Buy tickets: https://romhack.io/buy-tickets/

ROMHACK TRAINING

CODE REVIEW TRAINING

Silvio Cesare

COURSE OBJECTIVES

To be able to discover software bugs and vulnerabilities in C Code.

TRAINING OUTCOMES

> Demonstrate understanding of C Bug Classes
> Demonstrate understanding of the main automated vulnerability discovery techniques
> Demonstrate understanding of some Linux heap exploitation techniques
> Demonstrate being able to patch bugs and vulnerabilities

WHO SHOULD ATTEND?

Developers, IT Professionals, Embedded Developers, OS Developers, Penetration Testers, Software Security Auditors/Analysts, Vulnerability Researchers, Software Exploitation Developers, and anyone interested.

ABOUT THE TRAINER

Dr Silvio Cesare is the Managing Director at InfoSect. He has worked in technical roles and been involved in computer security for over 20 years. This period includes time in Silicon Valley in the USA, France, and Australia. He has worked commercially in both defensive and offensive roles within engineering. He has reported hundreds of software bugs and vulnerabilities in Operating Systems kernels. He was previously the Director for Education and Training at UNSW Canberra Cyber, ensuring quality content and delivery. In his early career, he was the scanner architect and a C developer at Qualys. He is also the co-founder of BSides Canberra - Australia's largest cyber security conference. He has a Ph.D. from Deakin University and has published within industry and academia, is a 4-time Black Hat speaker, gone through academic research commercialization, and authored a book (Software Similarity and Classification, published by Springer).

WHAT TO BRING

> A Laptop (running their favorite OS) capable of connecting to wired and wireless internet
> An installed valid VMWare Workstation or VMWare Player

Training page: https://romhack.io/training/code-review/
RomHack Training 2023: https://romhack.io/training/

Buy tickets: https://romhack.io/buy-tickets/

ROMHACK TRAINING

WHAT WILL BE PROVIDED?

> 800 page coil bound lecture materials
> Access to VMs with laboratories
> InfoSect Swag

PARTICIPANT SKILL SET

Students taking Code Review should have an intermediate C Development background. They should have hands on experience in:

> C Coding Experience
> Linux

CLASS SYLLABUS [1]

Tuesday, 12 September 2023 - Day 1 (C Refresher)

Introduction to the Training
Class Schedule. Bio. Course Objectives. Student Introductions.

Lecture 1 – History of C
Use in industry. C as a systems language. Evolution of C. C features, Comparison to other languages. C Standards

Lecture 2 – Developing in C
Compilation with GCC and Clang. Build systems using Make.

Lecture 3 – Review of C Programming Basics
Variables. Expressions. Control Flow. Functions.

Lab 1 – Review of C Programming Basics Simple programming tasks in Linux.

Lecture 4 – Pointers, Strings, and Arrays Lab 2 – Pointers, Strings, and Arrays
Simple programming tasks in Linux.

Lecture 5 – Structures and Unions

Lecture 6 – Dynamic Memory Management Lab 3 – Dynamic Memory Management
Simple Programming tasks in Linux.

Wednesday, 13 September 2023 - Day 2 (Vulnerability Research)

Lecture 7 – Virtual Memory
Paged memory. How a process's virtual memory looks. Text, Data, Heap, Stack, Shared Libraries.

[1] Schedule of lectures on the specified days may be subject to changes

Lecture 8 – Debugging
Program Tracing. Debugger internals. Introduction to GDB.

Lecture 9 – Compiler Construction
Modern construction of an optimizing compiler with a front, middle, and backend.

Lecture 10 – Data Structures
A look at a variety of methods to implement linked lists.

Lecture 11 – Linux Heap Allocator Internals
A close-up look view of the current Linux ptmalloc implementation with a view to heap metadata corruption.

Lab 4 – ptmalloc Heap Metadata Corruption
Modern heap metadata corruption on Linux using CTF style challenges and TCache Poisoining, TCache House of Spirit and TCache Double Free attacks.

Lecture 12 – Fuzz Testing
White, Grey, and Black-Box Fuzzing. Mutation and Generative-based fuzzers. Seed Selection. Code Coverage. Guided fuzzing. Triaging. Bug Deduplication.

Lab 5 – Fuzzing and AFL
Using AFL on real targets to find previously unknown crashes.

Lecture 13 – Dynamic Memory Checkers
A look at Electric Fence and Address Sanitizer.

Lab 6 – Dynamic Memory Checkers
Using Electric Fence and ASAN on sample programs.

Lecture 14 – SMT Solving
A look at SMT Solving and use in code review to discover edge cases. Examples use SMT-Lib.

Lecture 15 – Symbolic Execution
Taking SMT Constraints and examples of finding bugs in toy programs. A look at how to use Klee to perform symbolic execution in C programs.

Thursday, 14 September 2023 - Day 3 (C Bug Classes)

We will look at bug classes and vulnerabilities in the various parts of C. Lab 7 – Insecure Coding
Throughout the day we will have many insecure toy program sources. The objective is to try to crash each of these programs by identifying and triggering the appropriate bugs.

Lecture 16 – Bugs in Preprocessor

Lecture 17 – Bugs in Declarations and Initialisation

Lecture 18 – Bugs in Expressions

Lecture 19 – Bugs in Floating Point

Lecture 20 – Bugs in Arrays

Lecture 21 – Bugs in Characters and Strings

Lecture 22 – Bugs in Memory Management

Lecture 23 – Bugs in Input Output

Lecture 24 – Bugs in Environment

Lecture 25 – Bugs in Signals

Lecture 26 – Bugs in Error Handling

Lecture 27 – Bugs in Miscellaneous

Lecture 28 – Bugs in Posix

Lecture 29 – Navigating the Linux Kernel
How to navigate the Linux Kernel source tree.

Lecture 30 – Bugs in Unix Kernels
Bug classes seen in Linux, FreeBSD, NetBSD, OpenBSD et al.

Lecture 31 – Code Review Strategies
Strategies for auditing small, medium, and large C projects.

Friday, 15 September 2023 - Day 4 (Real Programs, Recommendations, and Coding Guides)
On this day, we will look at how to develop secure C code. The recommendations are practical and can be implemented on existing code bases.

Lab 8 – Auditing Real Programs
We will be given access to a number of open source programs of various degrees of complexity. Some guidance will be provided and the task is to discover real bugs.

Lab 9 – Patching
Throughout the day of providing recommendations to fix code, we will go back over previous programs and make appropriate patches.

Lecture 32 – Fixes in Preprocessor

Lecture 33 – Fixes in Declarations and Initialisations

Lecture 34 – Fixes in Expressions

Lecture 35 – Fixes in Integers

Lecture 36 – Fixes in Floating Point

Lecture 37 – Fixes in Arrays

Lecture 38 – Fixes in Characters and Strings

Lecture 39 – Fixes in Memory Management

Lecture 40 – Fixes in Input Output

Lecture 41 – Fixes in Environment

Lecture 42 – Fixes in Signals

Lecture 43 – Fixes in Error Handling

Lecture 44 – Fixes in Miscellaneous

Lecture 45 – Fixes in Posix

Lecture 46 – Training Close
Final words.