LinuxTesting.org

Инструменты для динамического анализа модулей и драйверов ядра Linux

Цыварев Андрей,
tsyvarev@ispras.ru

Institute for System Programming of the Russian Academy of Sciences

Тестирование драйверов Linux

- Сценарий реализован в программе, работающей в пользовательском пространстве
- При проверке корректности используются данные, доступные из пользовательского пространства

“Скрытые” свойства драйвера

- Отсутствие утечек памяти
- Устойчивость к системным сбоям
- другие(напр., отсутствие состояний гонки)

Инструменты для анализа “скрытых” свойств драйверов

- **Kmemleak**
- **Kmemcheck**
- **Fault injection**
- Kprobe-based: SystemTap, LTTng
- FTrace

Kmemleak

- “Наблюдает” за низкоуровневыми функциями выделения/освобождения памяти
- Обнаруживает утечку после потери указателя на память
- Нередки false positives, false negatives
- Проверка утечек памяти занимает много времени, нельзя проверить отдельный драйвер
- Не все механизмы выделения памяти отслеживаются (page_alloc, ioremap)
- Не всегда включено в ядро по-умолчанию (может потребоваться пересборка ядра)

Kmemcheck

- “Наблюдает” за низкоуровневыми функциями выделения/освобождения памяти
- Сильно замедляет работу всей системы(~50x)
- Обычно не включено в ядро по-умолчанию

Fault injection

- Симуляция ошибок для нескольких подсистем: выделения памяти, дисковые операции
- Расширение списка функция для симуляции, расширение сценариев симуляции требует изменения кода ядра

SUSE YES tests: "API Swapping"

Перехват вызовов функций реализован на основе подмены имен функций импортируемых объектным файлом модуля ядра.

На основе такого перехвата реализовано несколько вспомогательных инструментов для тестирования.

SUSE YES tests: "API Swapping"

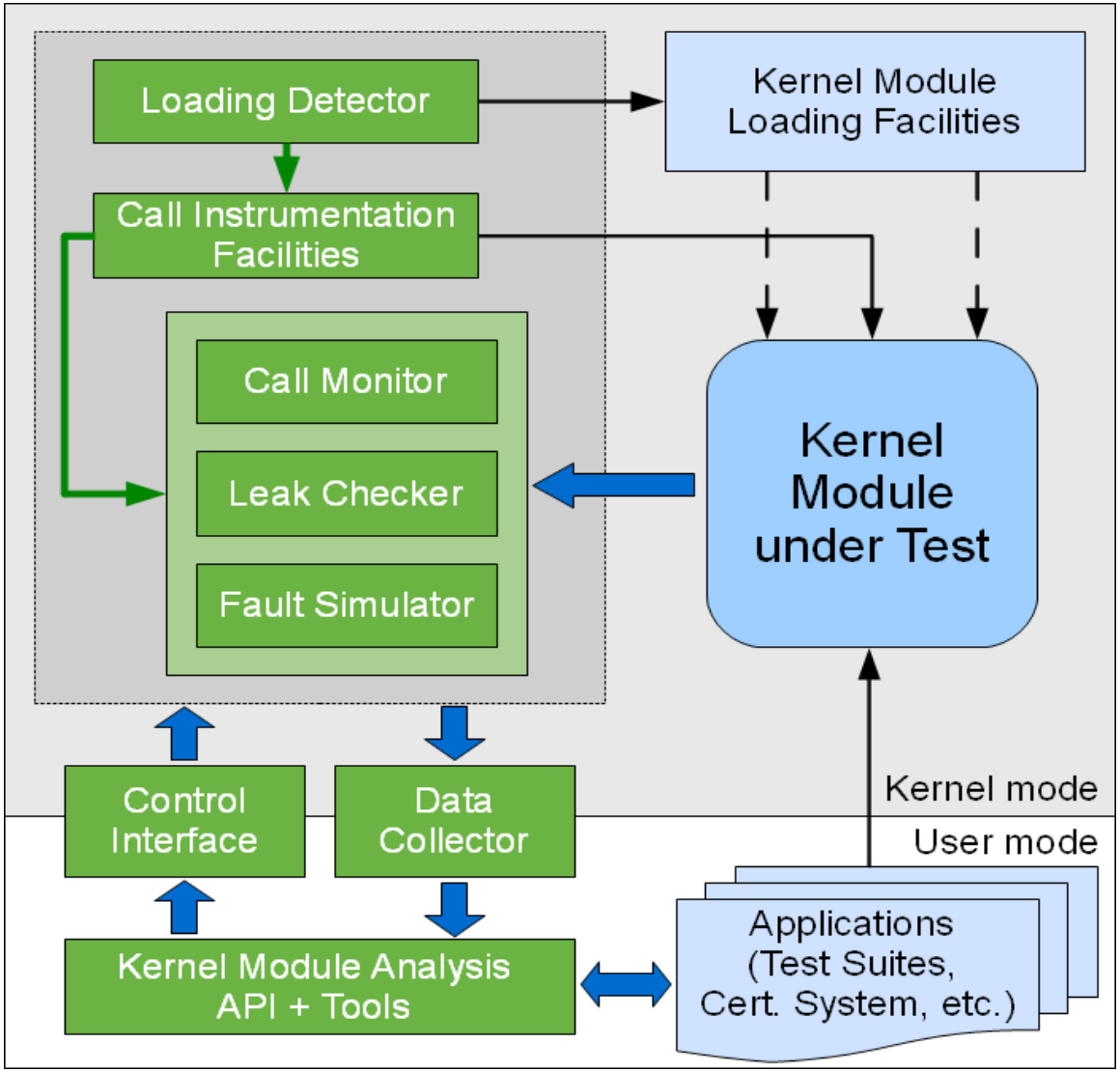
- Проверка утечек памяти, выходов за границы выделенной памяти, симуляция ошибок выделения памяти
- Влияет только на исследуемый драйвер (реализованный в виде модуля ядра)
- Сценарии симуляции ошибок явно заданы в коде
- Инструментирование неотделимо от системы тестирования

Платформа KEDR

Реализует подмену функций, вызываемых из кода модуля ядра, загруженного в память.

Инструменты используют эту подмену для выполнения своего кода вместо кода вызываемых функций.

KEDR: схема работы



Платформа KEDR: преимущества

- Воздействует только на целевой модуль, не трогая остальную систему
- Выполнена на основе модуля ядра, для его сборки и работы не требуется никаких специальных опций ядра
- Инструменты KEDR можно использовать одновременно

Платформа KEDR: недостатки

- Не работает с драйверами, встроенными в ядро
- Для каждой поддерживаемой архитектуры требуется разбор кода. На данный момент поддерживаются только x86 и x86_64
- Перехватываются только непосредственные вызовы функций, косвенные вызовы не отслеживаются

KEDR Leak Check

Отслеживает выделения/освобождения памяти через перехват вызовов соответствующих функций.

Если после выгрузки наблюдаемого модуля остались выделенные, но не освобожденные участки памяти, это считается утечкой.

KEDR Leak Check

- Собирает информацию только об исследуемом модуле
- Нетребователен к ресурсам системы
- Не учитывает, что в некоторых случаях ресурсы могут выделяться / освобождаться ядром.

KEDR Fault Simultaion

Позволяет симулировать ошибки в функциях, вызываемых наблюдаемым модулем ядра.

KEDR Fault Simulation

- Воздействует только на наблюдаемый модуль
- Гибкая настройка сценария симуляции:
  - Выражения, зависящие от параметров
  - Из пользовательского пространства
  - Можно менять во время работы

Выявление состояний гонки в драйвере Linux

Два неупорядоченных доступа к ячейке памяти, один из доступов - запись

Выявление состояний гонки в драйверах: инструменты

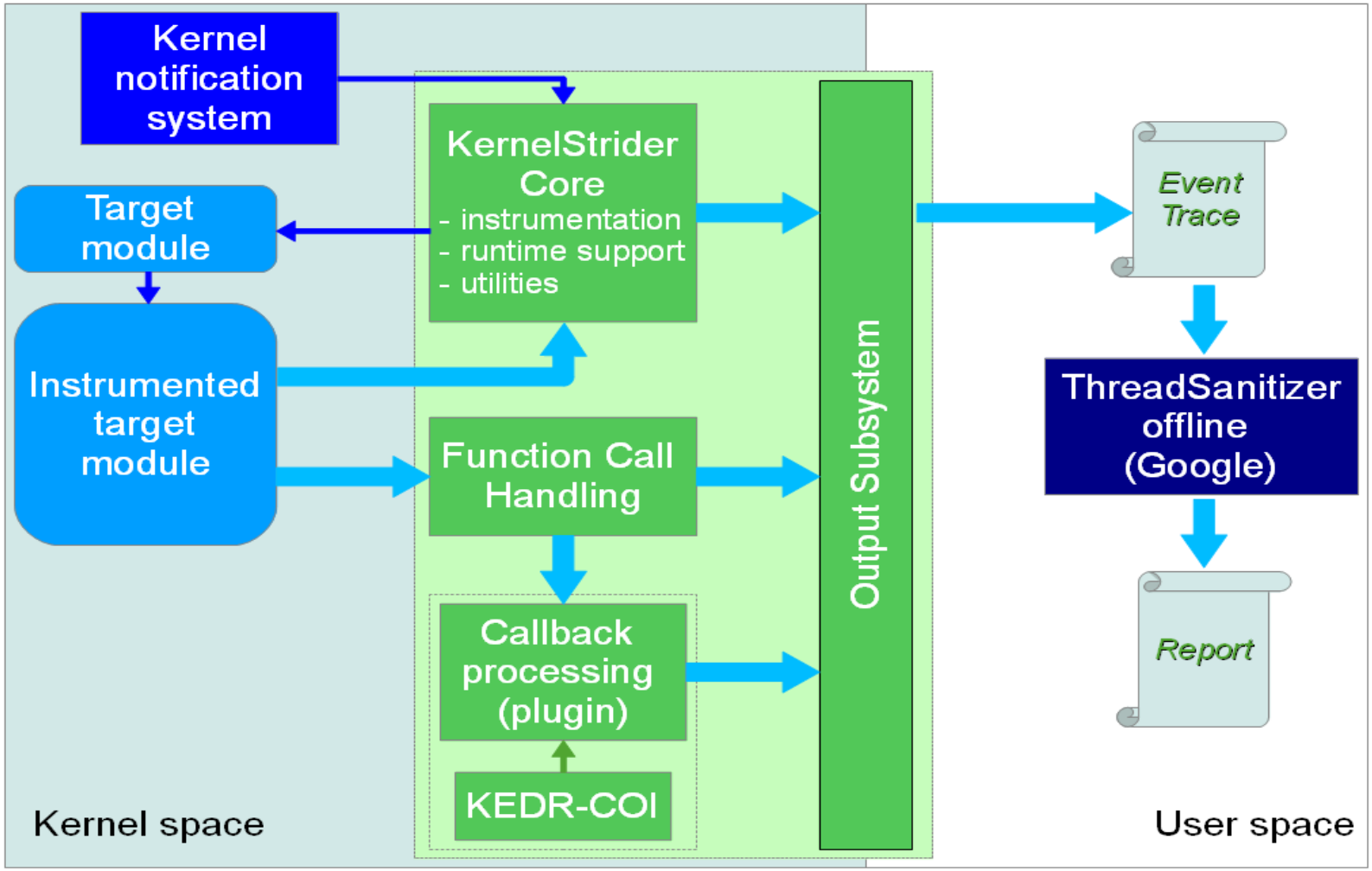
- ОС Windows
  - Data Collider(Microsoft Research)
  - Специальные планировщики (Microsoft Research)
  - DDT (Vitaly Chipounov, Volodymyr Kuznetsov, George Candea - École polytechnique fédérale de Lausanne)
- ОС Linux
  - Red Flag(Abhinav Duggal, Erez Zadok, Scott D. Stoller и др., Stony Brook University, NY, USA)
  - RaceHound(ISP RAS)

KernelStrider

Бинарное инструментирование кода модуля ядра в памяти. С помощью такого инструментирования собирается информация о доступах к памяти и элементах синхронизации.

Собранная информация передается инструменту ThreadSanitizer, который ищет состояния гонки.

KernelStrider: схема работы



KernelStrider: сделано

- Средства для сбора информации о работе модулей
- Средства для вывода собранной информации из пространства ядра
- Средства взаимодействия с Thread Sanitizer
- Обрабатываются некоторые часто используемые функции-примитивы синхронизации
- Поддержка аннотаций
- Поддержка драйверов в виде нескольких модулей

KernelStrider: в процессе/планы

- Модель операций для драйверов файловых систем и сетевых драйверов
- Поддержка других элементов синхронизации(tasklets, timers, waitqueues...)
- Включение в состав KEDR

LinuxTesting.org

Спасибо!

Цыварев Андрей,
tsyvarev@ispras.ru

ISP RAS
Institute for System Programming of the Russian Academy of Sciences