# Software Security 1

**Recitation**

> 11.12.2024

## Agenda

- Code Sanitizers (ASan)
- C++ Vtables

## Address Sanitizer (ASan)

- Runtime memory error detector
- Finds common memory bugs
- Minimal performance impact
- Easy to integrate

## Asan Examples (Live Demos):

- Unfreed Memory
- Buffer Overflow
- Use-After-Free

## Drawbacks of using ASan

- Performance overhead (2x-3x slowdown)
- Increased memory usage (3x-4x)
- Larger binary sizes
  - Debug symbols
  - Instrumentation code
- Some false positives possible
- Not suitable for production deployment

## Detour 1: Other Code Sanitizers

- **KSan (Kernel Sanitizer)**
    - Thread race detection
    - Kernel memory issues
    - Used in Linux Kernel
- **UBSan (Undefined Behavior Sanitizer)**
    - Integer overflow
    - Null pointer dereference
    - Array bounds checking
    - Division by zero

## Detour 2: Uses of Code Sanitizers in Fuzzers

- **AFL++ uses ASan** for memory corruption detection
- **libFuzzer** integrates with all sanitizers
- **Honggfuzz** works well with ASan/UBSan
- **OSS-Fuzz** uses multiple sanitizers
  - Memory sanitizer
  - UB sanitizer
  - Coverage sanitizer

Sanitizers help fuzzers find bugs faster!

> Links mentioned during recitation:
> [Jazzer Code Sanitizers][Fuzzing CSGO maps]

## C++ Refresher [Live Demo]

## Inheritance & Polymorphism [Live Demo]

## Virtual Tables - Introduction

- Compiler-generated tables
- Enable dynamic dispatch
- Key to C++ polymorphism
- Can be exploited!

## Virtual Table [Live Demo]

## Virtual Table Attack: Fake VTable [Live Demo]

- Create fake vtable
- Redirect object's vtable pointer
- Execute arbitrary function