20875 Software Engineering – Assignment 2

Due Friday, December 5th 2025, 23:59

The assignment consists in finding and fixing bugs in the Intel XED project. The project is open-source and its code is hosted at `github.com/intelxed/xed`. Specifically, we are targeting `v2025.06.08` (commit `dc6bdb`), which is the latest release as of this writing. **Edit:** In the meantime, XED `v2025.11.23` was released (commit `722cd23`), you can use that version as well.

XED is an open-source library for encoding and decoding `x86_64` instructions. In other words, it can:

1. convert assembly language to machine code (i.e. function as an assembler), and
2. convert machine code to assembly language (i.e. function as a disassembler),

for the Intel architecture. It is mostly used for the latter (2), and this is what we will focus on. The library also comes with a command-line tool `xed` which can read an executable or object file in the ELF format, and produce the corresponding assembly code. Your task is to find **two distinct** bugs in the XED project, explain them, and propose fixes.

Write your answers directly on Blackboard (under "Assignment 2") by the end of December 5th.

Bugs. A bug can be a crash, an assertion failure, any type of undefined behavior invocation, a memory leak, or a logic error (e.g. producing a wrong output). Error messages are not bugs. For example, if we feed XED with invalid machine code and it returns an error message, this is the correct, intended behavior.

For two bugs to qualify as distinct, the underlying error must be located in distinct parts of the code. Moreover, if two functions perform the same task with minor differences (e.g., one for the 64-bit architecture and the other for the legacy 32-bit architecture) and they contain the same error, it still counts as a single bug.

Questions. For each one of the two bugs you find, answer the following questions.

Q1. [1 mark] Create or find a file (of maximum ~~100000 bytes~~ 5000000 bytes) that triggers that bug when passed to the command-line utility `xed`. Upload that file. (As an alternative, you can upload C code that calls into the XED library to trigger the bug.)

Q2. [1 mark] Explain the *consequences* of the bug and where they happen in the source code of XED (in which file and at which line).

Example: "A crash happens at line 1234 in `example/example.c` in the function `cleanup()` because the pointer variable `p` is `NULL` and the expression `p[i]` dereferences a `NULL` pointer."

Q3. [2 marks] Determine the *causes* of the bug. Specifically, explain the chain of events that happens, starting from the particularities of the input file, and ending with the bug consequences described above.

Example: "When the `max_index` field is negative in the input file, the allocation at `example/example2.c` line 5678 in the function `initialize()` attempts to allocate a negative amount, fails, and returns `NULL`. This `NULL` pointer is then stored in `struct DATA *d`, specifically in the `d->pointer` field. When `cleanup()` is called, that pointer is stored in `p`, yielding the `NULL` pointer dereference."

Be as general as possible in your description of the range of circumstances that can lead to the bug. For example, "when the `max_index` field is negative" is more general than "when `max_index` is -1".

Be concise. The objective is that a reader familiar with XED understands just enough to then propose a bug fix. Between 2 and 5 sentences should suffice.

Q4. [2 marks] Propose a *fix* for the bug. Explain your strategy to address the root cause of the problem. You can include proposed changes to the source code (using `diff -u` or `git diff`). As an alternative, detail your proposed changes in plain English.

The fix must be general, and correctly address the underlying problem.

Be concise. The objective is that a reader familiar with XED understands just enough to then apply your proposed code changes. Between 1 and 3 sentences should suffice.

Bonus question. One bonus mark if the root cause of one (or both) of the two bugs is in the main XED library (i.e. in `src/*/*.[ch]`), as opposed to the tests, `xed` utility, and other examples.

Rules. This is an individual assignment. You are allowed (and encouraged) to talk about the assignment with your classmates. However, you must find input that triggers bugs yourself, on your own computer. You must also write your answers on your own. As a consequence, you will be able reproduce and explain all bugs upon request. **Please do not contact the developers of XED.**

Hints: There may be technical hurdles involved when compiling XED and looking for bugs. We will share as many hints as possible to overcome such issues, at this address:
`https://www.poirrier.ca/courses/softeng/hw2/`