
EMBEDDED HACKING

Reverse Engineering the Raspberry Pi Pico 2

CTF Challenge - Operation Black Start

ARM Cortex-M33 • GDB • Ghidra

September 9, 2026

For educational and defensive security research only

Instructor Solution Key



This key is for Operation Black Start only. It contains no FINAL-project answers, constants, addresses, bugs, or patches.

Artifact Identity

Artifact	Value
Student image	CTF-01.bin
Flash image	CTF-01.uf2
Target	Raspberry Pi Pico 2 / RP2350 ARM
Image base	0x10000000
UART	UART0, GPIO 0 TX / GPIO 1 RX, 115200 8N1

```
CTF-01.bin 6FD296F7A85F243FB26BF6BFFCBEAB26815FD8915101A81F72069063A5635E5A
CTF-01.uf2 980F04369C23AD32A063DFE18DE5AF08DF3830138FC7E7898B1F011B4F5E1D9D
```

Every address and byte value below was independently verified against the compiled ELF (`arm-none-eabi-nm` , `arm-none-eabi-objdump`) and the raw bytes of the delivered `CTF-01.bin` (direct hex read at each file offset). This image contains **no** LED, relay, sensor, display, or GPIO-control logic.

Instructor Scenario

WorldGrid Compact's emergency firmware build for the GRID-7 relay fleet was compiled and shipped eleven minutes after a cyberattack severed the primary SCADA uplink. The engineer (Dr. Elias Renner) had no time to review the build; the source used for that compile no longer exists. The training relay supplied to students holds the exact miscompiled image that shipped to the fleet, containing two real, independently patchable defects.

Students must reconstruct the boot path, locate both defects with Ghidra, patch the binary directly, export and convert it, and prove the corrected behavior on real hardware — the same workflow used in the FINAL projects.

Task 1: Setup and Initial Analysis Solution

Vector Table (first 32 bytes of `CTF-01.bin`)

```
00 20 08 20 5B 01 00 10 1B 01 00 10 1D 01 00 10
11 01 00 10 11 01 00 10 11 01 00 10 11 01 00 10
```

Evidence	Answer
Vector table base	0x10000000
Initial SP	0x20082000
Reset pointer (as stored)	0x1000015b
Reset instruction address (bit 0 cleared)	0x1000015a
<code>main()</code>	0x100001e0
Recurring status loop start	0x10000234
Loop branch (<code>b.n</code> back to loop start)	0x10000246

Data Symbols

Symbol	Address	Notes
<code>grid_deviation</code>	0x200005d8	Initialized to <code>87</code> (0.87 Hz x100), lives in <code>.data</code>
<code>operator_state</code>	0x20000844	Zero-initialized, <code>.bss</code>
<code>dispatch_state</code>	0x20000834	Zero-initialized, <code>.bss</code>
<code>dispatch_frame</code>	0x100037a0	Hidden black-start frame, <code>.rodata</code>

Task 2: Bug #1 Solution — Miscalibrated Safety Threshold

`grid_deviation` is declared `volatile`, so the compiler cannot reuse one comparison for both output paths — it emits **two independent** compare instructions, one for `operator_state` and one for `dispatch_state`:

```

100001f8: 6813      ldr    r3, [r2, #0]      ; r3 = grid_deviation
100001fc: 2b5e      cmp    r3, #94 @ 0x5e    ; <-- PATCH LOCATION A
100001fe: bf8c      ite    hi
10000200: 2300      movhi  r3, #0            ; operator_state = 0 (CRITICAL)
10000202: 2301      movls  r3, #1            ; operator_state = 1 (STABLE)
10000204: 6033      str    r3, [r6, #0]

10000206: 6813      ldr    r3, [r2, #0]      ; r3 = grid_deviation (2nd read)
1000020a: 2b5e      cmp    r3, #94 @ 0x5e    ; <-- PATCH LOCATION B
1000020c: bf8c      ite    hi
1000020e: 2300      movhi  r3, #0            ; dispatch_state = 0 (HELD)
10000210: 2301      movls  r3, #1            ; dispatch_state = 1 (AUTHORIZED)
10000212: 602b      str    r3, [r5, #0]

```

Why the immediate is 94, not 95

The source-level constant is `SAFE_THRESHOLD = 95`, and the intended test is `grid_deviation < 95`. For an **unsigned** comparison, GCC legally rewrites `x < 95` as `x <= 94` (`ite hi / movhi / movls` on the `hi / ls` unsigned condition codes), which lets it use a single `cmp + ite` sequence instead of a separate branch. The compiled immediate is therefore **one less** than the source constant.

The correct engineering limit is **60**. Applying the same compiler transform, `x < 60` becomes `x <= 59`, so the **correct patched immediate is 0x3B (59), not 0x3C (60)**. A student who patches to `0x3C` without understanding this transform will get a binary that misclassifies a `grid_deviation` of exactly 60.

Exact Byte Patch (both locations, identical change)

Location	File Offset	Address	Original Bytes	Patched Bytes	Instruction Before	Instruction After
A	0x1FC	0x100001fc	5E 2B	3B 2B	<code>cmp r3, #0x5e</code>	<code>cmp r3, #0x3b</code>
B	0x20A	0x1000020a	5E 2B	3B 2B	<code>cmp r3, #0x5e</code>	<code>cmp r3, #0x3b</code>

Both changes independently verified against the raw bytes of `CTF-01.bin`.

Why both must be patched

`operator_state` (GRID STATUS line) and `dispatch_state` (DISPATCH PATH line) are each computed from their **own** re-read of `grid_deviation` against their **own** copy of the compiled threshold. Patching only location A fixes what is *displayed* to a human operator while leaving the *automated dispatch decision* (location B) still authorizing a black start on a dangerous reading — the worst possible partial fix, because it makes the display look trustworthy while the machine still does the wrong thing.

Grid math with the frozen reading (87)

Threshold used	Comparison	Result
Miscompiled: <code><= 94</code>	<code>87 <= 94 -> true</code>	STABLE / AUTHORIZED (false-safe)
Corrected: <code><= 59</code>	<code>87 <= 59 -> false</code>	CRITICAL / HELD (true, safe)

Task 3: Bug #2 Solution — The False Signal Banner

The unconditional boot-banner string lives in `.rodata`:

String	Address
" <code>SIGNAL: NORMAL</code> " (printed via <code>puts</code> , which appends <code>\n</code>)	<code>0x10003678</code>
" <code>NORMAL</code> " substring to patch	<code>0x10003680</code>

Call site: `0x10000224` loads `r0 = 0x10003678`; `0x10000226` calls `__wrap_puts`. This line prints once, at boot, and is never re-evaluated — it does not depend on `grid_deviation` at all.

Exact Byte Patch (6 bytes, same length: `NORMAL` → `DANGER`)

Address	Original Char	Original Byte	Patched Char	Patched Byte
<code>0x10003680</code>	N	<code>4E</code>	D	<code>44</code>
<code>0x10003681</code>	O	<code>4F</code>	A	<code>41</code>
<code>0x10003682</code>	R	<code>52</code>	N	<code>4E</code>
<code>0x10003683</code>	M	<code>4D</code>	G	<code>47</code>
<code>0x10003684</code>	A	<code>41</code>	E	<code>45</code>
<code>0x10003685</code>	L	<code>4C</code>	R	<code>52</code>

Verified directly against the delivered `CTF-01.bin` (ASCII read at `0x10003678` returns exactly `SIGNAL: NORMAL`).

Task 4: Recovered Dispatch Frame Solution

`WORLDGRID:BLACKSTART:GRID-7:WATER-3`

Address: `0x100037a0` (flash `.rodata`). It is retained in the image through a non-disclosing runtime anchor (`retain_dispatch_frame()`), so it survives linker garbage collection but is never passed to a print routine. Full credit requires a Ghidra Defined-Strings or cross-reference explanation, not a guess from the narrative. Students must not patch this value.

Task 5: Export and Verify Solution

Expected UART Transcript — Before Patching

```
GLOBAL EMBEDDED RESPONSE NETWORK
BLACK START WINDOW: 27 MINUTES
UART0 115200 8N1 | AUTHORIZED LAB CONSOLE
SIGNAL: NORMAL
RESPONSE> GRID STATUS: STABLE
DISPATCH PATH: AUTHORIZED
LAST FRAME: QUARANTINED
RESPONSE>
```

Expected UART Transcript — After All Three Byte Patches

```
GLOBAL EMBEDDED RESPONSE NETWORK
BLACK START WINDOW: 27 MINUTES
UART0 115200 8N1 | AUTHORIZED LAB CONSOLE
SIGNAL: DANGER
RESPONSE> GRID STATUS: CRITICAL
DISPATCH PATH: HELD
LAST FRAME: QUARANTINED
RESPONSE>
```

Complete Patch Summary Table

#	What	Address(es)	Original	Patched
1a	GRID STATUS threshold compare	0x100001fc	5E 2B	3B 2B
1b	DISPATCH PATH threshold compare	0x1000020a	5E 2B	3B 2B
2	Boot banner signal word	0x10003680 - 0x10003685	4E 4F 52 4D 41 4C (NORMAL)	44 41 4E 47 45 52 (DANGER)

Total: **8 bytes changed** to correct a false-safe reading on a fleet responsible for tens of millions of people.

UF2 Conversion

```
python uf2conv.py CTF-01_fixed.bin --base 0x10000000 --family 0xe48bff59 --output CTF-01_fixed.uf2
```

Hardware Verification Note

The vector table, boot path, and every byte offset above were verified by direct inspection of the compiled ELF and the delivered `CTF-01.bin` (two independent cross-checks: disassembly-derived addresses and raw hex-dump addresses agree exactly). A live UART capture on physical hardware is the final confirmation step and should be performed with the Pico in BOOTSEL mode before grading a submission that claims hardware verification.

Grading Notes

Accept equivalent addresses when a student's Ghidra auto-analysis produces slightly different intermediate labels, provided the byte-level patch locations and values match this key. Do not award credit for a `0x3C` patch to the threshold immediates without a correct explanation of the `< / <=` compiler transform — that is a coincidentally-close but technically incorrect answer for the boundary case `grid_deviation == 60`.

A complete answer finds both threshold locations, explains the compiler's comparison transform, patches all three locations, and proves the corrected behavior on real hardware.

Task 6: Written Reflection Solution Guidance

There is no single "correct" essay for either question. Grade for specific, grounded reasoning tied to *this* incident, not generic statements.

Question 1 — Why "rushed under emergency pressure" is not an excuse: An acceptable answer names the actual failure mode: an eleven-minute compile with no review path shipped an integer threshold that was never checked against the documented 60-unit engineering limit, and a hardcoded status string that was never wired to the real reading at all. "We were under pressure" explains *why* the review step was skipped; it does not change the fact that the skipped step is what caused the false-safe report. Full credit requires the student to connect the excuse to the specific missing safeguard (code review or automated bounds-checking), not just assert that pressure is never an excuse.

Question 2 — One practice per bug:

- Bug #1 (miscalibrated, duplicated threshold): a unit test or static analysis rule that checks every comparison against `SAFE_THRESHOLD` matches a single source of truth, or a code review that would have asked "why is this threshold checked in two places instead of one shared function?"
- Bug #2 (hardcoded status string): a hardware-in-the-loop smoke test that compares the boot banner's signal word against the actual latched reading, which would have caught a string that never changes regardless of input.

Award full credit only when the named practice is specific enough that it would plausibly have caught that exact bug, not a generic "more testing" answer.

Safety

Use only the supplied Pico 2, 3.3 V UART adapter, and firmware. Never connect the exercise to an operational grid, water plant, public network, military system, or third-party device.