Cyber Apocalypse 2026 — Hardware — Wireless Connections

Category: Hardware

Challenge file: `firmware.bin` (8 MiB)

Flag: `HTB{solv3d_w1th_xt3ns4_dec00mp}`

---

1. Challenge

> Stormbound's scouts swept a shuttered Suncourt townhouse after a chain of forged writs began reaching Crownspire before their couriers did. Behind a wall plate they found a crude listening node stamped with a single useful marking, **ESP32_S3_DevKitC-1U**. Somewhere inside it lies the **relay key** used to pass whispers between rival agents. Cut the hidden channel before the next false order moves through the realm.

We get a raw SPI-flash dump of an **ESP32-S3** device. The "relay key" is not stored in the clear — it is **encrypted** and only decrypts at runtime using a value that never appears in flash. The intended solve is to reverse the Xtensa firmware, recover the cipher, and brute-force the tiny key space offline.

---

2. Identifying the image

```
$ xxd firmware.bin | head -1
00000000: e903 023f b888 3c40 ee00 0000 0900 0000  ...?..<@........
```

`0xE9` = ESP32 image magic; byte `0x09` at offset 8 = **chip-ID 9 (ESP32-S3)**. The header banner reads `v5.5.1 … Sep 30 2025`, and the app-descriptor says `arduino-lib-builder` → an **ESP-IDF 5.5 / Arduino-core 3.3.2** firmware.

3. Partition table & extraction

Partition table at `0x8000`:

Label	Offset	Size	Notes
`nvs`	`0x009000`	`0x005000`	empty (`0xFF`)
`otadata`	`0x00E000`	`0x002000`	boot selector
`app0`	`0x010000`	`0x330000`	**the application**
`app1`	`0x340000`	`0x330000`	empty
`spiffs`	`0x670000`	`0x180000`	empty (`0xFF`)
`coredump`	`0x7F0000`	`0x010000`	empty (`0xFF`)

Mapping the whole flash shows content only in the bootloader, the partition table, and `app0` (ends at `0x0E4F60`). All data partitions are blank, so the secret is baked into `app0`.

```
dd if=firmware.bin of=app0.bin bs=1 skip=$((0x10000)) count=$((0x330
```

`strings` turns up only three custom tokens — `blink_task`, `bugged26`, `SpyHouse` — the Wi-Fi SSID/PSK of the covert network the node joins. **These are decoys**: they are the "hidden channel", not the flag. There is no `HTB{` string anywhere and no single-byte XOR of the image produces one.

4. Building a disassemblable ELF (Xtensa LX7)

`capstone 5` has no Xtensa backend, so we rebuild an ELF from the esptool segment map and use **radare2** (which ships an Xtensa disassembler).

```
$ python -m esptool --chip esp32s3 image-info app0.bin
Segment  Load addr   File offs   Type
   0     0x3c0a0020  0x00000018  DROM  (rodata)
   3     0x42000020  0x00030018  IROM  (code)   # real functions sta
```

`build_elf.py` wraps the six segments into an `EM_XTENSA` ELF; then `r2 -a xtensa -e anal.arch=xtensa app0.elf ; aaa`.

5. Reversing the relay-key routine

The main worker task (`fcn.42002cd4`) does:

```
Serial.begin(115200);
WiFi.begin("SpyHouse", "bugged26");          // join the covert chan
xTaskCreate(blink_task, ...);
for (;;) {
    if (WiFi.status() != WL_CONNECTED) { delay(250); continue; }
    uint8_t mac[6];
    WiFi.macAddress(mac);                     // fcn.42003e80 -> fcn
    uint32_t seed = (mac[0]<<16 | mac[1]<<8 | mac[2]) * 0x01000193;
    char out[32];
    decrypt(CIPHERTEXT, out, 32, seed);       // fcn.42002ca8
    Serial.println(out);
}
```

The `mull` by `0x01000193` (the **FNV-1 32-bit prime**) was the first tell. The decrypt routine `fcn.42002ca8` is a **Linear Congruential Generator keystream cipher**:

```
0x42002cab  l32r a10, =0x0019660d      ; LCG multiplier
0x42002cae  l32r a11, =0x3c6ef35f      ; LCG increment
loop:
  mull  a5, a5, a10                    ; state *= 0x19660D
  l8ui  a13, [src + i]
  add   a5, a5, a11                    ; state += 0x3C6EF35F
  extui a9, a5, 16, 16                 ; ks = (state >> 16)
  xor   a9, a9, a13                    ; plain = cipher ^ (ks & 0xff
  s8i   a9, [dst + i]
```

```
def decrypt(cipher, seed):
    st = seed; out = bytearray()
    for c in cipher:
```

```
        st = (st*0x19660D + 0x3C6EF35F) & 0xffffffff
        out.append(c ^ ((st >> 16) & 0xff))
    return bytes(out)
```

The 32-byte ciphertext is loaded from DROM (right after the `/dtoa.c` library string):

```
10 a0 68 e7 5d e6 e7 0f 12 bb b2 f3 ca b4 d2 02
cb de d8 20 0d e5 68 92 42 00 be 14 f0 7b ba 01
```

6. Cracking it offline

The seed depends only on the **first 3 bytes of the device MAC** (the OUI), so there are just 2^{24} possible seeds. The device would print the flag on real hardware (its MAC is a genuine Espressif OUI); offline we brute-force:

```
MUL, INC, FNV = 0x19660D, 0x3C6EF35F, 0x01000193
ct = bytes.fromhex("10a068e75de6e70f12bbb2f3cab4d202cbded8200de56892
for val in range(1 << 24):
    seed = (val * FNV) & 0xffffffff
    pt = decrypt(ct, seed)
    if pt[:4] == b"HTB{":
        print(f"MAC prefix {val:06x} -> {pt}")
        break
```

```
MAC prefix 7cdfa1 -> b'HTB{solv3d_w1th_xt3ns4_dec00mp}\x00'
```

The winning prefix `7C:DF:A1` is a real **Espressif Inc. OUI** — confirming the seed is the ESP32's own MAC, so the firmware decrypts correctly on any genuine board.

7. Flag

> `HTB{solv3d_w1th_xt3ns4_dec00mp}`

TL;DR

1. 1. 8 MB **ESP32-S3** flash dump; app lives in `app0`, everything else blank.
2. 2. `SpyHouse` / `bugged26` are Wi-Fi creds — a decoy "hidden channel", not the flag.
3. 3. Rebuild an Xtensa ELF, disassemble in radare2.
4. 4. Found an **LCG stream cipher** (`×0x19660D +0x3C6EF35F`, keystream `= state>>16`) decrypting a 32-byte blob, seeded by `FNV_prime × (MAC[0:3])`.
5. 5. Brute-force the 2^{24} MAC-OUI seed space → OUI `7C:DF:A1` (Espressif) yields `HTB{solv3d_w1th_xt3ns4_dec00mp}`.