Reverse Engineering in some Dutch Wireless Routers

Research 'B'

Eduardo Pablo Novella Lorente
Co-author : Carlo Meijer
Supervisor : Roel Verdult

The Kerckhoffs Institute
Radboud University Nijmegen

Institute for Computing and Information Sciences

16th December 2014

Outline

DISCLAIMER (1/3)

This presentation only pretends to show a big picture of vulnerabilities discovered.

Both vendors and ISPs are being warned now.

DISCLAIMER (2/3)

RU Nijmegen is following the responsible disclosure guidelines according to the Dutch government.

DISCLAIMER (3/3)

Vendors will be informed 6 months prior to full disclosure, giving them time to solve the issues, inform their customers and hence preventing widespread abuse.

Main goals

Find out the secret WPA key generation algorithms

Do responsible disclosure to ISP and vendors
and try to protect Dutch wireless routers

Learn about reverse engineering and hardware stuff

Hw protocols → JTAG, UART, SPI, memory chip extraction,
MIPS assembler code → static & dynamic firmware analysis
Datasheet → reading and understanding
Memory mapping, decompiling, signatures analysis, debugging

Do research and write a paper

Read previous literature and resume it (not reinvent wheel)
Write a public paper?
Mitigations

Why is so hard to break WPA? (1/3)

1. *Types* → Enterprise (companies, faculties ...) and Personal (domestic routers, home,small offices)
2. *Authentication* →
   - Enterprise → Online through a RADIUS server
   - Personal → Offline through PSK (Pre-shared key)
3. *Derived key* → PBKDF2 (Password-Based Key Derivation Function-2)
   - Shared key → Derived key of 256 bits
   - Input data → password, ssid, crypto hash function

Why is so hard to break WPA? (2/3)

Password-Based Key Derivation Function-2

Derived Key = PBKDF2 (
pseudo random function,
password,
salt,
iterations,
derived key length
)

```
derived Key =
PBKDF2(HMAC-SHA1,"ImaginationIsPowerpassword","ResearchBWifi",4096,256)
```

Why is so hard to break WPA? (3/3)

1. *Encryption* →
   - WPA1 → TKIP (Temporal Key Integrity Protocol)
   - WPA2 → AES (also TKIP-AES)
2. *WPS* Wi-Fi Protected Setup → Serious breach on WPA
   - Strong crypto → only 8 numeric digits?
   - 2 chunks :
     4 digits 10^4 + 3 digits 10^3 + 1 digit checksum = 11.000 tries

ISP_1

Findings

1. *Default WPA algorithm* → Time : 2 models either 1 sec or 10 minutes mid-GPU
2. *Backdoors*→ hidden administrator accounts (activate telnet)
3. *Telnetd: Command injection* → Got root :)
4. *Httpd: Stack-buffer overflow*→ Just locally :(

ISP_1: How to got WPA keys?

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

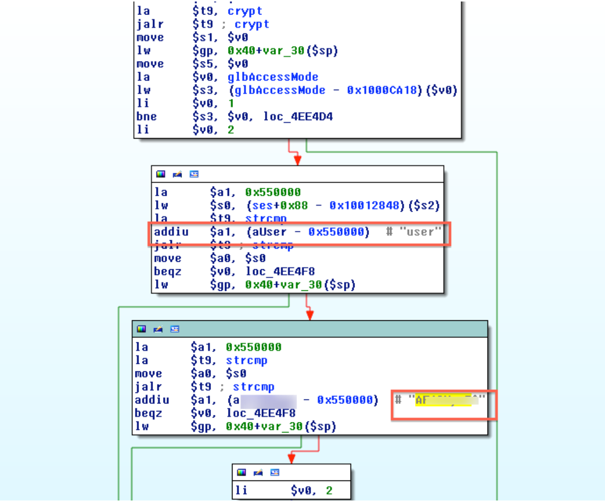
`802.11 headers reveal mac addresses in plaintext`

`(Monitor mode required)` $\rightarrow$ `Time reduces to seconds`

ISP_1: WPA key generation algorithm

```
la      $t9, sysNvRamGet
addiu   $a0, $sp, 0x2D8+var_150
li      $a1, 0x12C
jalr    $t9 ; sysNvRamGet
move    $a2, $zero
lw      $gp, 0x2D8+var_2B0($sp)
lbu     $a1, 0x2D8+var_2B($sp)
lbu     $v1, 0x2D8+var_2D($sp)
lbu     $a0, 0x2D8+var_2C($sp)
lbu     $v0, 0x2D8+var_2E($sp)
sw      $a1, 0x2D8+var_2BC($sp)
la      $a1, 0x550000
la      $t9, sprintf
lbu     $a3, 0x2D8+var_2F($sp)
lbu     $a2, 0x2D8+var_30($sp)
sw      $v1, 0x2D8+var_2C4($sp)
sw      $a0, 0x2D8+var_2C0($sp)
addiu   $a1, ([illegible] - 0x550000)    # "t[illegible]%02x%02x%02x%02x%02x%02x%s"
move    $a0, $s2
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
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 - 0x553930)
```

ISP_1: Backdoors: Hardcoded credentials and super-admin



ISP_1: Command Injection: Who knows the answer? :)

```
la      $a0, 0x550000
la      $t9, puts
jalr    $t9 ; puts
addiu   $a0, (aWarningOperato - 0x550000)  # "Warning: operator & is not supported!"
lw      $gp, 0xB8+var_A8($sp)
move    $a0, $s0
li      $a2, 2
la      $a1, 0x550000
la      $t9, strncpy
jalr    $t9 ; strncpy
addiu   $a1, (dword_549C84 - 0x550000)
lw      $gp, 0xB8+var_A8($sp)
```

```
loc_45E6D0:
la      $t9, strchr
move    $a0, $s3
jalr    $t9 ; strchr
li      $a1, 0x3B
move    $s0, $v0
beqz    $v0, loc_45E71C
lw      $gp, 0xB8+var_A8($sp)
```

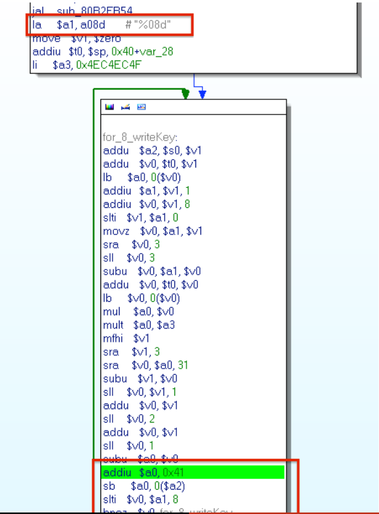
```
la      $a0, 0x550000
la      $t9, puts
jalr    $t9 ; puts
addiu   $a0, (aWarningOpera_0 - 0x550000)  # "Warning: operator ; is not supported!"
lw      $gp, 0xB8+var_A8($sp)
move    $a0, $s0
li      $a2, 2
la      $a1, 0x550000
la      $t9, strncpy
```

ISP_2

Findings

1. *Default WPA algorithm* → 10 minutes mid-range GPU (100.000 k/s)
   4.4 hours using a mid-range CPU (4000 keys/s)
   Serial number involved → 9 numerical digits to bruteforce
   Collisions → (63408999/1000000000) keys.

ISP_2 : WPA key generation algorithm



ISP_2

Observations

1. Blob stripped → no symbols, no functions, CHAOS!
2. Hard and time-consuming
3. Makes the task quite tough
4. Dynamic analysis with IDA and QEMU

BRAND_3

Previous findings

1. *Default WPA algorithm* → Around 5 routers. Time : 1 sec
2. *low entropy firmware images*→ Public firmware images (not anymore)
3. *Admin password == WPA key*

Findings

1. *Default WPA algorithm for a model specific* → Time : 1 sec
2. *Default WPA algorithm* → +21 routers Time : 1 sec
3. *New default WPA algorithm* → +11 routers Time : 1 sec
4. *Same problem with Admin password == WPA key*

BRAND_3 : WPA key generation algorithms

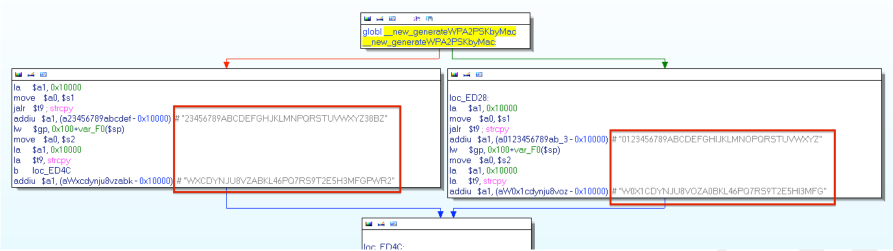


Figure : Default keys

BRAND_3

Observations

1. Blob not stripped → fast and easier
2. BIG FAIL!: Both WPA, WPS and web password are generated only from the mac address
3. Dynamic analysis with IDA and QEMU

ISP_3

Findings

1. *Router1: Default WPA algorithm* → just seconds
   Serial number involved → 5 numerical digits to bruteforce 100000 keys.
   No reversing engineering. Try old algorithms from same vendor
2. *Router2: obfuscation firmware* → we're working on it

ISP_3 : WPA key generation algorithm

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 $[x1, y1, z1, w1, x2, y2, z2, w2, x3, y3, z3, w3]$

Hardware tools

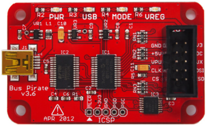


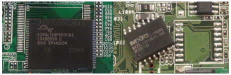
Figure : Bus pirate and USB Altera Blaster JTAG

Hardware tools

Figure : UART USB2TTL and Rework station

Hardware tools

Figure : EEPROM reader and memories desoldered



Software tools

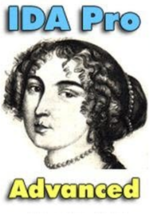
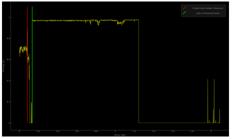


Figure : Ida Pro and Binwalk, QEMU-MIPS...



Thanks a lot!

Lejla Batina, Jaap-Henk Hoepman and Roel Verdult.

Stuff to take home

Remember:

1. Security in routers → "Security through obscurity"
2. Vendors REUSE algorithms
3. Break into in a wireless network with default config might be easy
4. WPA with good password and disabling WPS → SECURE
5. Hardware hacking is cool :)
6. JTAG is usually in CPUs and is opened

Mitigations:

1. Do not include algorithms into firmware
2. Write into flash a hardcoded value
3. SmartMIPS cores → crypto, memory protection, withstand SCA
4. Obfuscation makes harder
5. Firmware images stripped
6. Use a crypto-processor