

---

# 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

## ⚠ WORLDGRID EMERGENCY INCIDENT ⚠



## INCIDENT BRIEFING

### Background

**WorldGrid Compact** is the emergency interconnection standard shared by three allied national grid operators. When any member's primary SCADA network goes dark, a fleet of small embedded relay nodes — call sign **GRID-7** — is the only thing standing between an orderly recovery and an uncontrolled cascade. Each relay node watches the last known grid frequency deviation, decides whether conditions are safe, and either **holds** the automatic black-start dispatch or **authorizes** it.

At 03:11 UTC, a coordinated cyberattack severed the primary SCADA uplink across the GRID-7 corridor and the WATER-3 aqueduct pumping stations that depend on it. With the network coordination center offline and three continents' worth of hospitals, rail systems, and water treatment plants running on backup power, WorldGrid's engineering team did the only thing they could: they rushed an emergency firmware build for the relay fleet and pushed it within **eleven minutes** of the attack being detected.

### The Disaster

The engineer who built that emergency image, **Dr. Elias Renner**, has not slept in thirty-one hours. He compiled the fix, ran a five-second bench test, and shipped it — because the alternative was leaving the relay fleet completely blind. It appears to work. The relay boots. It prints a status report. It reports **GRID STATUS: STABLE** and **DISPATCH PATH: AUTHORIZED**.

There is a problem: the frozen frequency reading latched at the moment communications were cut shows a deviation of **0.87 Hz** — nearly *50% beyond* WorldGrid's hard engineering limit of **0.60 Hz**. A deviation this large, if trusted, means the grid is nowhere near stable enough for an automatic black-start dispatch. If the fleet authorizes dispatch on a false "STABLE" reading, cascading generator trips will follow within minutes, and GRID-7 and WATER-3 will go dark for the second time — this time with no backup plan.

**Dr. Renner's rushed build has a bug. Multiple relay nodes are already reporting the same false-safe status. Nobody has found where in the compiled firmware the error lives, because the source code used for that emergency compile was overwritten by the next build fifteen minutes later and cannot be recovered.**

## The Only Surviving Evidence

One relay node — the training/verification unit — still holds the exact miscompiled image that shipped to the fleet. This binary, and this binary alone, is the only remaining copy of the emergency build. There is no source code. There is no build log. There is only the compiled image, a UART cable, and whatever a skilled embedded reverse engineer can prove by reading machine code.

## The Human Stakes

| Consequence if the false "STABLE" reading is trusted  | Scale                    |
|-------------------------------------------------------|--------------------------|
| Hospitals on generator backup past their fuel reserve | 214 facilities           |
| Water treatment and pumping stations losing pressure  | 3 aqueduct systems       |
| Rail corridors stranded mid-route                     | 6 national rail networks |
| Estimated population affected by cascading failure    | 40+ million people       |

### The options are:

1. **✗ Trust the fleet's reported status** — dispatch fires on a false reading, cascading failure follows within the hour.
2. **✗ Shut the entire relay fleet down** — buys time, but leaves 40 million people with no automated recovery path at all.
3. **REVERSE ENGINEER THE EMERGENCY BUILD** — find the exact miscompiled bytes, patch them, verify the corrected image on real hardware, and hand the fix to the field team so the *rest of the fleet* can be safely repatched before the next attempt.

## THE SHORTAGE

For years, the world treated embedded systems as invisible infrastructure. The engineers who could read a vector table, decode a Thumb branch, or patch a miscompiled constant directly in a stripped binary were never numerous enough. Tonight almost all of them are already in the field chasing other failures. **You are the reserve team.**

You were called in because you can do something Dr. Renner's exhausted team cannot do right now: read what the processor is actually doing, with no source code, no time for a rewrite, and no room for a guess.

 **TIME PRESSURE:** The field team is standing by to push your verified patch to the rest of the GRID-7 fleet. Every relay node still reporting a false "STABLE" status is one dispatch cycle away from disaster.

**AUTHORIZED LAB ONLY:** This challenge uses a supplied Pico 2 training relay and its exact miscompiled firmware image. Do not connect this exercise to a public network, an operational grid, a water utility, or any device you do not own or have explicit written authorization to test.

## What This CTF Tests

| Week | Concepts Tested                                                                                         |
|------|---------------------------------------------------------------------------------------------------------|
| 1    | RP2350 architecture, ARM Cortex-M33 registers, stack, flash/RAM, Thumb assembly, Ghidra static analysis |
| 2    | GDB connection, breakpoints, disassembly, register and memory inspection, UART observation              |
| 3    | Bootrom handoff, vector table, reset handler, startup code, XIP, Thumb-bit addressing                   |

## Part 1: Understanding the Relay Node

### GRID-7 Relay Hardware

| Component           | Connection     | Purpose                                     |
|---------------------|----------------|---------------------------------------------|
| Raspberry Pi Pico 2 | RP2350         | Runs the miscompiled emergency firmware     |
| UART TX             | GPIO 0         | Relay telemetry output                      |
| UART RX             | GPIO 1         | Reserved (no command parser is implemented) |
| SWD debug interface | Supplied probe | Authorized GDB inspection                   |

No LED, relay output, sensor, display, or other peripheral is part of this CTF. Every graded finding lives in flash ( `.rodata` / `.text` ) or SRAM, and is reachable with only the Weeks 1-3 toolset: Ghidra, GDB, and a UART monitor.

### UART Configuration

- Baud: `115200`
- Data: `8 bits`
- Parity: `none`
- Stop: `1`
- Logic: `3.3 V`

### Normal (Intended) Behavior

The relay should latch the frozen deviation reading, compare it against the **real** WorldGrid safety limit of **60** (0.60 Hz, encoded as an integer `x100` ), and report honestly:

```
+-----+
| Intended Relay Behavior |
| 1. Boot and initialize UART |
| 2. Print the boot identity and a signal-quality banner |
| 3. Compare the frozen 87 (0.87 Hz) reading against the 60 |
|    (0.60 Hz) safety limit |
| 4. 87 exceeds 60, so the grid is NOT stable |
| 5. Report GRID STATUS: CRITICAL and DISPATCH PATH: HELD |
| 6. Repeat the report once per second until conditions change |
+-----+
```

## Observed (Buggy) Behavior — What You Will See When You First Flash `CTF-01.uf2`

```
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>
```

This is exactly what Dr. Renner's team is seeing on the deployed fleet. It is wrong, and it is wrong in **two independent ways** inside the compiled binary. Do not assume the first readable sentence is the full truth — treat every printed line as evidence to be checked against the machine code, not as a fact on its own.

---

## How To Connect the Training Relay

- Pico 2 **GPIO 0** / **UART TX** -> USB-UART adapter **RX**
- Pico 2 **GPIO 1** / **UART RX** -> USB-UART adapter **TX**
- Pico 2 **GND** -> USB-UART adapter **GND**
- Use **3.3 V logic only**. Never connect a 5 V line to a Pico GPIO.
- Connect the supplied SWD probe according to its documented pinout.

The supplied image is `CTF-01.bin` (for Ghidra analysis) and `CTF-01.uf2` (for flashing). If your instructor supplies different filenames, record the actual filenames in your report.

---

## Part 2: The Miscompiled Firmware

You do not have the source code. It was overwritten fifteen minutes after the emergency build shipped. You have only the compiled image. Your job is to reverse engineer it with Ghidra, locate the defects, and patch the binary directly — exactly the way Dr. Renner's field team will need to patch the rest of the deployed fleet.

### What The Firmware Does

1. Initializes UART0 and stdio.
2. Reads a frozen grid-frequency-deviation reading that was latched in memory before communications were severed.
3. Compares that reading against a compiled-in safety threshold — **twice**, once for each independent status line it reports.
4. Prints a boot banner containing an unconditional signal-quality line.
5. Enters an infinite loop printing the grid classification and dispatch decision once per second.

## Bug Summary — What You Are Graded On

| Bug #         | Category                    | Severity        | Description                                                                                                                                                                                                                                      | Hint                                                                                                          |
|---------------|-----------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| <b>Bug #1</b> | Miscompiled safety constant | <b>CRITICAL</b> | The safety threshold used to classify the frozen reading was compiled far too permissive. It is used <b>twice</b> — once for the operator-facing status and once for the automated dispatch decision — and <b>both</b> copies must be corrected. | The real WorldGrid safety limit is 60 (0.60 Hz). Search for the wrong immediate value used in the comparison. |
| <b>Bug #2</b> | Hardcoded string literal    | <b>HIGH</b>     | The boot banner unconditionally prints a signal-quality word that does not reflect the actual reading, regardless of what the relay later reports.                                                                                               | The correct word describes the true state of a 0.87 Hz deviation against a 0.60 Hz limit — not "NORMAL".      |

**Important:** The replacement text for Bug #2 **must be the same length** as the original — patching a shorter or longer string will corrupt adjacent flash data.

## A Third Finding — Not a Bug, a Recovery Task

Somewhere in this image is the **quarantined black-start authorization frame** — the exact frame the relay is supposed to transmit to the regional dispatcher once a human operator confirms it is safe to proceed. It is never printed by the firmware. Recovering it (without patching anything) is required evidence for your final report.

## Part 3: Your Assignment

### Submission Document

Whenever a task asks you to **Document** or **answer**, write your answers in a single file named `CTF-01-Answers.md` (or `.txt`).

### Task 1: Setup and Initial Analysis

1. Create a new Ghidra project named `Black_Start_Investigation`.
2. Import `CTF-01.bin`.
3. Configure the language as **ARM Cortex 32-bit, little endian**.
4. Set the base address to `0x10000000`.
5. Run auto-analysis.

#### Document:

- A screenshot of the Ghidra **Import Results** or **Program Information** window showing the project name, processor settings, and base address.
- The address of `main()`.
- The address of the recurring status loop (the branch target that repeats once per second).
- The vector-table base, the initial stack pointer, and the reset-handler pointer as stored (note its Thumb bit) versus the actual instruction address.

## Task 2: Find and Patch Bug #1 — The Miscalibrated Safety Threshold

1. Find **both** locations where the frozen reading is compared against the miscompiled safety constant.
2. Document the exact address, the original instruction, and the original immediate value at each location.
3. Determine the correct immediate value. **Caution:** the compiler may not have encoded the raw threshold you expect — a strict "less than" comparison against an unsigned value is often optimized into a "less-or-equal" comparison against one less than the threshold. Show your reasoning.
4. Patch **both** locations in Ghidra.

### Questions to answer:

- Why must both locations be patched? What happens if you only patch one?
- Why is a false "STABLE" classification on an 0.87 Hz reading dangerous for an automated black-start dispatch?

## Task 3: Find and Patch Bug #2 — The False Signal Banner

1. Find the boot-banner string that unconditionally reports the wrong signal quality.
2. Document its address and the exact bytes that must change.
3. Patch the string, preserving its exact length.

### Questions to answer:

- Document the original vs. patched bytes, character by character.
- Why is a hardcoded, unconditional status word more dangerous than one that is at least computed from a (miscalibrated) reading?

## Task 4: Recover the Quarantined Dispatch Frame

1. Use Ghidra's Defined Strings (or a raw string search) to locate the hidden black-start authorization frame.
2. Document its address and explain why it is never transmitted by the current firmware.
3. Do **not** attempt to patch this value — it is evidence, not a bug.

## Task 5: Export and Verify

1. Export your patched binary as `CTF-01_fixed.bin`.
2. Convert it to UF2 format for the RP2350:

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

3. Flash `CTF-01_fixed.uf2` to your Pico 2 and capture the corrected UART output.
4. Confirm that the corrected image now reports **GRID STATUS: CRITICAL, DISPATCH PATH: HELD**, and the corrected signal-quality word — an honest, safe report instead of a false "all clear."
5. Build a summary table of every patch: address, original bytes, patched bytes, and a one-line description.

## Task 6: Written Reflection (short answers, 150 words or less each)

1. Why is "the build was rushed under emergency pressure" not an acceptable excuse for shipping a firmware defect that could trigger a cascading grid failure?
2. Name one concrete engineering practice (code review, static analysis, hardware-in-the-loop test, etc.) that would have caught **each** of the two graded bugs before this image ever reached the fleet.

## Submission Format

Submit a folder containing:

- `CTF-01-Answers.md` ;
  - screenshots or terminal transcripts;
  - `CTF-01_fixed.bin` and `CTF-01_fixed.uf2` ;
  - the original image hash.
- 

## Success Criteria

You complete the challenge when you can prove all of the following:

- You can explain how the RP2350 reaches the relay's code from reset.
  - You can locate and patch both copies of the miscalibrated threshold.
  - You can locate and patch the false signal-quality string without corrupting adjacent data.
  - You can export, convert, and flash a corrected image.
  - You can prove on real hardware that the corrected image reports the true, dangerous state instead of the false "all clear."
  - You can recover the quarantined dispatch frame as evidence.
- 

## Academic Integrity and Safety

By submitting this CTF work, you certify that:

1. You used only the supplied training relay, image, and lab interface.
2. You did not connect the challenge to a public network, an operational grid, a water utility, or any third-party device.
3. You understand that embedded reverse engineering and binary patching require explicit authorization in any real-world context.
4. You will report any discovered weakness responsibly to the course instructor.

The world is short on people who can do this work. Treat that responsibility seriously: verify before you patch, patch before you trust, and never confuse a clean-looking status line with a safe system.

---

## Reference Material

- ARM Cortex-M33 Technical Reference Manual
- RP2350 datasheet
- GDB documentation
- Ghidra documentation: <https://ghidra-sre.org/>