# Автоматическое обнаружение использования неинициализированных значений в рамках полносистемной эмуляции

Никита Белов, ИСП РАН
zodiac@ispras.ru

# План

- Введение, основные термины.
- Проблемы отладки и существующие решения для них.
- Обнаружение использования неинициализированных значений: теория и реализация.
- Тестирование.
- Результаты.

# Введение

**Эмулятор** – это комплекс программно-аппаратных средств, предназначенный для реализации интерфейса и функциональности одной вычислительной системы на другой, таким образом, чтобы эмулируемое поведение как можно ближе соответствовало поведению оригинальной системы.

**Полносистемный эмулятор** моделирует все аппаратное обеспечение вычислительной системы и позволяет запускать набор программного обеспечения для реальных систем без внесения дополнительных изменений.

# Введение

Эмуляторы позволяют отлаживать следующие программы:

- базовые системы ввода-вывода (BIOS) и расширяемые интерфейсы прошивки (UEFI);
- загрузчики;
- ядра операционных систем.

**QEMU** —

- бесплатная система с открытым исходным кодом,
- эмулятор аппаратного обеспечения различных платформ,
- использует динамическую двоичную трансляцию.

# Проблемы отладки

**Реальная аппаратура.**

- Нет доступа ко всему контексту выполнения (регистры и память).
- Дорогой аппаратный отладчик.

**Эмуляторы.**

- Поддержка различных платформ.
- Весь контекст выполнения.
- Не существует инструментов для автоматического поиска ошибок при работе с памятью в низкоуровневых программах.

# Существующие решения

**Memcheck:**

- обнаруживает множество ошибок при работе с памятью;
- работает посредством динамической двоичной трансляции;
- используется для проверки пользовательских приложений.

**Android Memory Checker Component**:

- обнаруживает множество ошибок при работе с памятью;
- разработан на базе эмулятора QEMU;
- используется для проверки пользовательских приложений внутри эмулируемой системы.

# Задача

**Метод обнаружения использования неинициализированных значений при полносистемной эмуляции в QEMU:**

- разработать метод хранения и отслеживания состояния регистров и ячеек памяти гостевой системы;
- сформулировать критерии обнаружения использования неинициализированных значений и уведомления об ошибках;
- реализовать и протестировать разработанный метод в QEMU.

# Теневая память

**Теневая память содержит информацию об исходной памяти.**

- Используется для хранения информации об инициализированности значений по соответствующим адресам физической памяти гостевой системы.
- Работа с теневой памятью осуществляется инструментальным кодом, выполняющемся на основной машине.
- Операции над значениями теневой памяти реализуются аналогично операциям из Memcheck.

# Инструментирование внутреннего представления

Существует два метода инструментирования кода: внутри гостевой среды и вне ее.

Инструментирование выполняется во время трансляции гостевого кода во внутреннее представление до его исполнения основной машиной.

Инструментальный код выполняется основной машиной вместе с соответствующим кодом гостевой машины.

Для выставки инструментального кода используется программный интерфейс TCG.

# Проблема уведомления об ошибках

**Две стратегии уведомления:**

- сразу,
- при наступлении определенных условий.

# Проблема уведомления об ошибках

```
struct S { int i; char c; };
struct S s1, s2;
s1.i = 42;
s1.c = 'z';
s2 = s1;
```

# Проблема уведомления об ошибках

```
mov eax,DWORD PTR [esp+0x8]
mov edx,DWORD PTR [esp+0xc]
mov DWORD PTR [esp],eax
mov DWORD PTR [esp+0x4],edx
```

# Обнаружение использования неинициализированных значений

**Критерии уведомления об использовании неинициализированных значений:**

- неинициализированное значение является адресом для операций загрузки и сохранения значений из памяти,
- выполняется условный переход на основании неинициализированного значения,
- выполняется переход на неинициализированный участок памяти.

# Обновление значений теневой памяти

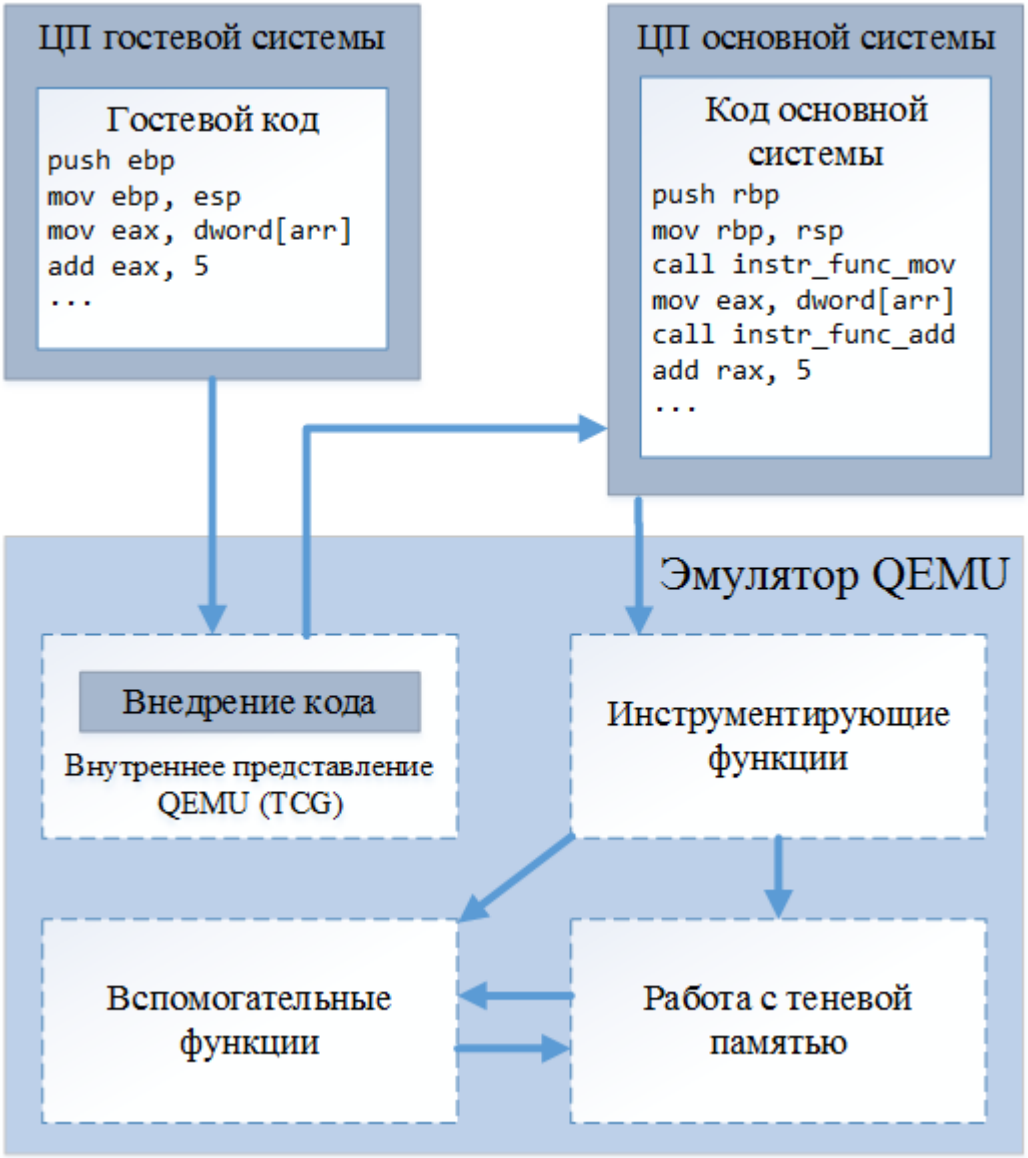
**Введем правила обновления значений теневой памяти после выполнения исходной операции.**

- Константы.
- Операции копирования данных.
- Арифметико-логические операции.
- Операции изменения размера.
- Операции, влияющие на флаги.

# Внедрение кода

- Функции-помощники в адресном пространстве основной машины.
- Вызываем перед каждой инструментируемой операцией.
- Работают с теневой памятью и проверяют наступление критериев уведомления.

# Схема программной системы



# Тестирование

Разработанный в QEMU метод был протестирован с использованием в качестве гостевой операционной системы Baremetal OS.

Все случаи использования неинициализированных значений были успешно обнаружены.

# Тестирование

| **Тип неинициализированного значения** | **Пример сообщения об ошибке** |
|---|---|
| Адрес для операций работы с памятью | `Load from undefined address 0x00000000.` |
| Операнд в условном переходе | `Jump based on undefined value from 0x00123456.` |
| Адрес для перехода | `Uninitialized jump to 0x00123456.` |

# Результаты

- Разработан способ инструментирования внутреннего представления и метод автоматического обнаружения использования неинициализированных значений при полносистемной эмуляции.
- Разработанный способ реализован в полносистемном эмуляторе QEMU.
- Реализованный метод был протестирован на гостевой системе архитектуры x86 и показал свою корректную работу на всех случаях.

# Спасибо за внимание!