



OFFENSIVE IoT FIRMWARE REVERSING AND ANALYSIS

Expert-Led Cybersecurity Training · Beginner to Intermediate

COURSE OVERVIEW

IoT devices are everywhere – from smart home appliances and IP cameras to industrial controllers and automotive systems. The firmware running on these devices is often the weakest link in the security chain, yet analyzing it requires a unique blend of hardware knowledge, reverse engineering skills, and embedded systems expertise. This course bridges that gap.

You will learn the complete firmware analysis lifecycle: acquiring firmware through hardware interfaces (UART, JTAG, SPI flash) and software channels, unpacking and extracting filesystems, reverse engineering binaries compiled for ARM64 and MIPS architectures using IDA Pro / Ghidra / Binary Ninja, and discovering real vulnerabilities in production IoT firmware.

The course emphasizes practical, hands-on methodology. You will analyze firmware from real-world devices, write custom analysis scripts, emulate firmware for dynamic testing using QEMU and Firmadyne, and practice exploitation techniques specific to embedded environments. By the end of the course, you will have a repeatable methodology for assessing any IoT device's firmware security posture.

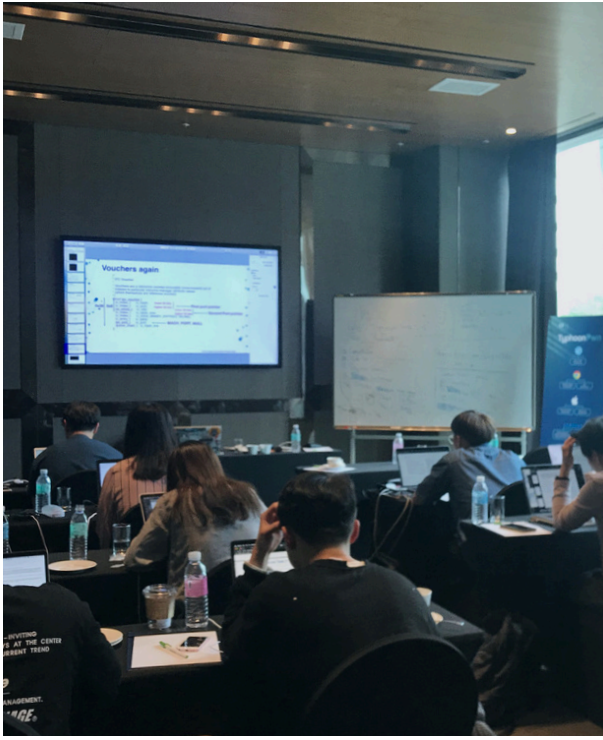
KEY LEARNING OBJECTIVES

- Extract firmware from IoT devices using UART, JTAG, SWD, and SPI flash interfaces
- Unpack and analyze firmware filesystems using Binwalk, Unblob, and EMBA
- Reverse engineer ARM64 and MIPS binaries using IDA Pro / Ghidra / Binary Ninja
- Write custom analysis scripts in IDAPython, Ghidra Python, and Binary Ninja API
- Emulate firmware for dynamic analysis using QEMU and Firmadyne
- Identify and exploit common IoT vulnerabilities: command injection, buffer overflows, hardcoded credentials
- Analyze bootloader security, secure boot chains, and firmware encryption schemes
- Assess IoT protocol security (MQTT, CoAP, BLE, Zigbee) and cloud backend interfaces
- Perform hardware-assisted debugging with JTAG and SWD
- Build a repeatable IoT firmware security assessment methodology



WHY SHOULD YOU TAKE THIS COURSE?

This is a hands-on course designed for intermediate and advanced security professionals. Students will master the full IoT firmware analysis lifecycle – from hardware-level firmware extraction through deep static analysis with IDA Pro / Ghidra / Binary Ninja to vulnerability discovery and exploitation across ARM64 and MIPS targets. All exercises use real device firmware and practical lab environments. You will leave with a repeatable assessment methodology and the scripting skills to automate firmware analysis at scale.



WHO SHOULD ATTEND?

This training program is designed for security researchers, penetration testers, embedded systems engineers, vulnerability analysts, and reverse engineers who want to build practical skills in IoT firmware security assessment.

WHAT WILL THE STUDENTS GET?

- Training Manual for the course
- A dedicated server with custom OS (Linux) for one month
- Lab setup (OVA) loaded with all course exercises
- Exercise material including solutions to all exercises
- A private dedicated channel where trainers will be available to answer your queries after the training

PREREQUISITE KNOWLEDGE

- Solid understanding of Linux command-line and system administration
- Basic knowledge of C/C++ programming and reading source code
- Familiarity with at least one disassembler or debugger (IDA Pro, Ghidra, Binary Ninja, GDB, or similar)
- Understanding of computer architecture fundamentals (registers, memory, stack)
- Basic networking knowledge (TCP/IP, HTTP, common protocols)
- Prior experience with embedded systems or IoT devices is helpful but not required

HARDWARE/SOFTWARE REQUIREMENT

- Laptop with a minimum of 16GB RAM and 60GB free hard disk space
- Administrative/root access on the system
- At least one of the following installed: IDA Pro, Ghidra, or Binary Ninja (trial licenses acceptable)

Detailed Course Setup instructions and Slack access will be sent a few weeks prior to the class

COURSE SYLLABUS

Module 1: IoT Security Landscape and Firmware Fundamentals

- IoT attack surface overview – device, network, cloud, mobile
- Firmware types: monolithic, RTOS-based, embedded Linux, bare-metal
- Common IoT SoC platforms and their security features
- ARM64 (AArch64) architecture primer – registers, calling conventions
- MIPS architecture primer – GP-relative addressing, delay slots
- Embedded Linux boot flow – U-Boot, kernel, rootfs, init systems
- Introduction to the lab environment and toolchain setup

Module 2: Hardware-Level Firmware Extraction

- Identifying debug interfaces on PCBs – UART, JTAG, SWD, SPI
- UART serial console access – baud rate detection, shell acquisition
- JTAG and SWD debugging – OpenOCD, J-Link, boundary scan
- SPI flash dumping – Flashrom, Bus Pirate, dedicated readers
- eMMC and NAND flash extraction techniques (ISP method)
- Software-based acquisition – OTA interception, vendor downloads
- Dealing with encrypted firmware – identifying encryption, key extraction

Module 3: Firmware Unpacking and Filesystem Analysis

- Firmware image structure – headers, partitions, filesystem offsets
- Unpacking with Binwalk – entropy analysis, signature scanning
- Unblob for modern firmware containers
- Filesystem types – SquashFS, JFFS2, UBIFS, CramFS, ext4, YAFFS2
- Automated firmware scanning with EMBA – SBOM, credential detection
- Manual filesystem triage – passwd/shadow, SSH keys, API tokens
- Identifying third-party libraries and SDK components
- Buildroot and Yocto-based firmware identification

Module 4: Reverse Engineering IoT Firmware with IDA Pro / Ghidra / Binary Ninja

- Loading ARM64 and MIPS firmware binaries – processor options, memory layout
- Navigating stripped binaries – function ID, cross-references, strings
- Decompiler workflows – Hex-Rays, Ghidra Decompiler, Binary Ninja HLIL
- Library identification – FLIRT, Function ID, signature libraries
- MIPS-specific analysis – GP-relative addressing, PIC relocations
- SVD Loader and peripheral register annotation – mapping MMIO addresses
- Loading bare-metal firmware – raw binary blobs, memory regions
- Comparing IDA Pro, Ghidra, and Binary Ninja – strengths & workflows

(Continued on the next page)

Module 5: Scripting and Automated Analysis with IDA Pro / Ghidra / Binary Ninja

- IDAPython scripting – finding dangerous functions, annotating MMIO
- Ghidra scripting with Python (Ghidrathon) and Java – batch analysis
- Binary Ninja Python API – BNIL, SSA form for data flow analysis
- Writing custom plugins for IoT specific analysis tasks
- Taint tracking from user input to dangerous sinks
- Cross-tool workflows – exporting annotations, sharing results
- AI-assisted binary analysis – LLMs for function naming

Module 6: Dynamic Analysis and Firmware Emulation

- Full-system firmware emulation with Firmadyne and FirmAE
- QEMU user-mode emulation for ARM64 and MIPS binaries
- QEMU system-mode emulation – booting full firmware images
- Remote debugging with IDA Pro / Ghidra / Binary Ninja – GDB server
- Intercepting and analyzing network traffic from emulated firmware
- Overcoming emulation challenges – NVRAM, peripheral stubs
- OFRAK for programmatic firmware modification and repacking

Module 7: IoT Firmware Vulnerability Discovery

- Command injection in web interfaces and CGI handlers
- Buffer overflows in embedded C services – stack, heap, format string
- Hardcoded credentials and backdoor accounts – systematic discovery
- Insecure firmware update mechanisms – missing signatures, downgrade
- Cryptographic weaknesses – weak keys, plaintext storage, custom crypto
- Authentication and authorization flaws in management interfaces
- Information leakage – debug endpoints, verbose errors, exposed APIs

Module 8: Exploitation Techniques for Embedded Targets

- Stack-based exploitation on ARM64 – ROP chains, gadget finding, ret2libc
- MIPS-specific exploitation – cache incoherency, delay slots, shellcode
- Bypassing exploit mitigations in embedded firmware
- ARM64 PAC and BTI – when present and how they affect exploitation
- Writing cross-architecture shellcode for ARM64 and MIPS
- Post-exploitation – persistence, lateral movement, credential harvesting

(Continued on the next page)

Module 9: Bootloader and Secure Boot Analysis

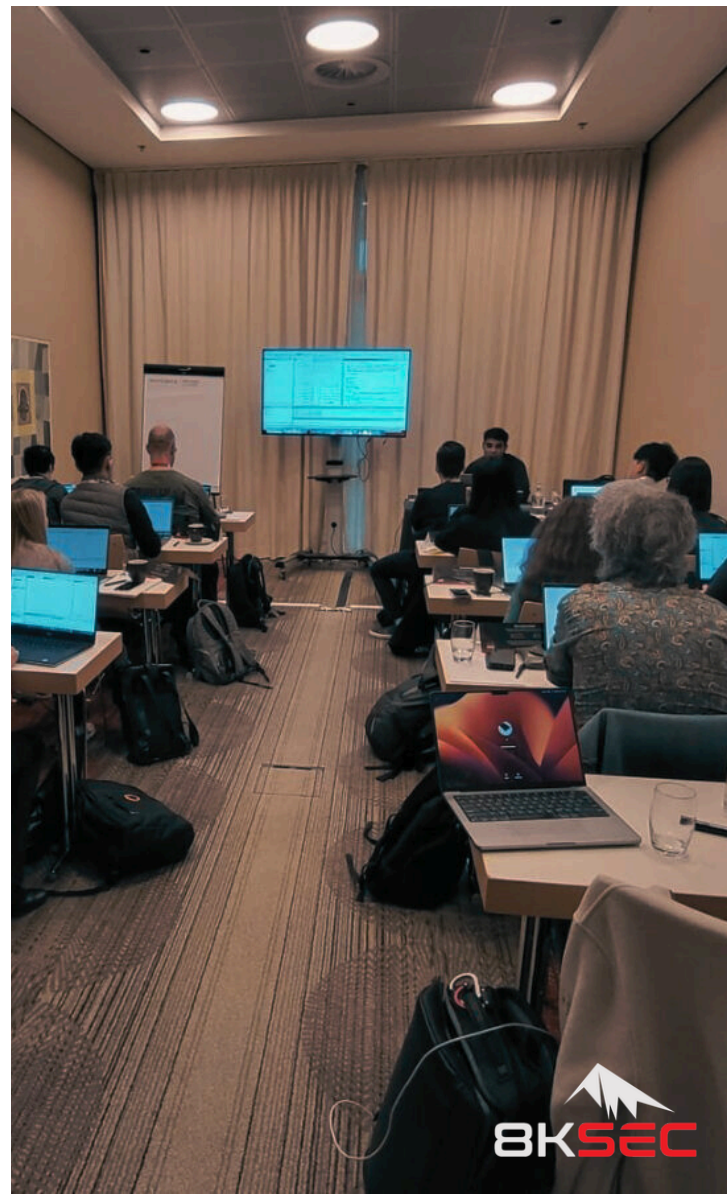
- U-Boot internals – environment variables, boot commands, shell access
- U-Boot exploitation – environment variable manipulation, memory primitives
- Secure boot chain analysis – hardware root of trust, chain of trust
- TF-A and OP-TEE analysis for ARM64 platforms
- Reversing bootloader code with IDA Pro / Ghidra / Binary Ninja
- Bypassing secure boot – fault injection, signed image manipulation
- Firmware encryption and decryption – AES-CBC, XOR, key recovery

Module 10: IoT Protocol and Network Security Analysis

- MQTT broker security – authentication bypass, topic enumeration
- CoAP protocol analysis – request forgery, resource discovery, DTLS
- BLE security – GATT characteristic abuse, sniffing, replay attacks
- Zigbee and Matter/Thread protocol security overview
- Cloud API backend analysis – device-to-cloud endpoints, auth flaws
- Firmware supply chain risks – SDK audit, ODM firmware, SBOM generation

Module 11: Assessment Methodology and Reporting

- Building a repeatable IoT firmware assessment methodology
- Threat modeling IoT devices – STRIDE applied to firmware and hardware
- OWASP IoT Top 10 – mapping findings to standard categories
- Writing effective firmware security assessment reports
- Responsible disclosure – vendor coordination, disclosure timelines
- Regulatory landscape – EU CRA, NIST guidelines, ETSI EN 303 645





About the company

8kSec is a foremost cyber security research company offering exceptional training and consulting services to aid clients in enhancing their security stance. Our experts possess extensive experience in delivering specialized cybersecurity training and consulting to multiple commercial and defense organizations across the United States, Europe, and the Middle East and North Africa region.

Get in touch

[8kSec.io](https://8ksec.io)

info@8ksec.io

