Secure Software Development – SSD

Tutorial - Tools 1

Andreas Kogler, David Schrammel

12.10.2022

Winter 2022/23, www.iaik.tugraz.at/ssd

Agenda

- BUGs
- Compilers
- Fuzzing
- Assembly Tools
- Q & A
- https://extgit.iaik.tugraz.at/sase/practicals/2022/exercise2022-demos

Agenda

- BUGs
- Compilers
- Fuzzing
- Assembly Tools
- Q & A
- `https://extgit.iaik.tugraz.at/sase/practicals/2022/exercise2022-demos`

Agenda

- BUGs
- Compilers
- Fuzzing
- Assembly Tools
- Q & A
- https://extgit.iaik.tugraz.at/sase/practicals/2022/exercise2022-demos

Agenda

- BUGs
- Compilers
- Fuzzing
- Assembly Tools
- Q & A
- https://extgit.iaik.tugraz.at/sase/practicals/2022/exercise2022-demos

Agenda

- BUGs
- Compilers
- Fuzzing
- Assembly Tools
- Q & A
- https://extgit.iaik.tugraz.at/sase/practicals/2022/exercise2022-demos

- BUGs
- Compilers
- Fuzzing
- Assembly Tools
- Q & A
- `https://extgit.iaik.tugraz.at/sase/practicals/2022/exercise2022-demos`

BUGs We Will See Today

- Race Conditions (SLP/OS)
  - Threads **modify** shared variable
- Buffer Overflows (INFOSEC)
  - Fill array with to many elements
  - C-strings are terminated with **0**
- Integer Overflows (INFOSEC)
  - Unsigned integers *wrap* arround
  - For n bit addition: $c = (a + b) \bmod 2^n$
- `https://godbolt.org/z/W9qrPjKeq`

BUGs We Will See Today

- Race Conditions (SLP/OS)
  - Threads **modify** shared variable
- Buffer Overflows (INFOSEC)
  - Fill array with to many elements
  - C-strings are terminated with **0**
- Integer Overflows (INFOSEC)
  - Unsigned integers *wrap* arround
  - For n bit addition: $c = (a + b) \bmod 2^n$
- `https://godbolt.org/z/W9qrPjKeq`

BUGs We Will See Today

- Race Conditions (SLP/OS)
  - Threads **modify** shared variable
- Buffer Overflows (INFOSEC)
  - Fill array with to many elements
  - C-strings are terminated with **0**
- Integer Overflows (INFOSEC)
  - Unsigned integers *wrap* arround
  - For n bit addition: $c = (a + b) \bmod 2^n$
- https://godbolt.org/z/W9qrPjKeq

BUGs We Will See Today

- Race Conditions (SLP/OS)
  - Threads **modify** shared variable
- Buffer Overflows (INFOSEC)
  - Fill array with to many elements
  - C-strings are terminated with **0**
- Integer Overflows (INFOSEC)
  - Unsigned integers *wrap* arround
  - For n bit addition: $c = (a + b) \bmod 2^n$
- `https://godbolt.org/z/W9qrPjKeq`

Compilers

- How to enable warnings:
  - `-Wall`
  - `-Wextra`
  - `-Wpedantic`
  - `-Weverything` (`clang` only)
- `-Wno-***`
  - Disables a certain category of warnings (not recommended)
- `-Werror`
  - Treat warnings as errors
- `https://godbolt.org/z/qK31ed4Ph`

- How to enable warnings:
  - `-Wall`
  - `-Wextra`
  - `-Wpedantic`
  - `-Weverything` (`clang` only)
- `-Wno-***`
  - Disables a certain category of warnings (not recommended)
- `-Werror`
  - Treat warnings as errors
- `https://godbolt.org/z/qK31ed4Ph`

- How to enable warnings:
  - `-Wall`
  - `-Wextra`
  - `-Wpedantic`
  - `-Weverything` (`clang` only)
- `-Wno-***`
  - Disables a certain category of warnings (not recommended)
- `-Werror`
  - Treat warnings as errors
- `https://godbolt.org/z/qK31ed4Ph`

- Thread Sanitizer `-fsanitize=threads`
- Address Sanitizer `-fsanitize=address`
- Integer Sanitizer `-fsanitize=integer`

- Thread Sanitizer `-fsanitize=threads`
- Address Sanitizer `-fsanitize=address`
- Integer Sanitizer `-fsanitize=integer`

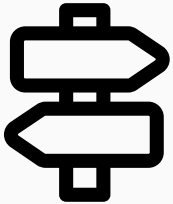
- Thread Sanitizer `-fsanitize=threads`
- Address Sanitizer `-fsanitize=address`
- Integer Sanitizer `-fsanitize=integer`

Demo - Sanitizers

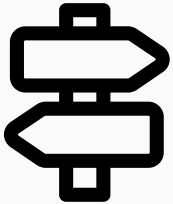
- Thread Sanitizer `-fsanitize=threads`
- How to fix?
  - locking / lock-free algorithms

- Thread Sanitizer `-fsanitize=threads`
- How to fix?
  - locking / lock-free algorithms

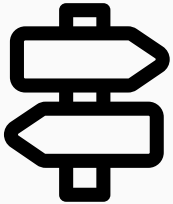
- Thread Sanitizer `-fsanitize=threads`
- How to fix?
  - locking / lock-free algorithms



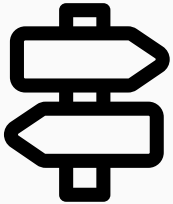
- Address Sanitizer `-fsanitize=address`
- How to fix?
  - check for out-of-bounds accesses
  - buffer sizes
  - use-after-free



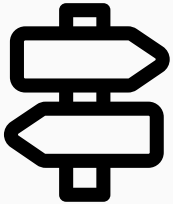
- Address Sanitizer `-fsanitize=address`
- How to fix?
  - check for out-of-bounds accesses
  - buffer sizes
  - use-after-free



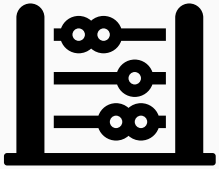
- Address Sanitizer `-fsanitize=address`
- How to fix?
  - check for out-of-bounds accesses
  - buffer sizes
  - use-after-free



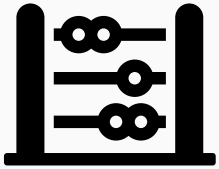
- Address Sanitizer `-fsanitize=address`
- How to fix?
  - check for out-of-bounds accesses
  - buffer sizes
  - use-after-free



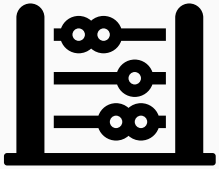
- Address Sanitizer `-fsanitize=address`
- How to fix?
  - check for out-of-bounds accesses
  - buffer sizes
  - use-after-free



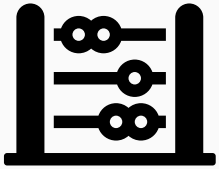
- Integer Sanitizer `-fsanitize=integer`
- How to fix?
  - check data types
  - also verify the ranges of your data
  - overflow aware operations like `__builtin_add_overflow`



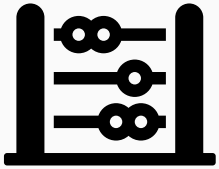
- Integer Sanitizer `-fsanitize=integer`
- How to fix?
  - check data types
  - also verify the ranges of your data
  - overflow aware operations like `__builtin_add_overflow`



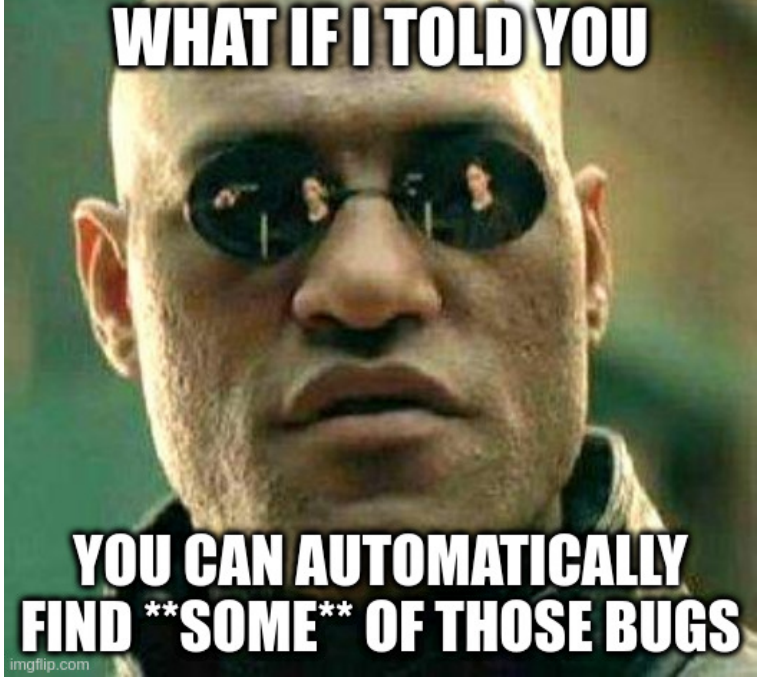
- Integer Sanitizer `-fsanitize=integer`
- How to fix?
  - check data types
  - also verify the ranges of your data
  - overflow aware operations like `__builtin_add_overflow`



- Integer Sanitizer `-fsanitize=integer`
- How to fix?
  - check data types
  - also verify the ranges of your data
  - overflow aware operations like `__builtin_add_overflow`



- Integer Sanitizer `-fsanitize=integer`
- How to fix?
  - check data types
  - also verify the ranges of your data
  - overflow aware operations like `__builtin_add_overflow`


WHAT IF I TOLD YOU
YOU CAN AUTOMATICALLY FIND **SOME** OF THOSE BUGS
imgflip.com

Fuzzing

- American Fuzzy Lop (AFL/AFL++)
- Run a program with *pseudo* random input
- Check if the program crashed
- Check if a sanitizer triggered

- American Fuzzy Lop (AFL/AFL++)
- Run a program with *pseudo* random input
- Check if the program crashed
- Check if a sanitizer triggered

- American Fuzzy Lop (AFL/AFL++)
- Run a program with *pseudo* random input
- Check if the program crashed
- Check if a sanitizer triggered

- American Fuzzy Lop (AFL/AFL++)
- Run a program with *pseudo* random input
- Check if the program crashed
- Check if a sanitizer triggered

- Recompile with AFL
- Instrumentation code, similar to sanitizer
- **Automatically** generate new inputs
- **Manually** check crashes

- Recompile with AFL
- Instrumentation code, similar to sanitizer
- **Automatically** generate new inputs
- **Manually** check crashes

- Recompile with AFL
- Instrumentation code, similar to sanitizer
- **Automatically** generate new inputs
- **Manually** check crashes

- Recompile with AFL
- Instrumentation code, similar to sanitizer
- **Automatically** generate new inputs
- **Manually** check crashes

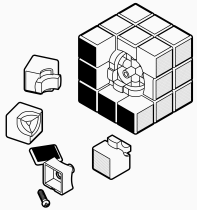
Demo - Fuzzing

- Cannot detect logic errors
- The input test cases *should* be representative
- Could take a long time to trigger a given BUG

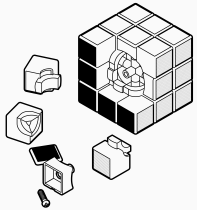
- Cannot detect logic errors
- The input test cases *should* be representative
- Could take a long time to trigger a given BUG

- Cannot detect logic errors
- The input test cases *should* be representative
- Could take a long time to trigger a given BUG

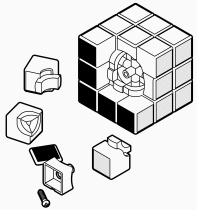
Assembly Tools



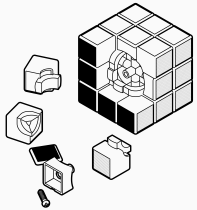
- What if the source code is **unavailable**
- Disassembler (e.g., Radare2, Ghidra, IDA)
- Decompilers (e.g., Ghidra, IDA)
- My personal daily driver: Cutter



- What if the source code is **unavailable**
- Disassembler (e.g., Radare2, Ghidra, IDA)
- Decompilers (e.g., Ghidra, IDA)
- My personal daily driver: Cutter



- What if the source code is **unavailable**
- Disassembler (e.g., Radare2, Ghidra, IDA)
- Decompilers (e.g., Ghidra, IDA)
- My personal daily driver: Cutter



- What if the source code is **unavailable**
- Disassembler (e.g., Radare2, Ghidra, IDA)
- Decompilers (e.g., Ghidra, IDA)
- My personal daily driver: Cutter

Demo - Cutter

Questions?