VCA-RE-101 — Reverse Engineering of Embedded Systems

Virtus Academy · Embedded Systems Reverse Engineering Stream

At a Glance

11 weeks · 4.0 units-equivalent · synchronous lecture + proctored laboratory Graduate-register · individual laboratory work · capstone with written report and oral defense

Course Description

A laboratory-intensive course teaching the complete methodology for recovering design information, firmware, and cryptographic material from commercial embedded systems — without access to vendor source, schematics, or documentation. Students progress from physical-layer PCB characterization, through serial-protocol and debug-interface analysis (SPI, I²C, UART, JTAG, SWD), to SoC boot-sequence recovery, firmware extraction, binary reverse engineering, patch-and-reflash, and the trust architectures that defend against these techniques. The capstone is an end-to-end reverse engineering of a real commercial cable modem (Motorola SURFboard SB6141) delivered as a peer-review-quality written report and oral defense. The course is appropriate preparation for offensive security research, defensive firmware assurance, hardware-trojan analysis, supply-chain integrity evaluation, and forensic device examination.

Learning Outcomes

On completion, graduates are able to:

1. Characterize an unknown printed circuit board by systematic physical measurement.
2. Identify SPI, I²C, UART, JTAG, and SWD protocols from observed bus waveforms.
3. Discover undocumented on-chip debug interfaces and verify discovered pinouts through independent methods.
4. Extract firmware from serial non-volatile memory using industry-standard programmers, with redundant-read cryptographic verification.
5. Analyze firmware images using binwalk, Ghidra, radare2, and QEMU.
6. Describe secure-boot and root-of-trust architectures, their threat models, and their practical attack surface.
7. Modify and reflash firmware to a target device while preserving recoverability to factory state.
8. Produce publication-quality technical reports suitable for USENIX Security, IEEE S&P, DEFCON, or coordinated-disclosure submission.

Schedule

Week	Topic	Laboratory
1	Foundations, threat models, laboratory setup	Bench qualification
2	Digital I/O, signal integrity, measurement	Logic analyzer fundamentals
3	Serial protocols: SPI, I²C, UART	Bus Pirate protocol exercises
4	JTAG and SWD	JTAG pinout discovery
5	SoC boot architectures and memory hierarchy	Boot sequence analysis
6	Non-volatile memory: NOR, NAND, eMMC	**Midterm practical exam**
7	Firmware extraction: in-circuit and ex-circuit	SB6141 flash dump
8	Firmware analysis: filesystems and signatures	SB6141 binwalk and extraction
9	Firmware analysis: executable reverse engineering	SB6141 binary analysis
10	Firmware modification and reflash	SB6141 patched firmware deployment
11	Trust architectures, secure boot, ethics	Live-device sniffing

Capstone oral defenses are held in Finals Week.

Assessment and Credential

Eleven laboratory exercises **55%** · midterm practical exam Week 6 **15%** · capstone written report **20%** · capstone oral defense **10%**. A minimum grade of B− on capstone components is required to earn the **VCA-RE-101 Certificate of Completion**. The program is independent; no affiliation with, or endorsement by, any government academy or university is claimed or implied.

VCA-RE-101 — Equipment, Software, and Reading

Virtus Academy · Embedded Systems Reverse Engineering Stream
Every student maintains a personal hardware workstation. A small inventory of shared instruments is provided per two-student team, and a larger course-level inventory is held by the program. All host software is free; the instructor supplies a prepared Docker container. Hardware prices verified April 2026.

Compute Path (choose one)

Option A — Personal VM-capable laptop. macOS, Windows 10+, or Linux with ≥ 8 GB RAM and a VM that supports USB passthrough (VMware, VirtualBox, UTM, or Hyper-V). Chromebooks, tablets, and laptops below 8 GB are not supported on this path.
Option B — Raspberry Pi 5 (rent or buy from the program). Pi 5 8 GB kit with instructor-baked `fwlab` SD card, PSU, and case — about **$250** to purchase outright, or a flat rental per cohort (refundable on return). Students interact with the Pi via any SSH-capable device — laptop, tablet, Chromebook, or older machine — or via a lab-provided keyboard-and-monitor station at the bench during proctored sessions.

Equipment per Student (Personal Workstation)

Item	Purpose	Approx. cost
Bus Pirate v3.6a with probe cable (SparkFun)	SPI / I²C / UART protocol work	$41
8-channel USB logic analyzer, 24 MHz, FX2LP-based	Bus waveform capture	$12 — $27
Digital multimeter, auto-ranging (basic consumer model)	Voltage and continuity measurement	$25
SOIC-8 test clip (Amazon generic, 25-series compatible)	Non-destructive 8-pin flash access	$10
SOP-16 test clip (Amazon generic, 25-series compatible)	Non-destructive 16-pin flash access	$10
Solderless breadboard and jumper-wire kit	Ad-hoc prototyping	$17
ESD mat and wrist strap kit	Electrostatic-discharge protection	$25
Kit subtotal (excluding compute path)		**≈ $140**

Equipment per Two-Student Team (Shared)

Item	Purpose	Approx. cost
JTAGulator (EXPLIoT — official maintainer since July 2025)	Debug-interface discovery	$249
Attify Badge (Attify Store) or FT2232H breakout	High-speed SPI flash programming	$48
Raspberry Pi Zero W (Adafruit)	Lab 4 JTAG training target	$15

Equipment per Course (Instructor / Facility)

Item	Purpose	Approx. cost
Hot-air rework station (858D class)	Desolder demonstrations	$80
Decommissioned Motorola SURFboard SB6141 (eBay), one per student	Capstone platform	$15 — $22 each
Unknown-board targets for the midterm practical	Proctored exam stock	instructor-sourced
Winbond W25Q80BV 1 MByte SPI flash breakout (Adafruit)	Reference chip, Weeks 2–3	$2
SparkFun TMP102 Qwiic temperature sensor breakout	I²C reference chip	$10
Macronix MX25L6406E SOIC-16 (eBay 5-pack)	NOR flash reference	$5 / 5
Lab power supply and USB hubs with isolated power rails	Workstation infrastructure	instructor-sourced

Software and Texts

All software is free. **Host-side:** `flashrom`, `openocd`, `pulseview`, `sigrok-cli`, `wireshark` (with `usbmon`), `minicom`, `screen`, Ghidra. **`fwlab` container (instructor-supplied):** `binwalk`, `squashfs-tools`, `jefferson`, `ubi_reader`, `hexedit`, `radare2`, QEMU user-mode and full-system.

VCA-RE-101 — Prerequisites and Student Profile

Virtus Academy · Embedded Systems Reverse Engineering Stream
VCA-RE-101 is calibrated for graduate-register academic rigor and proceeds at the pace of a working engineering course. It is intended for practicing engineers and analysts who will apply embedded-systems reverse engineering in professional work; it does not devote lecture time to remedial instruction in its prerequisite areas.

Ideal Learner Profile

- Early-career firmware or hardware engineer
- Defense-industrial-base hardware or supply-chain analyst
- Government civilian security researcher
- Advanced practitioner seeking professional calibration and a completion credential

Required Knowledge, Skills, and Abilities

Knowledge — what you must already understand

- Undergraduate-level electronic circuits: Ohm's law, voltage dividers, CMOS logic levels, pull-up / pull-down behavior, bypass capacitance
- Digital logic at the level of clock edges, logic-level thresholds, and voltage domains (5 V, 3.3 V, 1.8 V, 1.2 V)
- Operating-system fundamentals: processes, filesystems, permissions, and the Linux boot sequence at a conceptual level
- Number-system literacy: binary, hexadecimal, byte order
- Basic networking vocabulary at a recognition level (TCP/IP, MAC addressing, Ethernet framing)

Skills — what you must already be able to do

- Read and follow a component datasheet end to end
- Operate at a Unix shell with proficiency: navigation, redirection, piping, process management, remote access over SSH
- Use `git` to version-control text and small binary artifacts
- Read C source code (pointers, structs, `switch` statements); write small scripts in Python or Bash
- Operate a digital multimeter to measure voltage and verify continuity

Abilities — the dispositions the course assumes

- Methodological patience — hardware work is iterative; failure is a normal, expected, and instructive state
- Documentation discipline — lab notebook entries, instrument serial numbers, cryptographic hashes of every artifact
- Self-directed learning — the capstone is an independent, multi-week engagement
- Willingness to produce written technical prose at publication register

Skills Developed During the Course (Not Required on Entry)

Students do not need to arrive with any of the following. Each is taught in place.
- Soldering — the primary capstone extraction is non-destructive, using SOIC clips; a hot-air desolder demonstration is given but not graded
- ARM assembly reading — introduced in Week 4 and developed in Week 9
- SoC boot architecture — taught in Week 5
- Static binary analysis with Ghidra and radare2 — taught across Weeks 8 and 9
- Firmware carving, filesystem extraction, and repackaging — taught in Weeks 8 and 10
- Coordinated vulnerability disclosure practice — taught in Week 11

Who Is Not Well Matched

- Students who have never used a Unix terminal
- Students who find troubleshooting physical systems frustrating rather than engaging
- Pure web or application developers with no embedded or systems-programming exposure
- Students seeking a fast-paced certification badge rather than graduate-register instruction

Readiness Self-Check

Prospective students who can answer *yes* to all five of the following are ready to enrol:
1. Can I read an ARM or Cortex-class datasheet block diagram and identify which pins are power, ground, clock, and I/O?
2. Can I navigate a Linux filesystem, use `ssh`, `scp`, and `git`, and write a small script that parses a text file?
3. Can I read a simple C function — with pointers, structs, and a `switch` statement — and explain what it does?
4. Am I prepared to write a twenty-page technical report with figures, citations, and a reproducibility section?
5. Will I find it satisfying, rather than demoralizing, to debug a wiring error for two hours and emerge with a hash-verified firmware dump?