Анализ бинарного файла, скомпилированного afl компилятором - Lab4

Matthew Rusakov m.rusakov@innopolis.university SD-03

May 2025

1 Предисловие

Анализ проводился с использованием AFL++, поскольку он обнаружил больше уникальных сбоев за намного меньшее количество времени.

Также я добавил сантитайзеры ASan и UBSan для обнаружения большей части уязвимостей

2 Найденные проблемы с AddressSanitizer

2.1 Heap-buffer-overflow

Найдена проблема: Обнаружено чтение за пределами выделенной области памяти в процессе освобождения JSON-структур.

Местоположение: Функция `json_value_free_ex`, строка 200 исходного кода.

Анализ и причина: Ошибка возникает при итеративном освобождении элементов объекта JSON. Проверка границ массива значений (`values`) выполняется некорректно из-за возможного несоответствия между декларируемым количеством элементов (`length`) и фактическим размером массива. Дефект проявляется при обработке специально сформированных входных данных, вызывающих повреждение метаданных объекта.

2.2 Corrupted pointer

Найдена проблема: Выявлены случаи обращения по невалидным указателям во время освобождения памяти.

Местоположение: Функция `json_value_free_ex`, строка 179.

Анализ и причина: Санитайзер ASan зафиксировал попытки обращения к памяти по адресам, не принадлежащим процессу. Анализ стека вызовов указывает на возможную порчу указателей в структуре JSON-объекта до момента его освобождения. Наиболее вероятная причина - отсутствие проверки целостности указателей перед операцией освобождения.

2.3 Heap-buffer-overflow при парсинге

Найдена проблема: Обнаружен off-by-one error при обработке Unicode-последовательностей.

Местоположение: Функция `json_parse_ex`, строка 338.

Анализ и причина: Ошибка возникает при парсинге escape-последовательностей формата \uXXXX вблизи конца входного буфера. Логика обработки предполагает наличие 4 байт после управляющего символа, но не выполняет корректную проверку доступности этих байт в буфере. Преинкремент указателя (`*++state.ptr`) приводит к чтению за границами выделенной памяти.

3 Undefined Behavior Sanitize

3.1 Неопределённое поведение

Найдена проблема: Необрабатываемое исключение SIGILL на валидных входных данных.

Местоположение: Инициализация парсера перед циклом в `json_parse_ex`.

Анализ и причина: Санитайзер UBSan детектировал неопределённое поведение на этапе инициализации парсера. SIGILL (Illegal Instruction) свидетельствует о серьёзном повреждении состояния программы, вероятно вызванном: (1) некорректными операциями с указателями, (2) нарушением выравнивания данных, или (3) переполнением целочисленных типов. Точная причина требует дополнительного исследования с отладочной сборкой.

4 Control Flow Integrity (CFI)

- Обнаруженные сбои проявлялись как `free(): invalid pointer`
- Анализ подтвердил ошибки управления памятью, а не нарушения потока управления
- Подчеркнул необходимость использования ASan для диагностики проблем памяти

5 Выводы

Фаззинг-тестирование выявило несколько уязвимостей безопасности памяти:

- Критические проблемы управления кучей при очистке JSON-структур
- Ошибки проверки границ при обработке Unicode
- Уязвимости целостности указателей при операциях с памятью

Список литературы

[1] GitHub Link: https://github.com/MattWay224/reverse-engineering-course В этом репозитории можно найти все лабы и информацию про каждое задание в каждой лабе