Отчёт по результатам фаззинга с помощью libfuzzer - Lab7

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

May 2025

1 Инструмент

Для тестирования использовался **libFuzzer** с подключенными санитайзерами:

- AddressSanitizer (ASan) - для обнаружения ошибок работы с памятью
- UndefinedBehaviorSanitizer (UBSan) - для выявления неопределённого поведения

2 Найденные проблемы

В ходе тестирования обнаружены следующие критические уязвимости:

1. **Heap-buffer-overflow** при обработке неполного токена "nul"
2. **Heap-use-after-free** при обработке массива с пробелом
3. **Bad-free** при обработке пустой строки
4. **Undefined behavior** (применение ненулевого offset к null pointer)
5. **Повторный heap-use-after-free** при обработке пустого массива "[]"

3 Анализ

3.1 Heap-buffer-overflow

Проблема возникает при проверке токена "null". Парсер не проверяет границы входного буфера при последовательном чтении символов:

```
if ((end - state.ptr) < 3 || *(++ state.ptr) != 'u' ||
    *(++ state.ptr) != 'l' || *(++ state.ptr) != 'l')
```

3.2 Heap-use-after-free

Проблема возникает из-за неправильного управления памятью при обработке массивов:

- Память освобождается слишком рано (в функции new_value)
- Последующий код продолжает использовать указатель на освобождённую память
- Особенно проявляется при обработке пробельных символов после закрывающей скобки

3.3 Bad-free

Новая обнаруженная проблема связана с некорректным освобождением памяти при обработке пустых строк:

- Парсер пытается освободить память, которая не была выделена через malloc/calloc
- Адрес 0x502000004a2f находится прямо перед выделенной областью (1 байт до 1-байтной области)
- Проблема проявляется на входе "¨¨"(двойные кавычки без содержимого)
- Ошибка возникает в функции json_value_free_ex при освобождении строковых значений

3.4 Undefined behavior

- Тип: Неопределённое поведение (UB)
- Местоположение: json_fuzz.c:442:65
- Описание: Применение ненулевого смещения к нулевому указателю
- Условия возникновения:
  - Проявляется при обработке специально сформированных JSON-структур
  - Связано с неправильной проверкой указателей при работе с памятью
- Риски:
  - Непредсказуемое поведение программы
  - Потенциальная уязвимость для атак типа "undefined behavior exploitation"

4 Выводы

Как и в предыдущих анализах получилось опознать проблемы с парсером: heap-buffer-overflow, heap-use-after-free, undefined behavior, однако в прошлых репортах не получалось обраружить bad-free

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