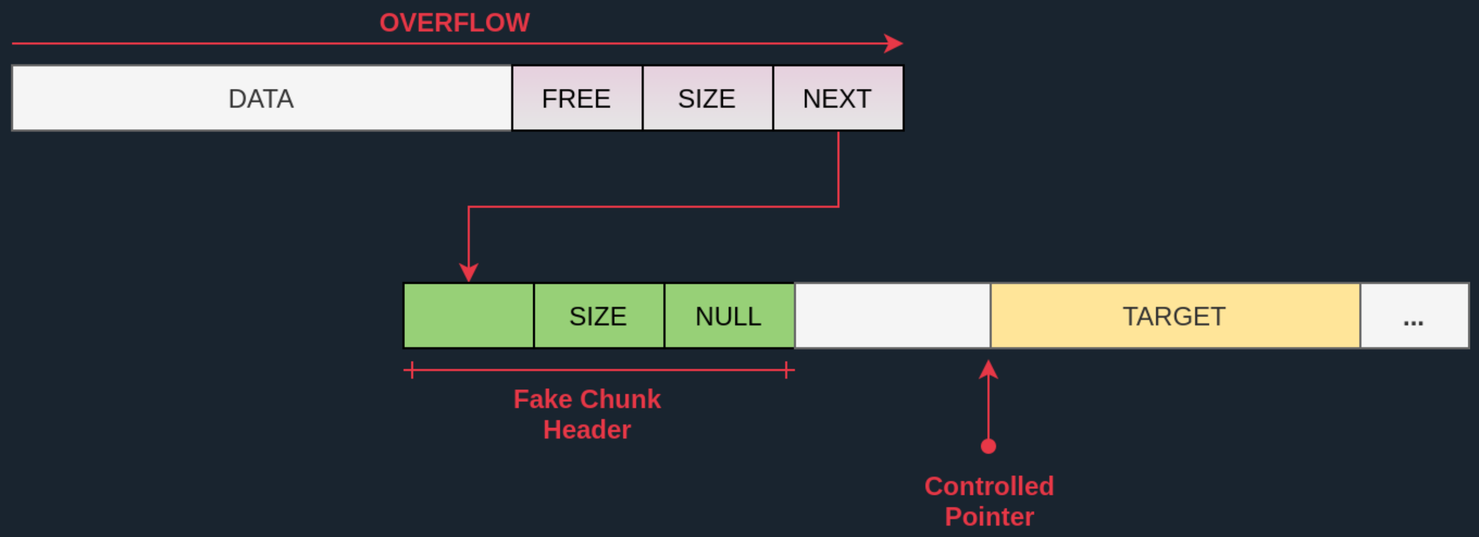
Attacking the allocator

No security checks

All chunk's metadata can be corrupted

Arbitrary allocation

Overwrites the "Next" field pointer



Exploitation

Exploitation strategy

1 **Shape the heap**

Set up the heap in the desired state

2 **Trigger overflow**

Corrupt the ‘next’ pointer of adjacent memory chunk

Make it point to a memory region holding function pointers

3 **Allocate fake chunk**

Write shellcode

Overwrite function pointer

4 **Trigger code execution**

Jump to shellcode

Exploitation

Shaping the heap

Goal:

Force allocation from a large chunk

Prevent the allocator from serving our fake chunk at an early stage

---

How:

Perform an HTTPS request will fragment the heap with large chunks

- Insert large chunks in the freelist

Exploitation

Triggering overflow

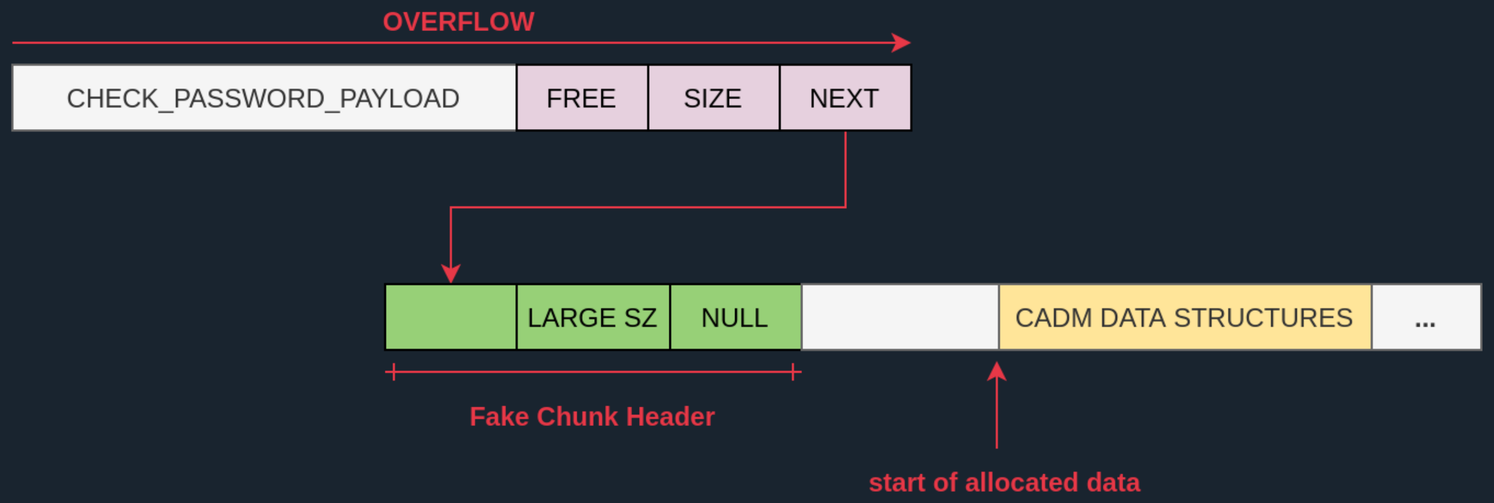
Send a crafted CADM CheckUserPassword payload

- Corrupt adjacent memory chunk
  - Overwrite ‘Next’ field pointer with the address holding CADM data structures (state machine, handlers, etc.)

Fake chunk

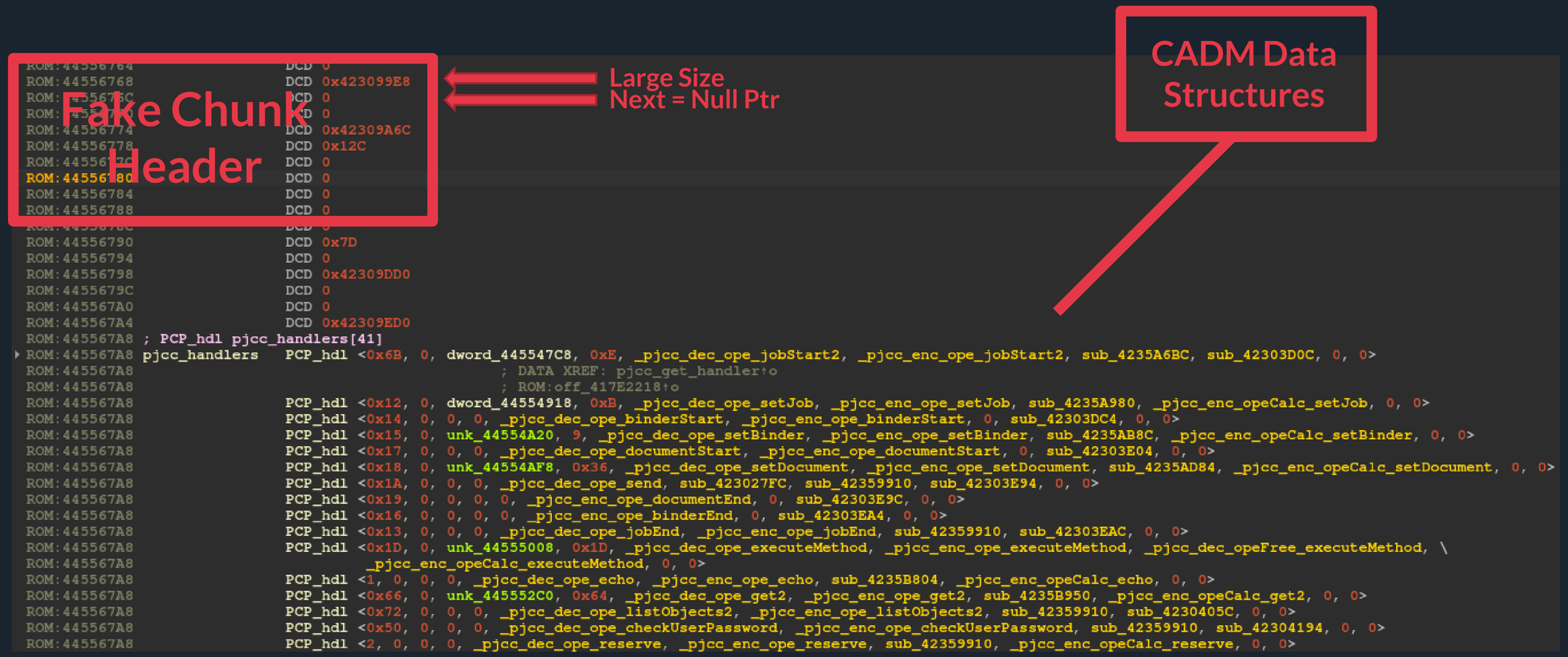
Large size

Next = NULL (close the freelist)



Exploitation

Fake chunk



Exploitation

Allocating fake chunk

CADM Echo Operation

Sends back identical copy of received data

Controlled allocation

- **Controlled size**
- **Controlled data**

Exploitation

Getting code execution

Overwrite CADM Data Structures

Copy shellcode

Overwrite the handler responsible for processing CADM Echo requests

Immediate code execution

Preserve the rest of the data to avoid crashes due to a corrupted state machine internal data

Exploitation

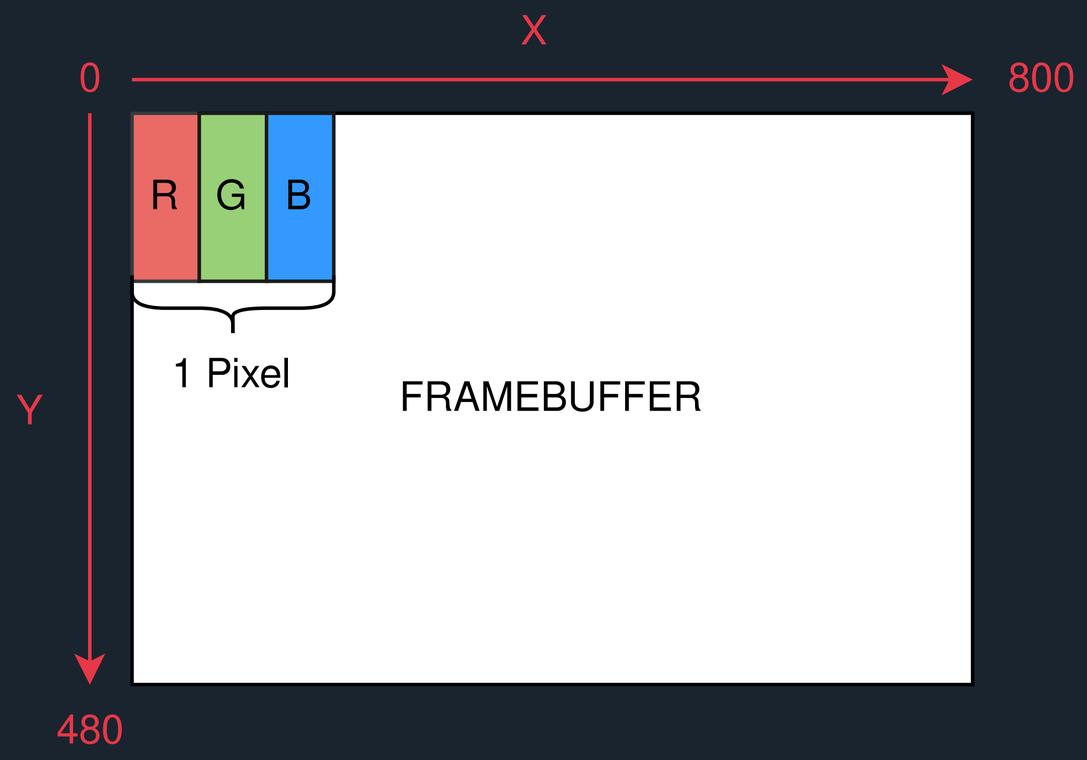
Displaying a ninja

800x480 LCD Screen

```
// String used in function close to
// frame buffer initialization
log("BOOTLOADER LCD_TYPE_%s\n", v0);
```

Frame Buffer

- Mapped at 0x40900000
- 3 bytes to encode 1 RGB pixel



Testing effect on LCD Screen

- Use DryShell ‘xm’ command

Exploitation

Displaying a ninja

Shellcode

Read picture from a socket

Implemented in ARM assembly (binutils-arm-none-eabi)

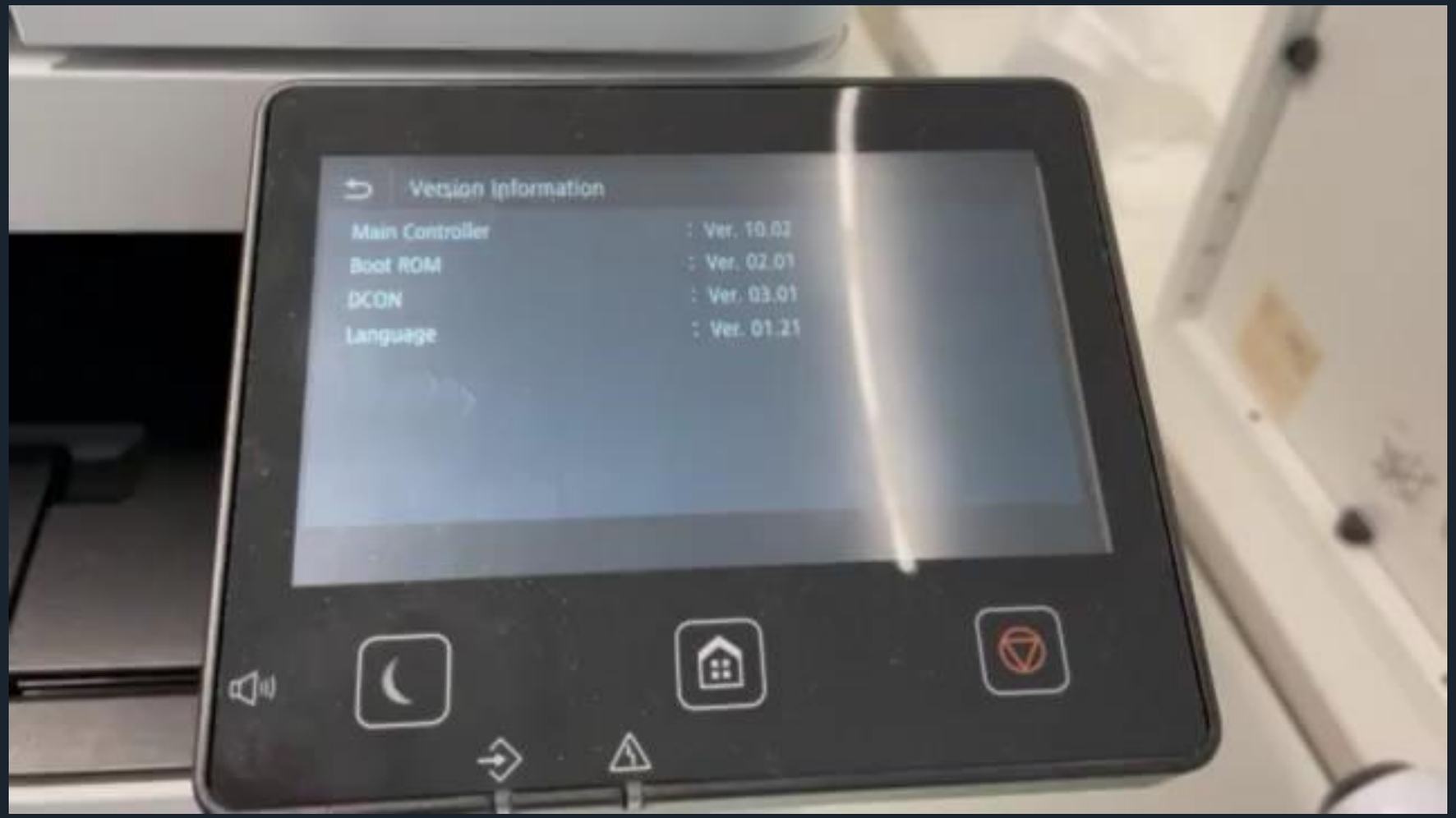
```
struct sockaddr_in addr = {
    .sin_familly = AF_INET,
    .sin_port = htons(9000);
    .sin_addr = htonl(0xC0A80102); // 192.168.1.2
};

int sockfd = netSocket(1, 1, 0, 0);
netConnect(sockfd, addr, 8);

while (1) {
    for(char *addr = 0x40900000; addr < 0x40a19400; addr += 3) {
       netRecv(sockfd, addr, 3, 0);
    }
    sedev_powerOnImgSns();
}
```

Server

Python script based on PIL

Exploitation

Demo



Conclusion

Conclusion & Perspectives

Conclusion

First Pwn2Own participation, lots of fun!!

Perspectives

Persistence mechanism

Process continuation

Pwn2Own 2022? 

Tools

Tools & Scripts

Available at Synacktiv's Github Repository

Exploit working on firmware v10.02

IDA Python loader for Canon firmware

IDA Python script for function renaming

https://github.com/synacktiv/canon-mf644/

References

Reversing a Japanese Wireless SD Card Zero to Code Execution

THCON 2021 – Zombies Ate my Printer's Ink

µITRON 4.0 Specification

Questions?