Bypassing the Secure Boot: The Terminal Only Methodology.

LIVE 4H Masterclass with Amichai Yifrach

Secure Boot is not a binary security feature. It is a chain of trust assumptions implemented in software.

If you can observe the boot process, you can reconstruct the trust model.
If you can reconstruct the trust model, you can identify enforcement gaps.
If you can identify enforcement gaps, you can design a bypass.

This webinar teaches a structured, repeatable methodology for bypassing secure boot protections using only terminal access.
No hardware attacks.
No firmware dumping.
No fault injection.

Just logs, environment analysis, trust boundary mapping, and controlled exploitation.

Participants move step by step:
Boot sequence → Trust reconstruction → Attack surface discovery → Exploit design → Root shell.

A structured, hands-on methodology for bypassing Secure Boot using only terminal access. Attendees reconstruct the boot chain from log evidence, map trust boundaries, identify enforcement gaps, and design a controlled bypass that takes a lab VM from boot evidence to root shell. No firmware dumping, no fault injection, no hardware attacks. The method generalizes across U-Boot, shim, Coreboot, and embedded Linux appliances.

About TrainSec Academy

TrainSec is an online learning hub for current and aspiring cybersecurity pros who understand the need for excellence. Level up or change career using skill-expanding modular learning way. We turn beginners into experts and experts into masters. TrainSec is designed for those who are serious about being at the vanguard of the latest knowledge, skills and trends. We're already working with the best.

TrainSec Academy is managed by Scorpio Software LLC, 95 Newcomb Rd., Tenafly, NJ 07670, USA
info@TrainSec.net | https://TrainSec.net

trainsec
where pros train pros.

Schedule & Logistics

- **Event Date:** 18.05.26
- **Start Time:** 10:00 EDT
- **End Time:** 14:00 EDT
- **Timezone:** Eastern Daylight Time
- **Total Duration:** 4 hours
- **Platform:** Microsoft teams, Dedicated Discord channel.
- **Access**: Invite only, recordings will be available only for registered students.
- **Replay Access Duration:** Lifetime.
- **Price per seat:** **$**49
- **Refund Policy:** Refunds available up to 24 hours before the event.
- **Bonus**: Each student will receive $49 value in vouchers to TrainSec catalog.

Course objectives

After this masterclass, attendees will be able to:

1. Decompose a secure boot chain using only terminal evidence.
2. Build a trust boundary map from observable system behavior.
3. Identify enforcement gaps in real boot implementations.
4. Enumerate realistic terminal-only attack surfaces.
5. Design secure boot bypass strategies under constrained access.
6. Apply the methodology across different vendors and bootloaders (U-Boot, shim, Coreboot, embedded Linux appliances).

Prerequisites

- Comfortable with the Linux terminal and core command line tools.
- Familiarity with the boot process (UEFI, GRUB, kernel, initramfs, systemd).
- Basic understanding of kernel command line arguments and dmesg, /proc, /sys.
- Ability to run QEMU locally after the event for continued practice (optional for the live session).

Tools required:

- Modern web browser, to access the lab environment during the live session.
- SSH client, for optional direct access to the lab VM.
- QEMU installed locally, only if you want to run the downloadable VM image after the masterclass.

Format and details

- **Target audience:** intermediate to advanced. Designed for embedded security engineers, product security architects, firmware validation teams, red team engineers, secure boot implementers, pre-silicon validation engineers, and OEM security teams.
- **Duration:** 4 hours, including a short break and live Q&A.
- **Delivery:** live online, hands-on lab environment, full recording included.
- **Language:** English.
- **Price:** $49 per seat. Every registrant also receives a $49 TrainSec course voucher, usable on any course in the catalog.
- **Lab platform:** Ubuntu Secure Boot Challenge VM on QEMU, terminal-only access.

Course overview

Secure Boot is not a binary security feature. It is a chain of trust assumptions implemented in software.

If you can observe the boot process, you can reconstruct the trust model. If you can reconstruct the trust model, you can identify enforcement gaps. If you can identify enforcement gaps, you can design a bypass.

This masterclass teaches a structured, repeatable methodology for bypassing secure boot protections using only terminal access. No hardware attacks. No firmware dumping. No fault injection. Just logs, environment analysis, trust boundary mapping, and controlled exploitation.

The live lab runs on an Ubuntu Secure Boot Challenge VM. The method generalizes across U-Boot, shim-based Secure Boot, Coreboot, and embedded Linux appliances. The session closes with an end-to-end exploitation chain, from boot evidence to root shell, executed in a single continuous run.

This is not a lecture on how Secure Boot works. It is a structured engineering walkthrough of how to reason about and bypass Secure Boot implementations using only terminal access, from observable boot evidence through exploit design and down to a root shell.

trainsec
where pros train pros.

Modules

Module 1: Secure Boot as an Assumption Graph

Reframing the problem before touching the terminal. Verified does not mean continuously enforced. Enforcement is stage-bound. Trust is delegated. Logs are evidence of policy. Attendees leave able to model Secure Boot as a dependency graph of trust relationships, not as a single cryptographic gate.

Module 2: Boot Chain Reconstruction from Terminal Evidence

Using only the GRUB console, kernel command line, dmesg, /proc, /sys, initramfs traces, and systemd early logs, attendees identify each boot stage, detect signature verification routines, map bootloader-to-kernel transitions, surface fallback paths and recovery mechanisms, and flag debug or permissive indicators. Outcome: a structured boot chain diagram built entirely from live evidence.

Module 3: Trust Boundary Mapping

Converting logs into a security model. Where does verification start? Where does it stop? Which components are unsigned? What remains writable? When does kernel lockdown activate? When does policy enforcement shift from boot-time to runtime-only? Outcome: a trust boundary map of the lab system that generalizes to real-world platforms.

Module 4: Terminal-Only Attack Surface Enumeration

Under strict constraints (no firmware extraction, no binary patching outside the terminal, no hardware attacks), systematically enumerate GRUB entry manipulation, kernel parameter injection, init override, rdinit abuse, systemd debug paths, SELinux toggling, lockdown mode inspection, recovery console exposure, update mechanism weaknesses, rollback vectors, and writable-mount timing windows. Outcome: systematic enumeration instead of ad hoc testing.

Module 5: Vulnerability Clue Hunting in Logs

Reading logs as forensic evidence. Silent verification fallback, "continuing despite failure" logic, development configuration artifacts, test key remnants, weak error handling, policy enforcement inconsistencies, recovery mode exposure. Outcome: extracting exploit clues directly from boot behavior.

Module 6: Exploit Design Under Terminal Constraints

Transition from evidence to action. Design controlled bypass strategies using boot parameter manipulation, enforcement timing gaps, policy misconfiguration abuse, early userspace pivot, kernel parameter exploitation, and runtime enforcement weaknesses. Focus on exploitation reasoning, not memorized tricks.

Module 7: Runtime and Memory Reasoning

Within Ubuntu constraints, evaluate kernel lockdown, inspect ASLR, analyze /dev/mem exposure, review module loading restrictions and the capability model, and probe early mount manipulation. Outcome: understanding how runtime protections intersect with boot trust.

Final demonstration: From Boot to Root

Complete exploitation chain executed live. Evidence gathering, trust mapping, weakness identification, controlled manipulation, Secure Boot bypass, root shell. Attendees witness the full reasoning chain executed in a single continuous run on the lab VM.

Cross-bootloader methodology discussion

Generalizing the approach across U-Boot, shim-based Secure Boot, Coreboot, and embedded Linux appliances. The same methodology adapted to different implementations, with notes on where the evidence shifts and where it does not.

Deliverables (included with registration)

1. **Secure Boot Trust Mapping Framework.** Structured analysis template for any boot chain.
2. **Terminal-Only Secure Boot Assessment Checklist.** Field-ready exploitation workflow.
3. **Ubuntu Secure Boot Challenge VM.** Downloadable lab image from the TrainSec GitHub for continued practice.
4. **Masterclass recording.** Lifetime access, available after the event.
5. **$49 TrainSec course voucher.** Per purchaser, usable on any course in the catalog.

Instructor

Amichai Yifrach (Th3_H1tchH1ker) is a cybersecurity architect and researcher specializing in embedded systems, firmware security, and trust-chain analysis, with

over three decades of hands-on engineering and offensive research experience. He is the founder of CYMDALL and teaches structured, evidence-driven security reasoning through TrainSec. Recent published research includes the NVIDIA Jetson Orin Secure Boot bypass chain (CVE-2026-24154, CVE-2026-24153).

- **Handle:** Th3_H1tchH1ker
- **Instructor profile:** trainsec.net/amichai-yifrach

References and further reading

- Amichai's TrainSec article on the NVIDIA Jetson Orin Secure Boot bypass (CVE-2026-24154, CVE-2026-24153). The same methodology applied to a different platform.

Contact & support

- **Student support:** info@trainsec.net
- **Discord community:** discord.com/invite/qugcNyWdaU
- **LinkedIn:** linkedin.com/company/trainsec
- **X (Twitter):** x.com/TrainSec
- **Website:** trainsec.net