ADVANCED FUZZING AND CRASH ANALYSIS
Trainer: Richard Johnson

Descripción general:

This training is designed to introduce students to the best tools and technology available for automating vulnerability discovery and crash triage with a focus on delivering a practical approach to applying this technology in real deployments.

Through an applied understanding of introductory program analysis and binary translation, techniques for finding various bug classes and methods for improved crash debugging will be discussed. We will take a deep dive into fuzzing, covering all aspects of this practical approach to finding bugs. As the most approachable and versatile of the available tools, the student will apply various fuzzing techniques to several real-world pieces of software. Students will learn strategies for analyzing attack surface, writing grammars, and generating effective corpus. We will explore in detail the latest innovations such as harnessing code coverage for guided evolutionary fuzzing and symbolic reasoning for concolic fuzzing.

We approach crash analysis through the lens of scriptable debuggers and program analysis. We will apply tools like reverse debugging and memory debuggers to assist in interactively diagnosing root cause of crashes. Then we will leverage the power of dynamic taint tracking and graph slicing to help isolate the path of user controlled input in the program and identify the exact input bytes influencing a crash. Lastly, we will look at possible ways to determine the impact of a vulnerability.

This class will focus on x86/x64 architecture and target file parsers, network parsers and browsers on both Windows and Linux environments.

This class is meant for **professional developers or security researchers looking to add an automation component to their software security analysis.** Students wanting to learn a programmatic and tool driven approach to analyzing software vulnerabilities and crash triage will benefit from this course.

Temario:

Analysis of generational and mutational fuzzing

- Attack surface analysis
- Effective mutation engines
- Effective corpus generation
- Protocol and file format grammars
- Crash detection

Fuzzing file and network parsers with coverage guided fuzzing

- Fuzz any Ubuntu/Debian package with AFL
- Modifying targets and writing harnesses with LibFuzzer
- Fuzzing closed source parsers with QEMU and Dyninst

Best practices for high performance fuzzing

- System configuration
- Corpus generation techniques
- Cross-fuzzing difficult parsers

Dynamic Binary Translation for Fuzzing and Triage

- Effectively instrument Linux and Windows with binary translation
- Introduction to Valgrind, Dr. Memory, and Address Sanitizer
- Introduction to PIN, DynamoRIO, and Dyninst internals
- Identifying hook locations with Debuggers and DBI
- Fuzzing kernels and other architectures with QEMU

Fuzzing parsers with WinAFL

- Optimizing harnesses for exported APIs
- Hooking closed source command line applications
- Deep hooks into private library functions with global state
- Fuzzing internal data streams in complex OLE objects

Fuzzing browsers with evolutionary grammar fuzzing

- Understanding grammars and object models
- Fuzzing object models with dynamic grammar fuzzing
- Improving grammar fuzzers with feedback metrics

Time Travel Debugging

- Introduction to time travel debugging
- Crash analysis with reverse debugging on Linux
- Crash analysis with reverse debugging on Windows

Taint assisted root cause analysis

- Introduction to dynamic taint analysis
- Taint slicing for root cause analysis

Symbolic and Concolic Execution

- Introduction to constraint solving
- Concolic execution for test case generation
- Hybrid fuzzing with concolic execution

Requerimientos:

Students should be prepared to tackle challenging and diverse subject matter and be comfortable writing functions in C/C++ and python to complete exercises involving completing plugins for the discussed platforms. Attendees should have basic experience with debugging native x86/x64 memory corruption vulnerabilities on Linux or Windows.

Hardware / Software Requirements:
Students should have the latest VMware Player, Workstation, or Fusion working on their machine.

Reservá tu lugar

Costo:

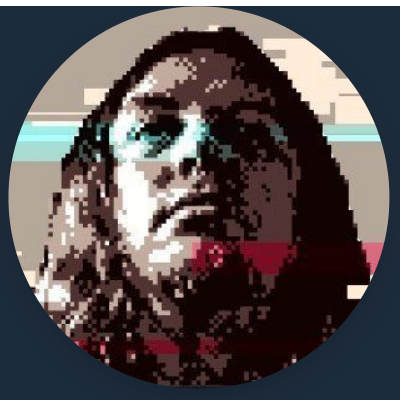
Consultá el precio de este training escribiendo a: capacitacion@ekoparty.org

Reservá tu lugar

CONSULTAS

Para realizar consultas sobre el training o alguno de sus beneficios, escribir a: capacitacion@ekoparty.org

Instructor: Richard Johnson



Richard Johnson es un especialista en seguridad informática con un enfoque en el análisis de vulnerabilidades de software. Actualmente, es investigador principal sénior de seguridad en Trellix y director de investigación en Fuzzing IO. Cuenta con más de 20 años de experiencia profesional, donde resalta su liderazgo en la industria de la seguridad de la información. Sus responsabilidades actuales incluyen la investigación de vulnerabilidades y el desarrollo de soluciones avanzadas de fuzzing e ingeniería inversa automatizada. Antes de Trellix, formó equipos de búsqueda de errores e investigación de seguridad para Oracle Cloud y Cisco Talos. D urante más de 15 años ha brindado capacitaciones y charlas en conferencias de primer nivel y en varios eventos líderes del mercado, incluyendo: Black Hat, Defcon, Hack in the Box, RECON y OffensiveCon. Richard fue cofundador de Uninformed Journal y ha participado en los comités de programa de USENIX WOOT, RECON y Toorcon.

Resources

- ARCHIVE: PAST EDITIONS
- CODE OF CONDUCT
- NEWSLETTER
- EKOMAGAZINE

About Ekoparty

- OVERVIEW
- CTFs & CHALLENGES
- HACKTIVITIES
- SPONSORS

organizacion@ekoparty.org