AZERIA LABS

Advanced IoT Exploit Development for Arm 32-bit

Azeria Labs
Security Reviews and Advanced Training Courses
contact@azeria-labs.com

IoT EXPLOIT DEVELOPMENT FOR ARM32

COURSE OUTLINE

Arm internals, Assembly, Shellcoding, and Memory Corruption Vulnerabilities	**Day 1**
Exploit Development with ROP and the NX Exploit Mitigations	**Day 2**
Exploiting Format String Vulnerabilities, Infoleaks, and ASLR Bypass Techniques	**Day 3**
Introduction to Heap Exploit Engineering	**Day 4**

Course Abstract

Based around two real-world IoT targets that we will emulate, this course gets students to learn the process of building and debugging a memory-corruption exploit from scratch, bypassing exploit mitigations such as NX and ASLR along the way.

Our course begins with an introduction into the Arm architecture and assembly language, and how to build shellcode that can be used in exploits against Arm targets. Students then learn about the theory and practice of attacking memory-corruption exploits by finding and exploiting a stack-overflow vulnerability. Students then learn about exploit mitigations, what they are, and how to bypass them, and how to take over the process using both ret2libc, as well as complex ROP-chains to run in-memory only shellcode directly in the target process.

The **second day** of the course focuses on exploiting two real-world routers, including the process of how to emulate, debug and trigger vulnerabilities on real-world devices, and how to adapt exploits from one target to work on a different target, even when the devices use identical library versions.

The **third day** of the course provides a deeper study of exploit categories and techniques to make exploits reliable. Students will cover vulnerability discovery and use of “information leaks” to stabilize memory-corruption exploits, and learn about the ASLR and stack canary exploit mitigations, and how to exploit format-string vulnerabilities to bypass these mitigations.

For **four-day courses**, the final day is a deep-dive into the process of heap exploitation, and using heap vulnerabilities to construct exploitation primitives that can be engineered together to build powerful and reliable exploits, bypassing NX, ASLR and GCC’s in-built exploit mitigations. We begin with a review of how the glibc heap works, and begin writing an exploit against a network service containing a heap linear buffer overflow. Students will learn how to turn this buffer overflow into a reliable relative read exploit primitive to bypass ASLR, how to construct arbitrary read primitives to search target memory for useful binaries, and how to exploit and construct malicious vtables to fully take control of the target device.

REQUIREMENTS & PREREQUISITES

Software Requirements

Azeria Labs Training Courses are designed to minimize the amount of installation and set-up costs for students at the start of the course. For this reason, target devices and development environments are provided as virtual machines.

Students will need to bring their own laptop, and **before the class** will need to:

- Install the latest version of VMware Player/Workstation/Fusion on their laptop
- Download and extract the virtual machine via a link provided to students several days before the course begins

In order to ensure a usable experience when running the hands-on exercises, laptops should have at least the following specifications:

- One or more USB ports for file transfer
- Minimum 8GB RAM and minimum 20GB of available disk space

Student Prerequisites

- Able to read simple code written in the C programming language
- Familiar with writing and editing basic scripts written in the Python programming language
- Willing to dive into a Linux debugging environment
- Willing to read and interact with Arm assembly language for several hours
- Have a strong desire to learn, and be ready to think outside-of-the-box
- Comfortable with troubleshooting the laptop’s host operating system
- Comfortable with administering Linux from the command line
- Able to use VMware to access the hands-on labs

What Students get

- 626 total slides (including solution slides) full of technical graphics
- Comprehensive 90-page workbook
- Lab VM with all the challenges including solution scripts

IoT exploit development for ARM32

Day 1

ARM 32-bit Architecture

- Architectures and Microarchitectures
- Overview of different ARM processor types
- RISC CPU features
- Privilege Modes
- TrustZone Extension
- Memory model, memory types and segments
- Arm Instruction Set

Writing Arm Shellcode

- Overview of Shellcode writing process
- Tracing and issuing system calls
- Determining syscall numbers
- ARM calling convention
- Translation to assembly
- Shellcode null-bytes
- Converting assembly into hex string

Lab: Writing Shellcode

Memory Corruption Vulnerabilities

- Understanding stack frames
- Looking at vulnerable functions
- Visualization of a vulnerable function call
- “Stack corruption”: what it is, and what it means
- Executing shellcode on the stack without NX
- “Gadgets” and introduction to gadget hunting

Lab: Exploiting a simple stack-buffer overflow

Exploit Mitigations

- Introduction to the NX Exploit Mitigation
- Why this kills our previous simple exploit
- Defeating this mitigation using ROP
- Invoking system() via ROP to execute a command

Lab: Exploiting a stack-buffer overflow via Ret2Libc

IoT exploit development for ARM32

Day 2

Return Oriented Programming

- Advanced ROP engineering
- Gadgets that break your ROP chain
- Workaround through register initialization
- Noisy vs stealth ways of exploitation

Mprotect ROP Chains

- Introduction to Mprotect
- Building mprotect ROP chains
- How to avoid null-bytes in your exploit
- How to make stack addresses page aligned

First target: D-Link DIR890

- Process of physically gaining access to the device
- Process of extracting and emulation the firmware
- Debugging the vulnerable service

Lab: Bypassing memory-write issue to take over PC
Lab: invoking system to get a reverse shell
Lab: D-Link mprotect ROP chain

Second target: Tenda router

- Process of physically gaining access to the device
- Process of extracting and emulation the firmware
- Debugging the vulnerable service

Lab: Bypassing memory-read issue to take over PC
Lab: Tenda mprotect ROP chain

Day 3

Address Space Layout Randomization

- Introduction to ASLR
- Format String vulnerabilities
- Information leaks
- How to bypass ASLR with information leaks

Lab: Write exploit to bypass ASLR with a format string vulnerability

Stack Canaries

- Introduction to stack canaries
- How to leverage infoleaks to bypass Stack Canaries

Lab: Write exploit to bypass Stack Canaries with an information leak

Day 4

Heap Exploit Engineering

- Introduction to the heap
- Heap layout
- Malloc and free
- Corrupting virtual tables
- Categories of heap vulnerabilities
- Concepts and exploitation techniques

Lab: Use-after-free Heap Exploit
Lab: Trigger the heap overflow
Lab: Build and use a relative read exploit primitive
Lab: Build arbitrary read primitive & bypass ASLR
Lab: Build a complete exploit chain