# Advanced Fuzzing Techniques for Software Testing Training Course

Professional Development Training Course Outline

| | | | |
|---|---|---|---|
| **Category** | Data Security | **Duration** | 10 Days |
| **Base Fee** | $3,000 USD | **Delivery Mode** | Online / On-site Hybrid |

## Course Introduction

Advanced Fuzzing is the crucial, cutting-edge technique for Automated Vulnerability Discovery in modern software. As system complexity increases driven by Cloud-Native Applications, Microservices, and intricate protocol stacks traditional security testing methods often fail to achieve sufficient Code Coverage and uncover deep-seated logical flaws. Advanced Fuzzing Techniques for Software Testing Training Course moves beyond basic mutation and black-box testing, diving into the core of Coverage-Guided Fuzzing, Hybrid Fuzzing, and advanced Program Analysis techniques like Dynamic Taint Analysis and Time Travel Debugging. Mastering these skills enables security researchers and developers to significantly Shift-Left vulnerability detection, finding and triaging critical Zero-Day Exploits faster and more efficiently than ever before.

This intensive, hands-on course empowers participants to build, optimize, and scale their own sophisticated fuzzing campaigns. Focusing on high-impact targets like Operating System Kernels, complex Protocol Parsers, and proprietary File Formats, the curriculum leverages state-of-the-art tools such as AFL++, LibFuzzer, WinAFL, and custom DBI frameworks. Participants will gain the practical expertise to instrument closed-source binaries, automate complex Crash Triage, and perform precise Root Cause Analysis, ensuring software robustness and resilience against the most advanced digital threats.

## Programme Curriculum

**Advanced Fuzzing Techniques for Software Testing Training Course**

**Introduction**

Advanced Fuzzing is the crucial, cutting-edge technique for Automated Vulnerability Discovery in modern software. As system complexity increases driven by Cloud-Native Applications, Microservices, and intricate protocol stacks traditional security testing methods often fail to achieve sufficient Code Coverage and uncover

deep-seated logical flaws. Advanced Fuzzing Techniques for Software Testing Training Course moves beyond basic mutation and black-box testing, diving into the core of Coverage-Guided Fuzzing, Hybrid Fuzzing, and advanced Program Analysis techniques like Dynamic Taint Analysis and Time Travel Debugging. Mastering these skills enables security researchers and developers to significantly Shift-Left vulnerability detection, finding and triaging critical Zero-Day Exploits faster and more efficiently than ever before.

This intensive, hands-on course empowers participants to build, optimize, and scale their own sophisticated fuzzing campaigns. Focusing on high-impact targets like Operating System Kernels, complex Protocol Parsers, and proprietary File Formats, the curriculum leverages state-of-the-art tools such as AFL++, LibFuzzer, WinAFL, and custom DBI frameworks. Participants will gain the practical expertise to instrument closed-source binaries, automate complex Crash Triage, and perform precise Root Cause Analysis, ensuring software robustness and resilience against the most advanced digital threats.

**Course Duration**

10 days

**Course Objectives**

Upon completion of this course, participants will be able to:

1. Master Coverage-Guided Fuzzing principles for high-performance, efficient Bug Hunting.
2. Implement and optimize AFL++ and LibFuzzer for both open-source and proprietary targets.
3. Harness Dynamic Binary Instrumentation (DBI) to instrument closed-source applications for coverage feedback.
4. Develop effective Fuzzing Harnesses and strategies for complex API Fuzzing and library functions.
5. Apply Grammar-Based Fuzzing and stateful models to find flaws in complex Network Protocols and file formats.
6. Understand and utilize Symbolic Execution and Concolic Testing in a Hybrid Fuzzing pipeline to reach deep code paths.
7. Perform in-depth Crash Triage using advanced techniques like Time Travel Debugging (TTD) on Windows and Reverse Debugging on Linux.
8. Leverage Dynamic Taint Analysis and Vulnerability Slicing for automated Root Cause Analysis (RCA) of crashes.
9. Scale Fuzzing Campaigns using cloud infrastructure and distributed frameworks for continuous DevSecOps integration.
10. Identify and exploit common bug classes uncovered by fuzzing, including Memory Corruption.
11. Adapt fuzzing techniques for specialized environments, including Kernel Fuzzing, Hypervisors, and Embedded Systems.
12. Design and deploy robust Corpus Minimization and Corpus Generation strategies for maximizing test efficiency.
13. Integrate fuzzing seamlessly into CI/CD Pipelines to achieve Continuous Security Testing and validate security patches.

**Target Audience**

1. Security Researchers and Penetration Testers.
2. Software Security Engineers.
3. Advanced QA Engineers and SDETs.
4. Reverse Engineers and Malware Analysts.
5. DevSecOps Practitioners.

6. Vulnerability Triage Specialists.
7. Professional Developers.
8. Cloud and Embedded Systems Engineers.

**Course Modules**

**Module 1: Foundations of Coverage-Guided Fuzzing**

- Deep dive into the Instrumentation process and mechanism of fuzzers like AFL and AFL++.
- Understanding the feedback loop.
- The principles of Mutational Fuzzing and Input Scheduling for path exploration.
- Harnessing Fundamentals.
- Case Study: Fuzzing a popular Open-Source Image Parser to uncover basic memory corruption bugs found by AFL.

**Module 2: Advanced Fuzzer Optimization and Tuning**

- Corpus Generation and Minimization strategies for maximizing test-case impact.
- Implementing Dictionaries and structural guidance for protocol-aware fuzzing.
- Fuzzer Parallelization and scaling on multi-core systems and cloud environments.
- Techniques for overcoming Input Checksums and magic numbers in closed-source targets.
- Case Study: Optimizing a large-scale fuzzing campaign targeting the Google Chrome V8 Engine, focusing on parallel execution and corpus management.

**Module 3: Fuzzing Closed-Source and Binary-Only Applications**

- Introduction to Dynamic Binary Instrumentation tools for coverage feedback.
- Deep dive into WinAFL for fuzzing Windows Userland binaries without source code.
- Writing specialized Fuzzing Harnesses for closed-source command-line tools and libraries.
- Techniques for defeating common fuzzer detection and anti-analysis mechanisms.
- Case Study: Using WinAFL to fuzz a proprietary Windows PDF reader or media player DLL, demonstrating DBI-based coverage.

**Module 4: Grammar-Based and Generational Fuzzing**

- Principles of Structure-Aware Fuzzing using grammars
- Defining Protocol Primitives and state models for network protocols
- Applying grammar-based fuzzing to complex data formats like JSON and XML schemas.
- Integrating LibProtocol models with coverage guidance for hybrid-structured testing.
- Case Study: Developing a Grammar Fuzzer for a financial API protocol, uncovering logical and parsing errors.

**Module 5: Symbolic and Concolic Execution**

- Fundamentals of Symbolic Execution and Constraint Solving
- Introduction to Concolic Fuzzing for deep path exploration.
- Combining CGF with symbolic execution to generate inputs that bypass complex checks.
- Practical limitations and optimization strategies for symbolic execution in large binaries.
- Case Study: Utilizing a Hybrid Fuzzer to bypass a deep, multi-stage input validation check in a target program, demonstrating the power of concolic path finding.

**Module 6: Advanced Crash Triage and Root Cause Analysis (RCA)**

- Advanced use of Memory Debuggers for crash detection.
- Mastering Time Travel Debugging on Windows and rr Debugger on Linux.
- Automating crash analysis with scripts for sorting, deduplication, and severity ranking.
- Techniques for crash reproduction and isolating the precise faulty code path.
- Case Study: Performing Root Cause Analysis on a Use-After-Free bug found by a fuzzer, tracing memory operations back to the allocation/free site using a TTD tool.

**Module 7: Dynamic Taint Analysis and Vulnerability Slicing**

- Principles of Taint Tracking to trace user-controlled input through a program.
- Using dynamic taint analysis to pinpoint the exact instruction causing a vulnerability.
- Vulnerability Slicing to isolate the minimal set of code and data dependencies for a crash.
- Applying taint-assisted techniques for targeted fuzzer seed generation.
- Case Study: Using a Taint Analysis Framework to track malicious input through a Web Browser Rendering Engine to an exploitable instruction.

**Module 8: Fuzzing Operating System Kernels**

- Understanding the unique challenges of Kernel Fuzzing
- Introduction to kernel interfaces and attack surface
- Tools and techniques for full-system emulation and snapshot fuzzing
- Handling crash reporting and non-reproducible kernel panics.
- Case Study: Reviewing the methodology and findings of the Syzbot project in uncovering critical vulnerabilities in the Linux Kernel Syscall Interface.

**Module 9: Fuzzing Network Services and Protocols**

- In-memory fuzzing techniques for high-speed testing of stateful server applications.
- Harnessing tools like Boofuzz for custom network protocol fuzzing.
- Handling complex network state, session management, and asynchronous I/O in fuzzers.
- Fuzzing the SSL/TLS handshake and custom cryptographic implementations.
- Case Study: Designing a stateful fuzzer to target a proprietary SCADA/ICS protocol, demonstrating sequence-aware input generation to trigger deep state bugs.

**Module 10: Advanced Heap and Memory Analysis**

- Deep dive into modern Heap Allocator designs
- Targeting advanced Memory Corruption bugs.
- Techniques for finding vulnerabilities in languages with garbage collection/memory safety via FFI or native components.
- Using fuzzer feedback to optimize search for specific memory safety bug classes.
- Case Study: Analyzing a Double-Free vulnerability in an open-source C library, detailing the heap exploitation primitives the bug creates.

**Module 11: WebAssembly (Wasm) and Browser Fuzzing**

- Understanding the attack surface of modern web browsers and scripting engines
- Techniques for fuzzing WebAssembly runtimes and binary format parsers.
- Using Fuzzing Generators for realistic, browser-specific test cases.
- Advanced crash analysis in complex multi-process browser architectures.

- Case Study: Examining a real-world Zero-Day vulnerability in a popular browser's JavaScript Engine or WebAssembly runtime that was discovered using advanced grammar/generator-based fuzzing.

**Module 12: Fuzzing in a DevSecOps/CI/CD Environment**

- Integrating fuzzing tools into continuous build and deployment pipelines.
- Automated reporting, tracking, and prioritization of fuzzing-found defects
- Setting up Continuous Fuzzing to ensure regression testing of all security patches.
- Best practices for measuring and demonstrating the security ROI of a fuzzing program.
- Case Study: Implementing an OSS-Fuzz style continuous fuzzing setup for a large, actively developed GitHub project, demonstrating the integration and automated bug reporting process.

**Module 13: Fuzzing Firmware and Embedded Systems**

- Introduction to the architectures and attack surfaces of Embedded Systems and IoT devices.
- Using full-system emulation and Firmware Rehosting for fuzzing.
- Harnessing techniques for low-level protocol and driver fuzzing without a full OS.
- Overcoming emulation hurdles.
- Case Study: Fuzzing a network parser within a real-world IoT Router Firmware, requiring the setup of a custom QEMU-based rehosting environment to achieve code execution.

**Module 14: Fuzzing AI/ML Systems and Data Formats**

- Fuzzing the input handlers and data transformation pipelines of Machine Learning Models.
- Techniques for finding security and reliability flaws in serialized ML data formats
- Applying advanced fuzzing to bespoke data formats and proprietary serialization schemes.
- Detecting Data Poisoning and Model Evasion vulnerabilities via tailored input fuzzing.
- Case Study: Developing a fuzzer to test the robustness of a component that processes user-uploaded data for an AI Image Generation service, looking for resource exhaustion or memory corruption.

**Module 15: Future of Fuzzing and Vulnerability Disclosure**

- Machine Learning for Fuzzing, and smarter input generation.
- The role of Formal Verification and static analysis in complementing advanced fuzzing.
- Ethical and legal considerations for vulnerability discovery and Responsible Disclosure.
- Building a professional Fuzzing Platform and framework for large organizations.
- Case Study: Analyzing the Disclosure and Patching Process for a high-profile CVE discovered by a major fuzzing campaign

## Training Methodology

This course employs a participatory and hands-on approach to ensure practical learning, including:

- Interactive lectures and presentations.
- Group discussions and brainstorming sessions.
- Hands-on exercises using real-world datasets.
- Role-playing and scenario-based simulations.
- Analysis of case studies to bridge theory and practice.
- Peer-to-peer learning and networking.
- Expert-led Q&A sessions.
- Continuous feedback and personalized guidance.

**Register as a group from 3 participants for a Discount**

**Send us an email: info@fineskilltrainingcenter.org or call** +254769199797

**Certification**

Upon successful completion of this training, participants will be issued with a globally- recognized certificate.

**Tailor-Made Course**

We also offer tailor-made courses based on your needs.

**Key Notes**

**a.** The participant must be conversant with English.

**b.** Upon completion of training the participant will be issued with an Authorized Training Certificate

**c.** Course duration is flexible and the contents can be modified to fit any number of days.

**d.** The course fee includes facilitation training materials, 2 coffee breaks, buffet lunch and A Certificate upon successful completion of Training.

**e.** One-year post-training support Consultation and Coaching provided after the course.

**f.** Payment should be done at least a week before commence of the training, to Fineskill Training Center account, as indicated in the invoice so as to enable us prepare better for you.

## Upcoming Scheduled Sessions

| Start Date | End Date | Location | Price |
|---|---|---|---|
| **21 Sep 2026** | 02 Oct 2026 | Online | $2,000 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $3,000 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |
| **21 Sep 2026** | 02 Oct 2026 | Physical | $0 |

Ready to enroll or request a customized program? Book online or contact us:

**Register Online Now**

Email: info@fineskilltrainingcenter.com | Website: www.fineskilltrainingcenter.com