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?