Статико-динамическая верификация драйверов файловых систем ядра Linux

Ефремов Денис

Автоматизированные способы тестирования

- Random testing
  - Запуск приложения на случайных входных данных
  - Использование шаблонов для порождения новых входных данных
  - Большое количество входных порождаемых данных
  - Совсем немногие из них действительно позволяют увеличить покрытие тестирования

Автоматизированные способы тестирования

- Symbolic testing
  - Символическое значение для всех входных данных (A,B,…) вместо конкретных значений (9, «test», …)
  - Каждое присваивание обновляет значение переменных программы символическим выражением (x = A - 2)
  - Условное выражение (if x> 0 then… else …) разделяет поток управления на две новых ветви
    - A-2>0
    - A-2<=0
  - Используется solver
  - Множества входных данных группируются по классам эквивалентности и тестируются лишь один раз
  - Подход плохо масштабируем

Автоматизированные способы тестирования

- Concolic = Symbolic + Concrete
  - Программа функционирует в конкретном режиме и в символическом
  - Символическое выполнение используется с целью сбора ограничений на пути для последующего порождения новых конкретных входных данных
  - В случае сложных ограничений, конкретные значения могут использоваться вместо символических, что позволяет достичь большей масштабируемости подхода

Concolic testing tools

Tool	Language	Platform	Constraint Solver
DART	C	NA	lp_solver
SMART	C	Linux	lp_solver
CUTE	C	Linux	lp_solver
CREST	**C**	**Linux**	**Yices**
EXE	C	Linux	STP
KLEE	**C (LLVM bitcode)**	**Linux**	**STP**
Rwset	C	Linux	STP
PathCrawler	C	NA	NA
SAGE	Machine code	Windows	Disolver

Limitation of concolic testing tools

Tool	float/double	pointer	native call	non-linear arithmetic op.	bitwise op.
DART	Y	Y	Y	NA	NA
SMART	Y	Y	Y	NA	NA
CUTE	Y	N	Y	NA	NA
CREST	**Y**	**Y**	**Y**	**P**	**P**
EXE	Y	N	Y	N	N
KLEE	**Y**	**N**	**P**	**N**	**N**
Rwset	Y	N	Y	N	N
PathCrawler	NA	NA	Y	NA	NA
SAGE	NA	Y	N	NA	NA

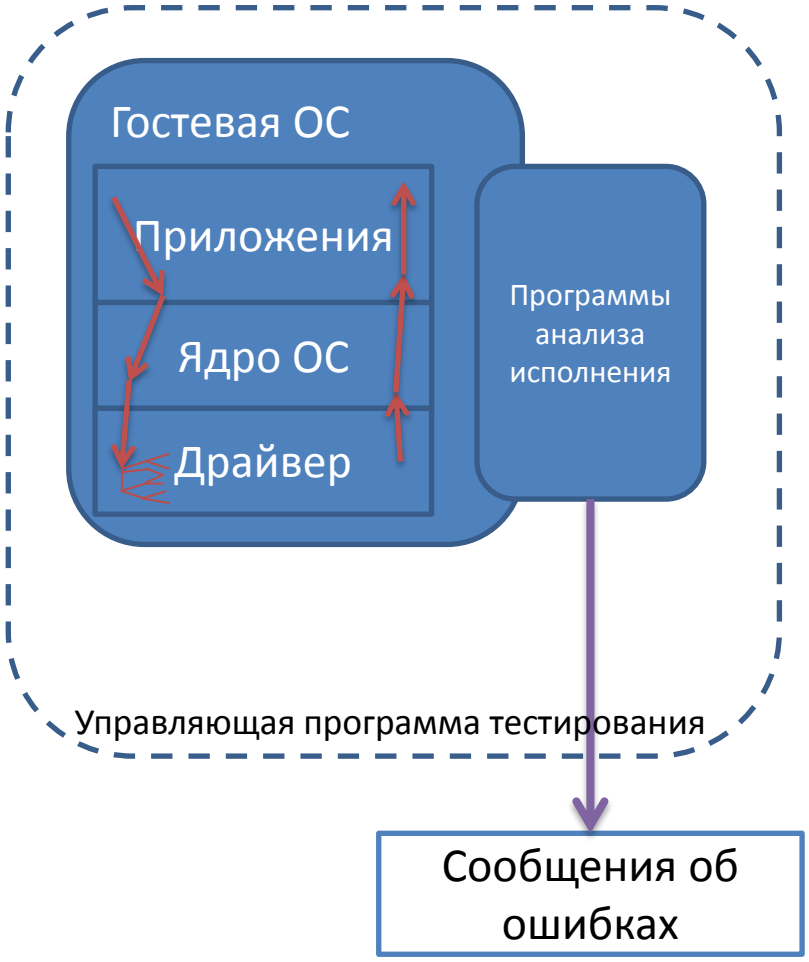
Эффективность

- CREST (покрытие по ветвям)
  - replace 31.1%
  - grep 37.3%
  - vim 19%
- KLEE (покрытие по строкам)
  - 94% coreutils
- CUTE (покрытие по ветвям)
  - sglib
    - red black trees alg. 71.18%
  - vim 37.86% branch coverage
    - hybrid concolic testing

S2E

- Основан на KLEE
- Использует модифицированный QEMU
  - Не зависит от наличия исходного кода
- Позволяет проводить общесистемный анализ
- Поддерживает систему плагинов

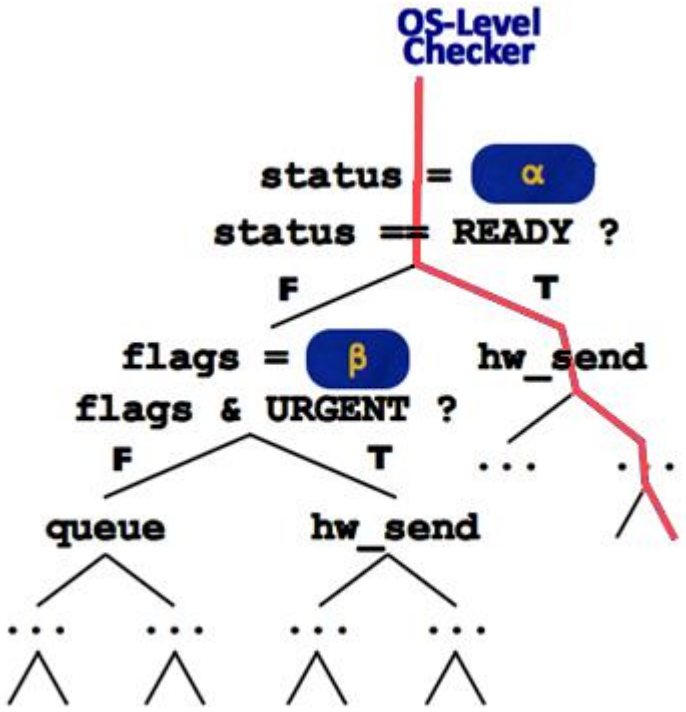
Схема работы S2E в применении к анализу драйверов



- Управляющая программа
- Программы анализа
- Отчёты об ошибках

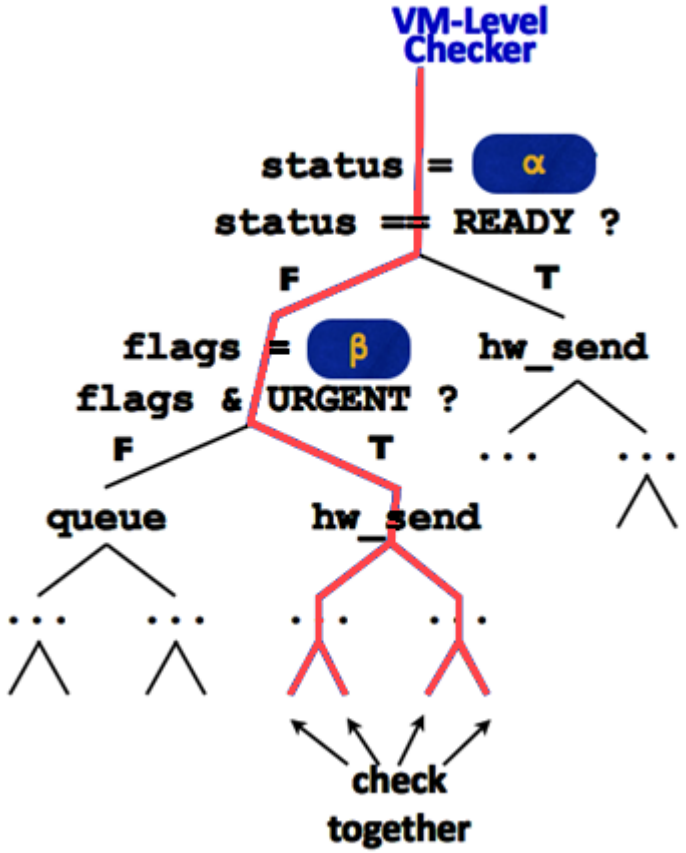
Проверки уровня ОС

- Работает в гостевой ОС
- Переиспользует существующие однопутевые динамические средства анализа
  - На множестве путей
- Пример анализа: захват и освобождение блокировок (lockdep), проверка на sleep в atomic контексте



Проверки уровня ВМ

- Анализ множества путей
- Пример анализа: проверка допустимости обращения драйвера по конкретному адресу, поиск бесконечных циклов

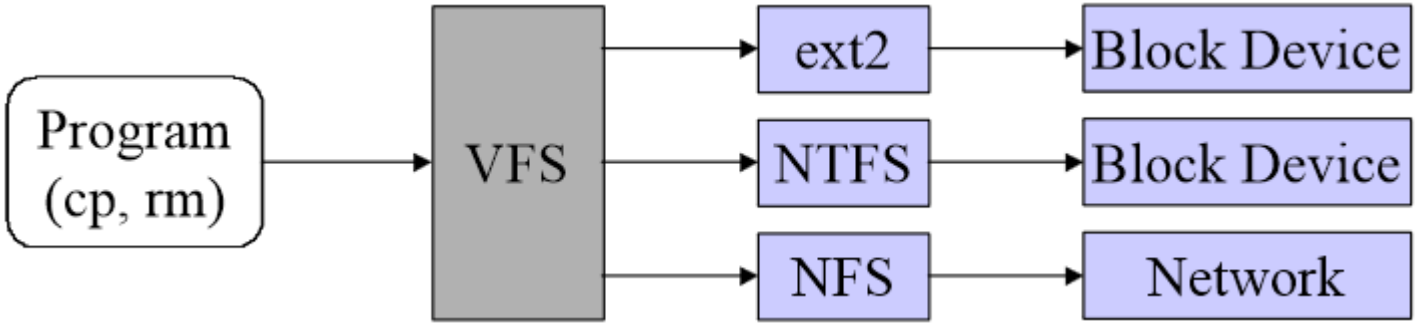


Подход к тестированию драйверов файловых систем

- Использование базового набора тестов для файловой системы
  - Spruce, LTP, PTS, …
- Вклинивание между интерфейсами ядра и драйвера
  - Пометка символическими входных данных для драйвера
- Увеличение эффективности покрытия
  - Отследивание выхода за пределы иследуемого драйвера
  - Приоритезация ветвей выполнения
  - Сокращение количества ограничений на пути

The Linux Virtual File System

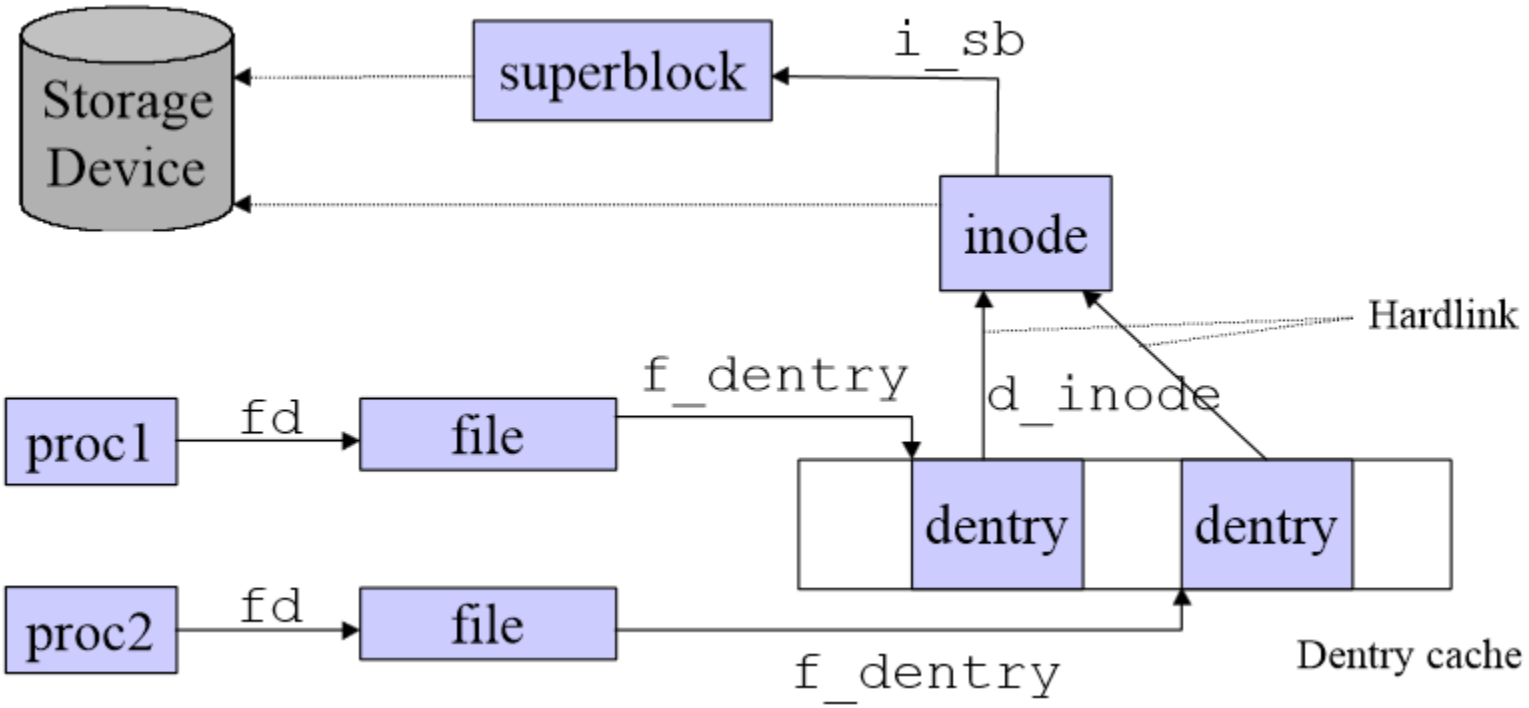
- Virtual Filesystem (VFS)
  - Предоставляет уровень абстракции между приложением и реализациями файловых систем
  - Обеспечивает поддержку множества типов файловых систем
    - Дисковые, сетевые, специальные



Общая файловая модель

- VFS вводит общую файловую модель для всех поддерживаемых файловых систем
- Общая файловая модель ориентирована на файловые системы UNIX. Все остальные файловые системы должны отображать свои концепции на общую файловую модель.
- Ключевые компоненты общей файловой модели:
  - superblock (информация о смонтированной файловой системе)
  - inode (информация о конкретном файле)
  - file (информация об открытом файле)
  - dentry (информация о записях каталога)

Взаимодействие объектов общей файловой модели



Драйвер фаловой системы

- Каждый объект общей файловой модели имеет набор определенных операций, которые могут быть выполнены на нём
- VFS предоставляет обобщённые реализации для некоторых операций
- таблица указателей на функции используется для каждого объекта, чтобы обеспечить объекту собственную версию операций
- Драйвер файловой системы состоит из реализаций этих методов

Пометка данных символическими

```
static inline void s2e_make_symbolic(void *buf, int
    size, const char *name)
{
  __asm__ __volatile__(
    ".byte 0x0f, 0x3f\n"
    ".byte 0x00, 0x03, 0x00, 0x00\n"
    ".byte 0x00, 0x00, 0x00, 0x00\n"
    : : "a" (buf), "b" (size), "c" (name)
  );
}
```

SystemTap

- Средство, которое позволяет собирать и анализировать информацию о работающей Linux-системе
- В отличие от встроенных средств, таких как netstat, ps, top, SystemTap был разработан с целью предоставить больше возможностей для сбора и представления информации
- SystemTap представляет собой интерфейс командной строки и скриптовый язык

- В контексте доклада рассматривается как средство инъекции кода в любую часть ОС

SystemTap

- В данном случае SystemTap рассматривается как средство выполнения определённых действий, при наступлении отслеживаемых событий
  - syscall.read отслеживание системного вызова read
  - syscall.close.return отслеживание возврата из системного вызова close
  - module("floppy").function("*“) вызов действий при передаче управления в любую функцию модуля floppy
  - kernel.function("*@net/socket.c").return отслеживание возврата из любой функции файла net/socket.c
  - kernel.statement (“*@kernel/sched.c:2917”) вызов действий при достижении строки 2917 файла kernel/sched.c

Инструментация

```
probe
     vfs.read,
     vfs.write
{
   s2e_make_symbolic(
        &($file->f_path->dentry->d_inode->i_count),
         4,
        "i_count")
}
```

Вопросы?