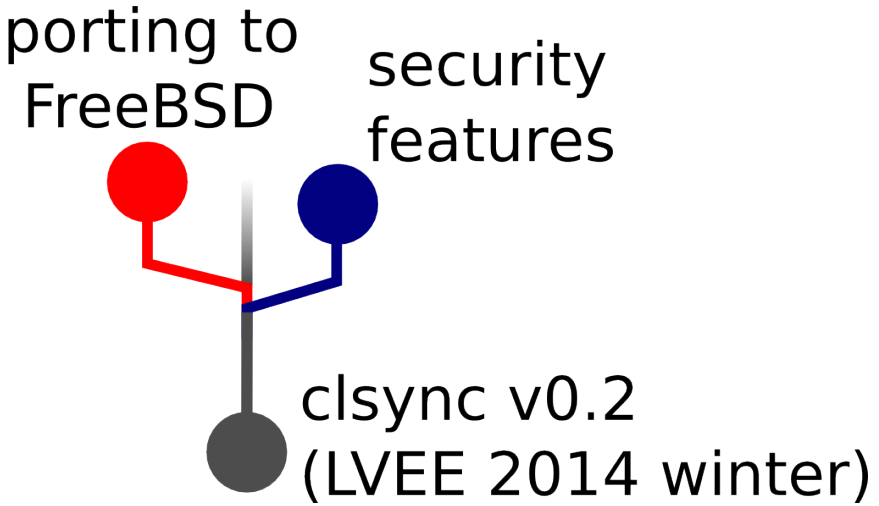
clsync progress: security and porting to freebsd



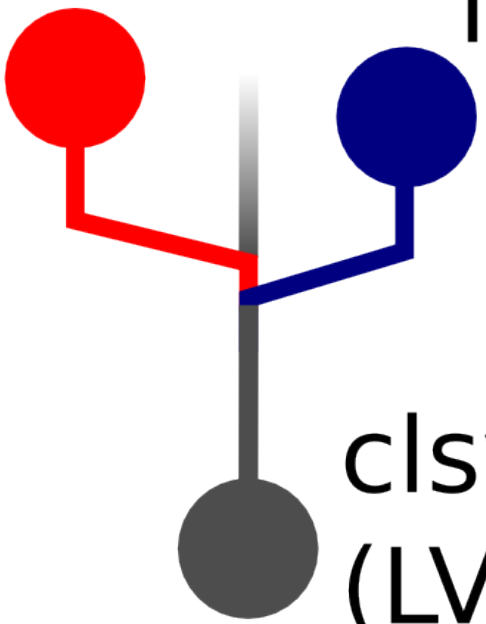
LVEE 2014

Linux Vacation / Eastern Europe

overview

porting to FreeBSD

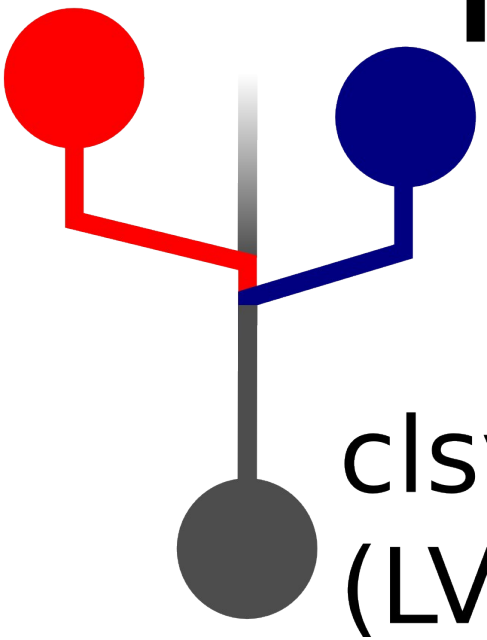
security features



clsync v0.2
(LVEE 2014 winter)

porting to FreeBSD

security features



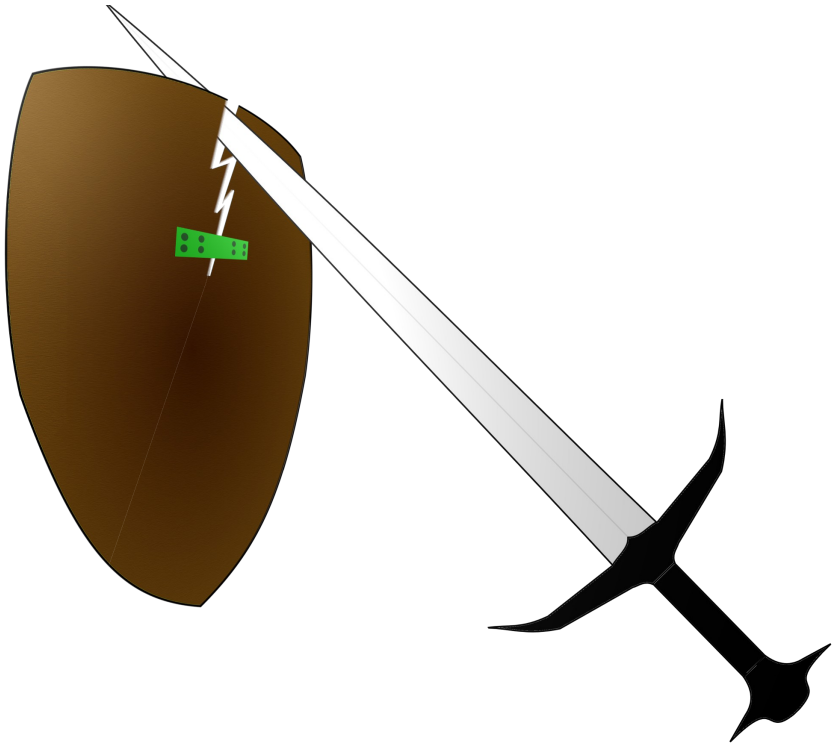
clsync v0.2
(LVEE 2014 winter)

LVEE 2014
Linux Vacation / Eastern Europe

security: aim

Цель:

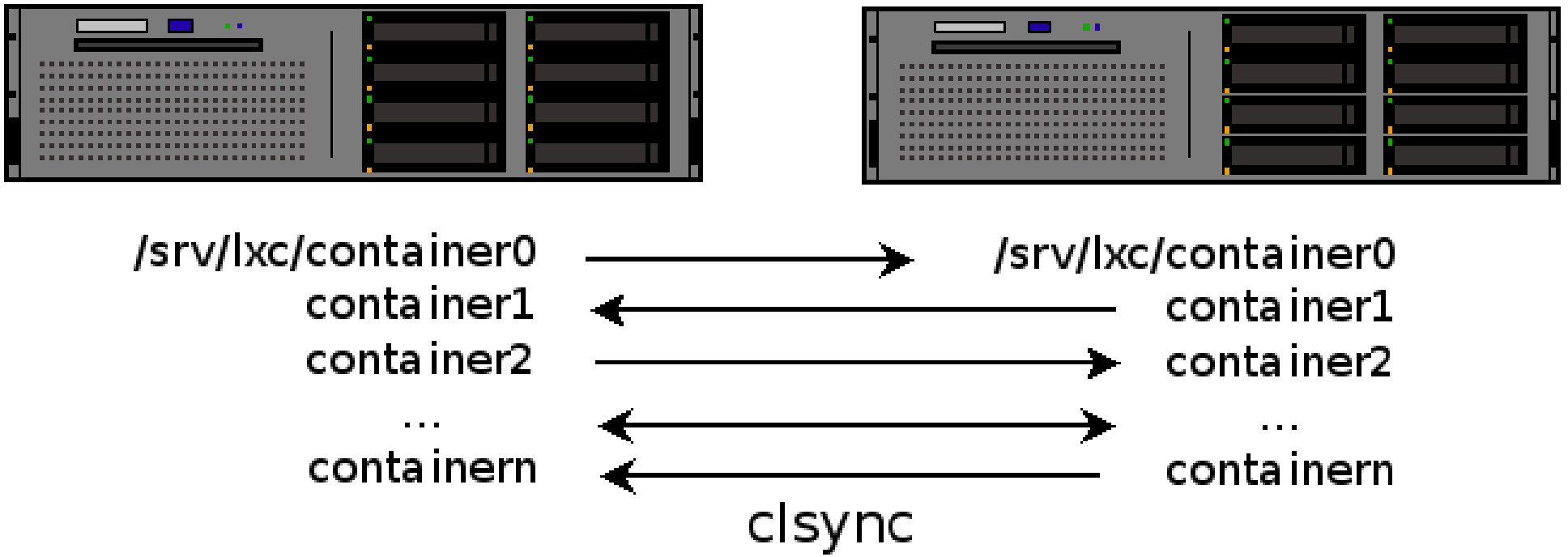
Минимизация последствий обнаружения и эксплуатации уязвимостей злоумышленником.



security: task

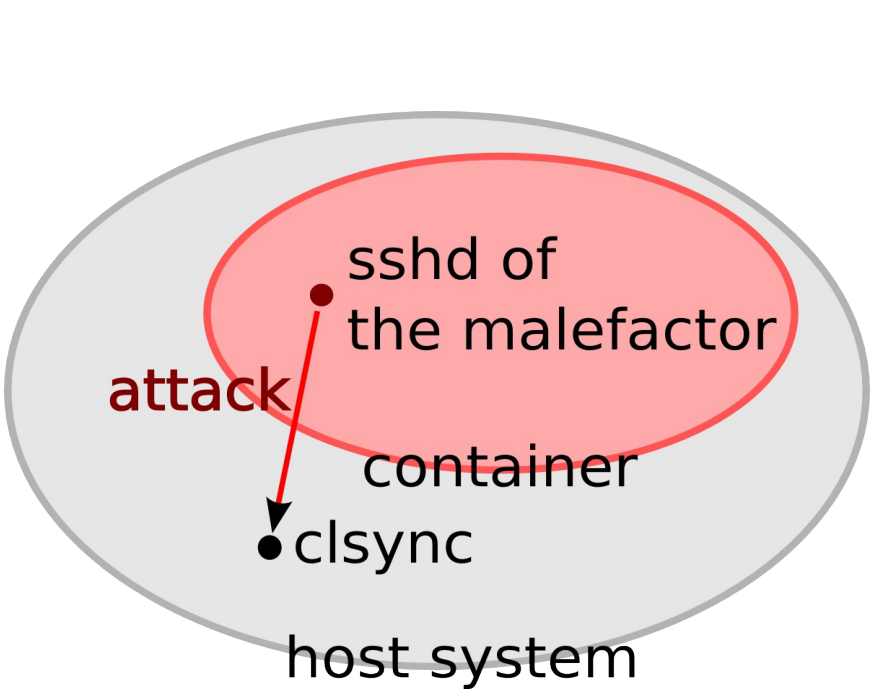
- **clsync наблюдает за файловым деревом с произвольными правами на объекты внутри.**
- **Злоумышленник имеет полный доступ к данному файловому дереву.**
- clsync не наблюдает за файлами, которые представляют интерес для злоумышленника
- clsync запускает внешний процесс для осуществления синхронизации

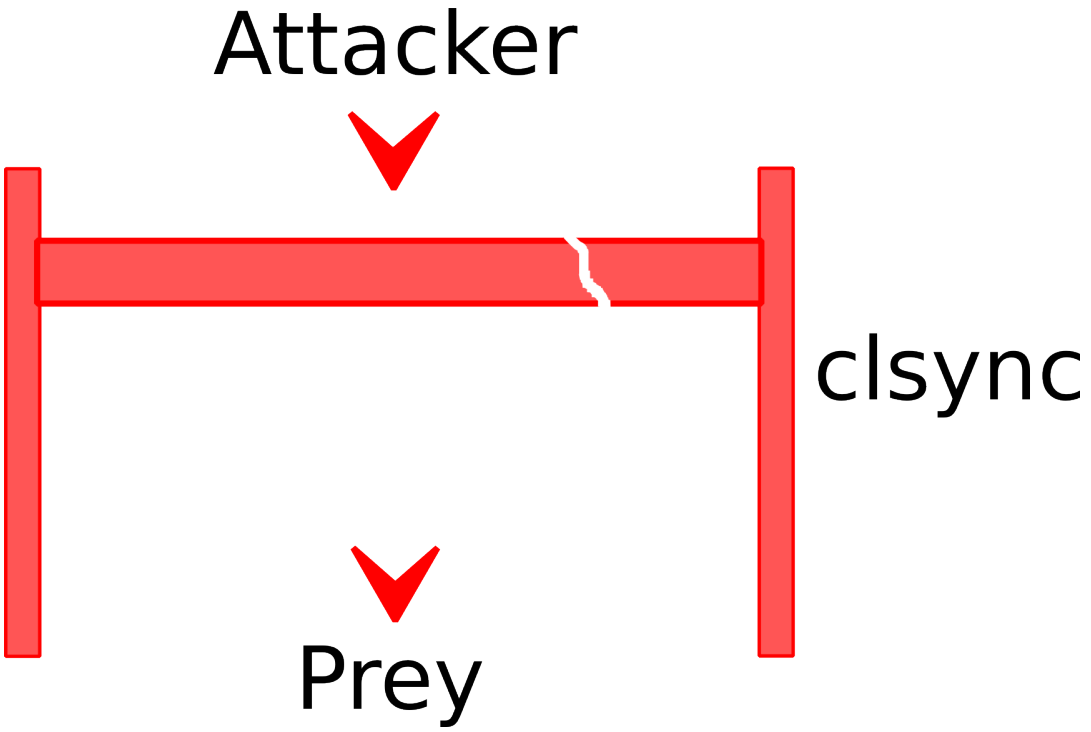
security: application example



- **Злоумышленник заперт внутри контейнера.**
- **clsync запускается с host-системы.**

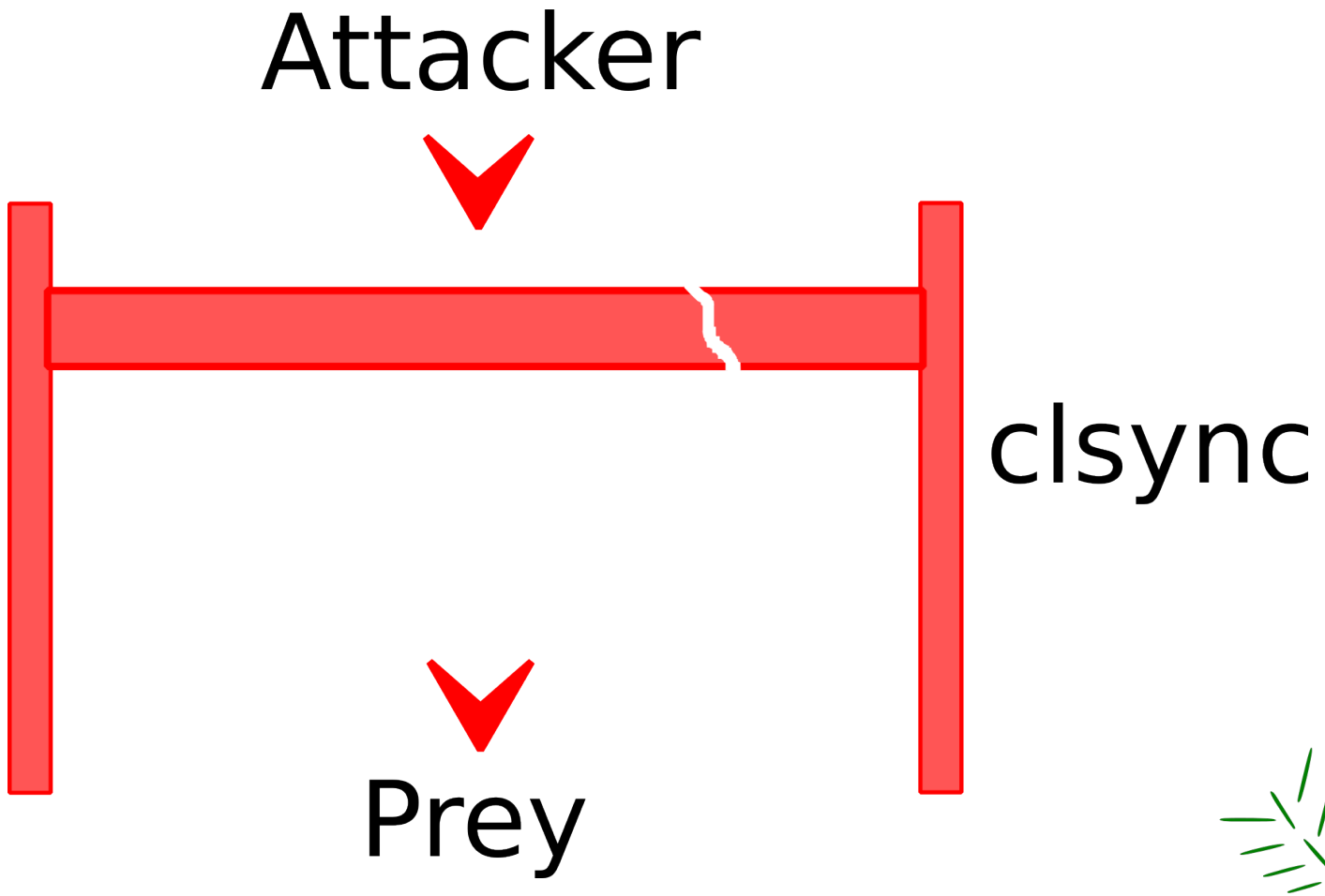
security: clsync v0.3



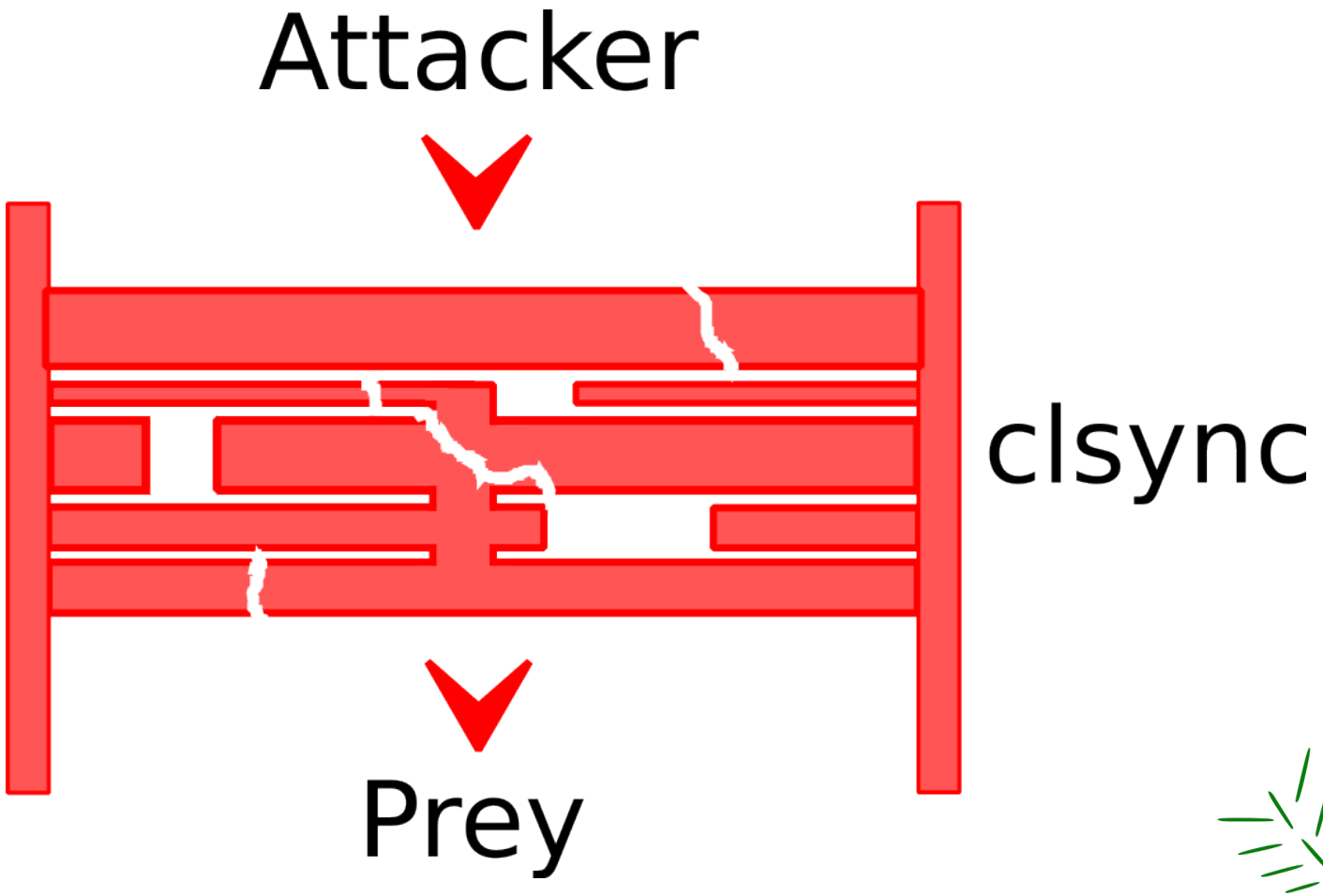


- **Злоумышленник заперт внутри контейнера**
- **clsync запускается с host-системы.**

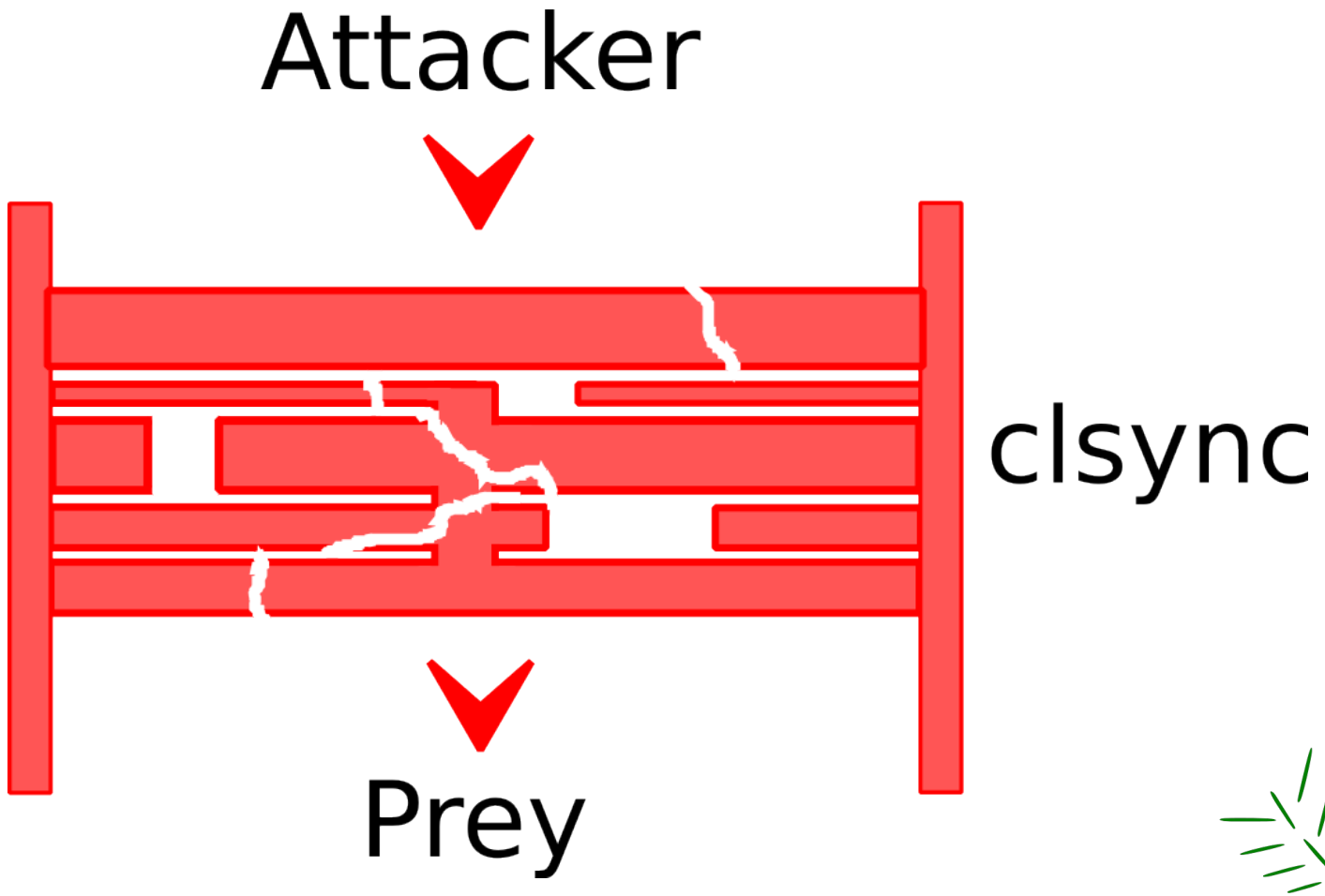
security: clsync v0.3



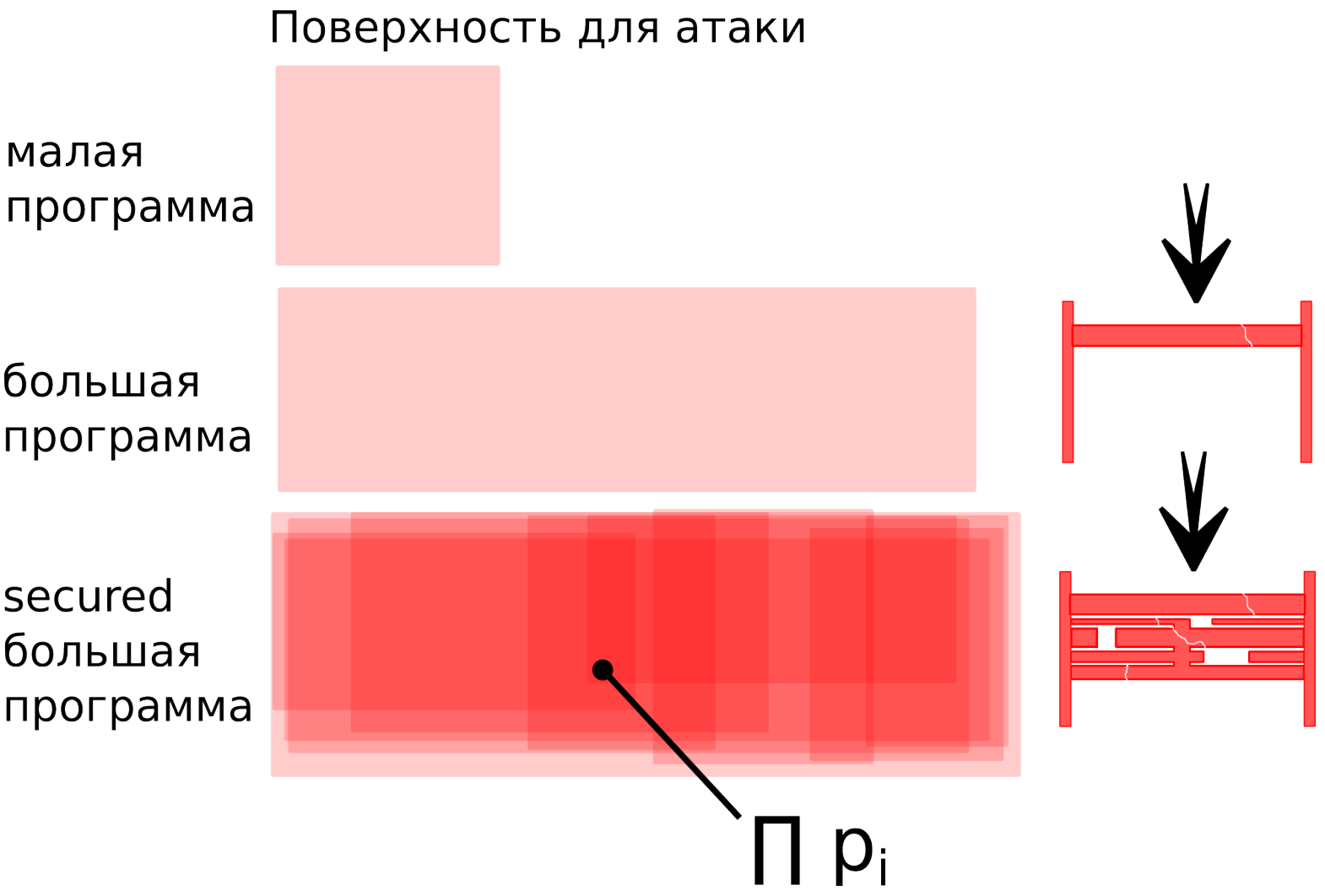
security: clsync v0.4



security: no guarantee



security: surface



security: overview

- Сброс привилегий
- Изоляция в собственные namespace-ы
- Thread splitting
- seccomp
- cgroups
- fork()

security: what is the first?

- **Сброс привилегий**
- Изоляция в собственные namespace-ы
- Thread splitting
- seccomp
- cgroups
- fork()

security: drop privileges

Тривиальная процедура с setuid()/setgid().

Проблемы:

- **clsync должен иметь возможность наблюдать за файловым деревом**, что требует либо привилегий root, либо capability “CAP_DAC_READ_SEARCH”.
- В определённых конфигурациях, **clsync запускает внешнее приложение** (например “rsync”) для осуществления синхронизации. У данного приложение должны быть достаточные права на чтение всего файлового дерева.

security: CAP_DAC_READ_SEARCH

Решение проблемы №1 (наблюдение):

- Сохранение capability “CAP_DAC_READ_SEARCH” для наблюдения.

Новая проблема:

- Данный capability даёт возможность непрямого доступа на чтение ко всем файлам раздела (через bruteforce handle-ов).

Решение новой проблемы:

- Разделять процесс на два thread-а [thread splitting].

Замечание: setuid()/setgid() действует на все thread-ы, а capabilities индивидуальны.

security: CAP_SETUID | CAP_SETGID

Решение проблемы №2 (“запуск rsync”):

- Аналогичное решение не работает из-за необходимости активации capability в дочернем процессе. Поэтому предлагается сохранение capabilities “CAP_SETUID” и “CAP_SETGID” и использование setuid()/setgid() перед запуском.

Новая проблема:

- Появляется возможность получить привилегии root-а, что полностью ликвидирует защиту

Решения новой проблемы:

- Разделять процесс на два thread-а [thread splitting] с разными привилениями (используя capabilities).
- Разделять процесс на два полноценных процесса [через fork()]. Более безопасный вариант.

Замечание: атака с thread-а на thread на порядок проще, чем на “fork()-нутый” процесс

security: what is next?

- Сброс привилегий
- → **Изоляция в собственные namespace-ы**
- Thread splitting
- seccomp
- cgroups
- fork()

security: unshare()-ing

unshare() flags (Linux 3.13):

- CLONE_FILES – **File descriptors**
- CLONE_FS – **FS attributes**
- CLONE_NEWIPC – **SysV IPC**
- CLONE_NEWNET – **Network**
- CLONE_NEWNS – **Mounts**
- CLONE_NEWUTS – **UTS**
- CLONE_SYSVSEM – **SysV semaphores**

security: unshare()-ing mountpoints

Наиболее автоматический вариант:

- chdir(newroot);
- mkdir(“old_root”);
- **unshare(CLONE_NEWNS);**
- mkdir(newroot_bind, 0700);
- mount(newroot, newroot_bind, ..., MS_BIND | … , NULL);
- chdir(newroot_bind);
- **pivot_root(“.”, “old_root”);**
- chroot(“.”);
- **umount2(“old_root”, MNT_DETACH);**

security: problem of unshare()-ing mountpoints

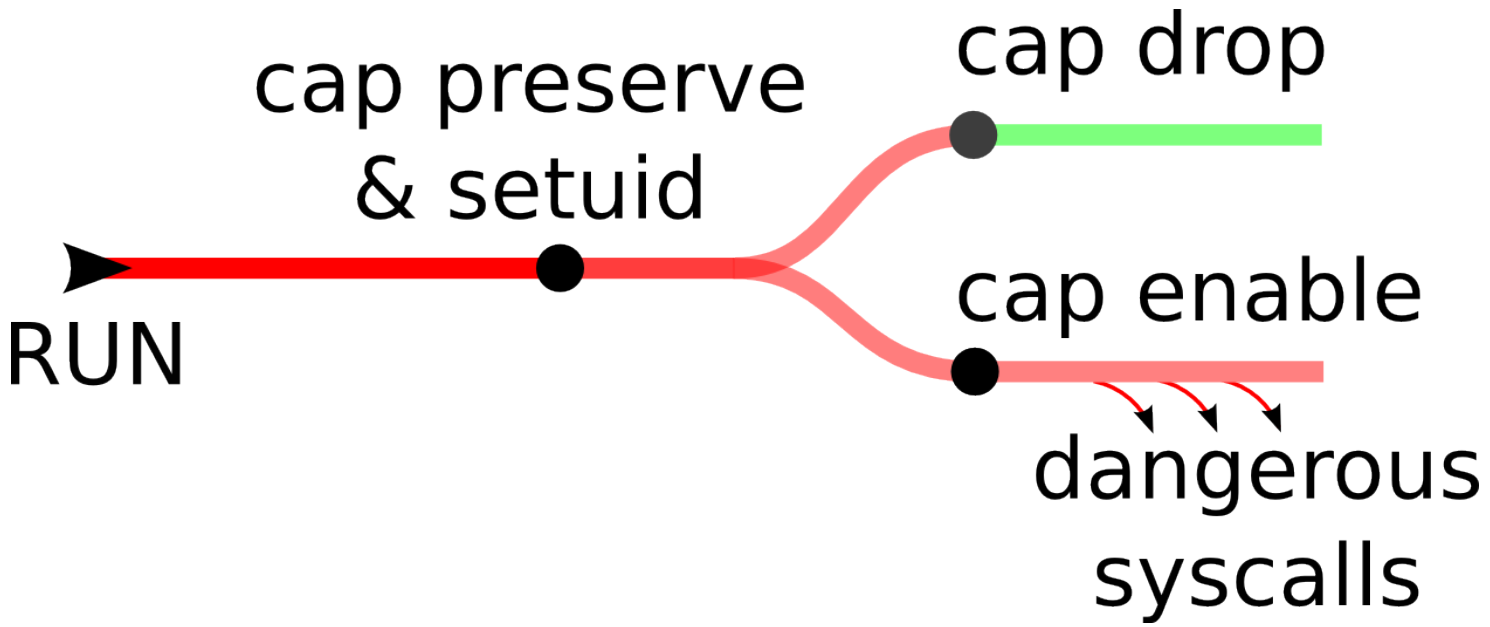
- Включенный capability “CAP_DAC_READ_SEARCH” создаёт уязвимость, за счёт разрешения bruteforce-a handle-ов ФС.
- Но “CAP_DAC_READ_SEARCH” необходим для возможности наблюдения за файловым деревом при сброшенных привилегиях.
- Решение данной проблемы – это thread splitting

security: what is next?

- Сброс привилегий
- Изоляция в собственные namespace-ы
- → **Thread splitting**
- seccomp
- cgroups
- fork()

security: what is thread splitting?

Под термином “thread splitting” понимается создание дополнительного thread для обработки syscall-ов, требующих особых привилегий.



security: privileged syscalls list

Список функций, требующий повышенных привилегий:

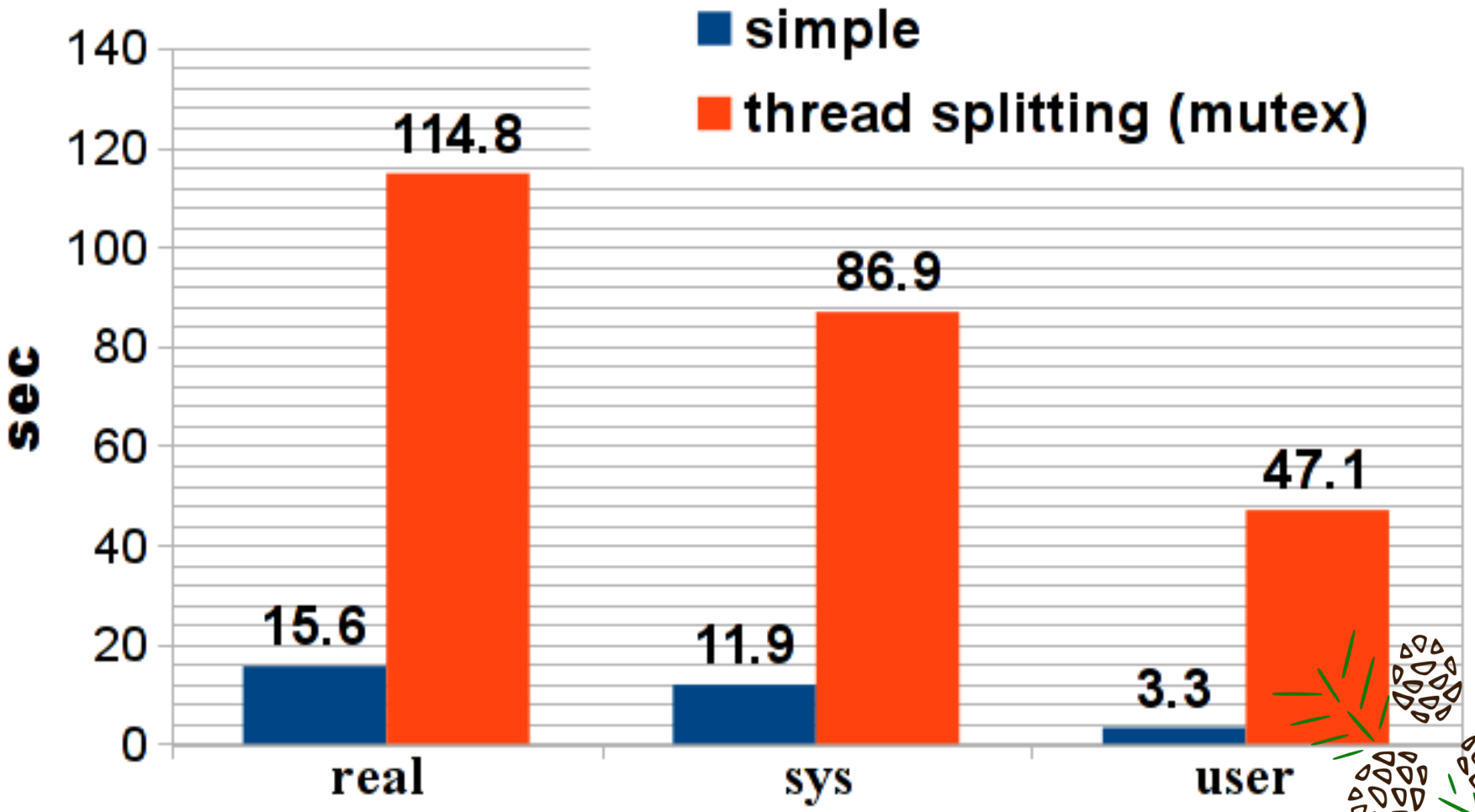
- fts_open()
- fts_read()
- fts_close()
- inotify_add_watch()
- setuid()/setgid()

security: problems of the thread splitting

Проблемы данного подхода:

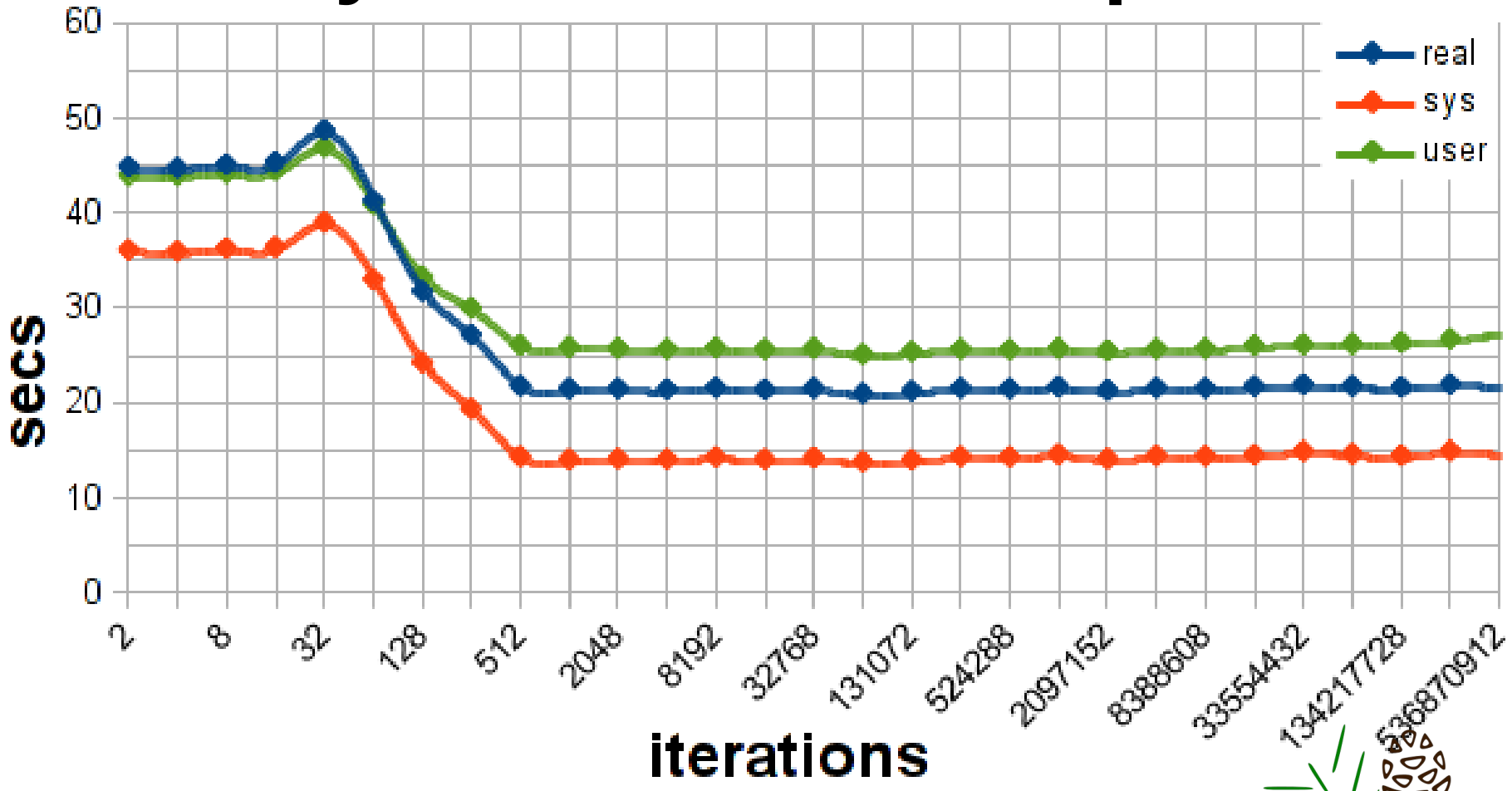
- Требуется использовать блокировки для каждого привилегированного syscall, что снижает производительность в несколько раз.
- Требуется защитить привилегированный thread от непривилегированного. Для этого используется seccomp, о чём будет рассказано позже.

security: the performance problem of the thread splitting



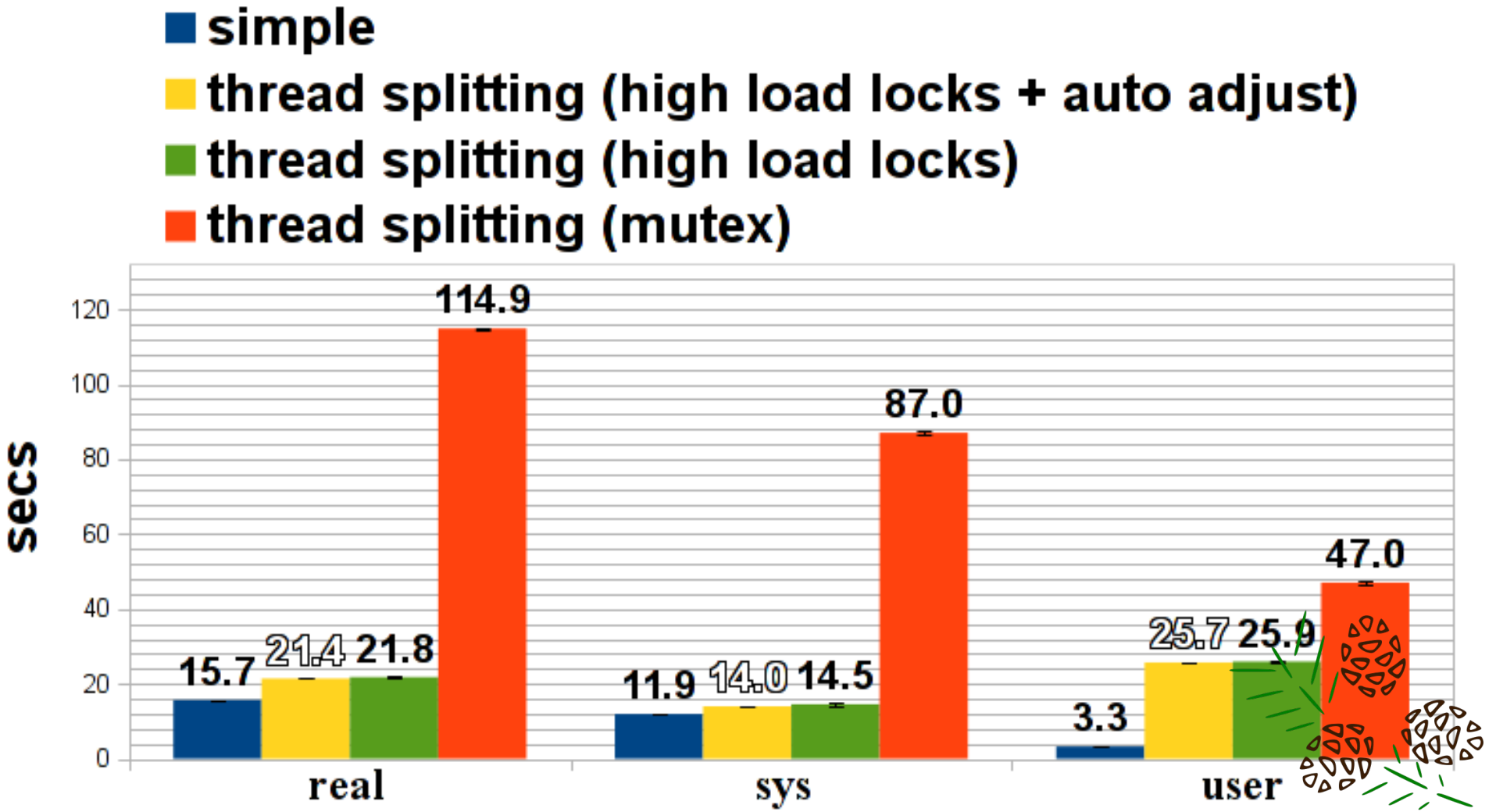
- Проблема вызвана использованием pthread_**mutex**_*(), которые расчитаны на более медленную блокировку-разблокировку.
- Для быстрых блокировок-разблокировок бывает **spinlock**, однако он не подходит для блокировок и разблокировок с большими интервалами (что является штатной ситуацией для clsync).
- Было предложено сделать комбинированный механизм блокировок **timed spinlock with fallback on mutex**.

Зависимость времени выполнения initial sync от timeout-a spinlock-a



- После initial sync системные вызовы бывают достаточно редко, поэтому использование spinlock-а с высоким timeout приводит к бесполезным дополнительным затратам ресурсов CPU.
- Оптимальное значение timeout зависит от конкретного ядра, окружения и аппаратного обеспечения.
- Было предложено использовать 8192 итерации как нулевое приближение, а далее делать автоматическую калибровку.

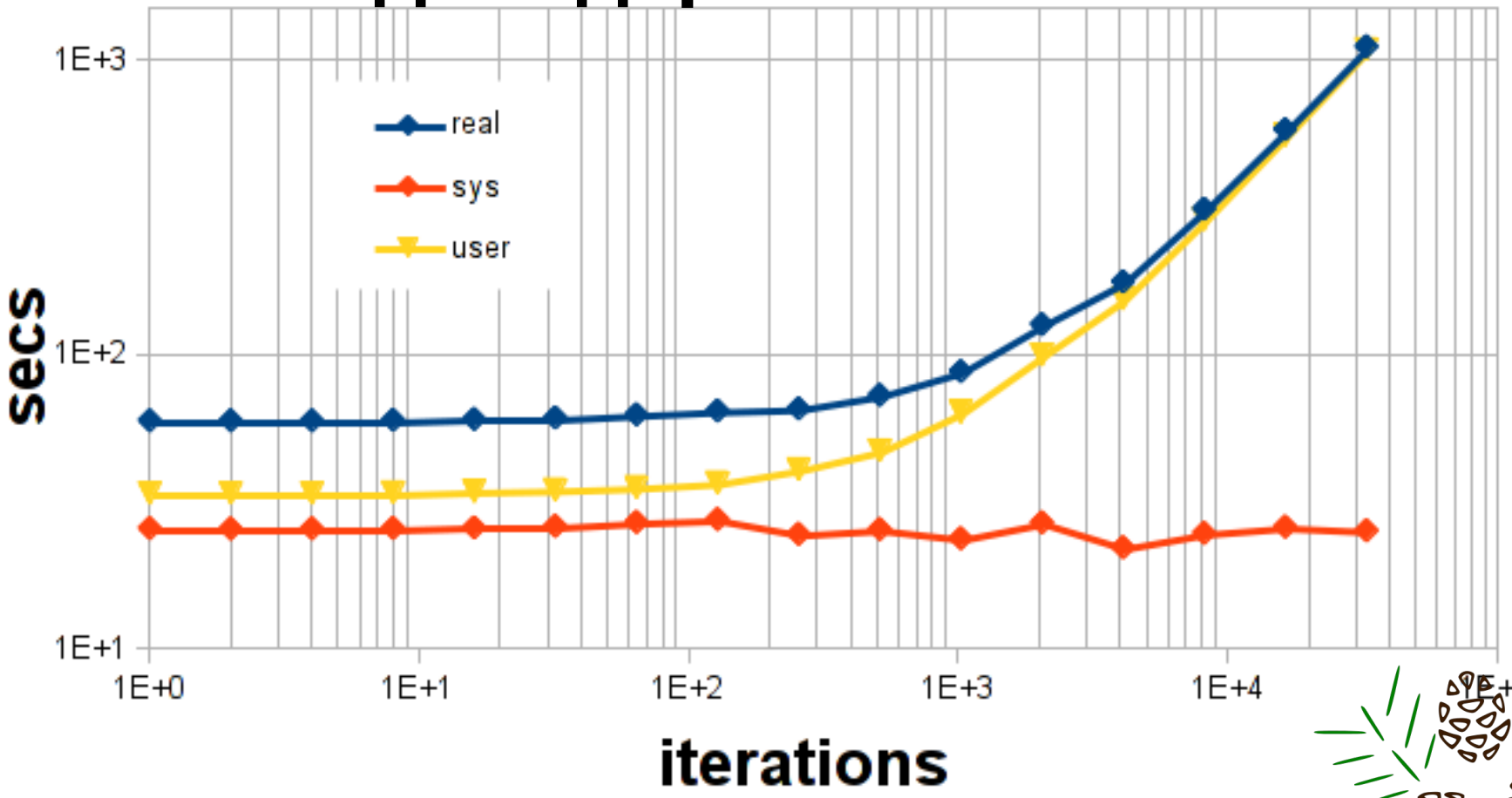
Зависимость времени выполнения initial sync от режима clsync



security: problem of the new locks

- Использование spinlock на одноядерных системах приводит к ещё бóльшей потере производительности.
- Калибровка может работать неточно в системах с существенно неравномерной нагрузкой.

Зависимость времени выполнения Initial sync от spinlock timeout на одноядерной системе



security: what is next?

- Сброс привилегий
- Изоляция в собственные namespace-ы
- Thread splitting
- ➔ **seccomp**
- cgroups
- fork()

security: seccomp

- Предлагается использовать seccomp filter для задания whitelist syscall-ов для непривилегированного thread.
- Это запретит использование mprotect, что даёт возможность обеспечить защиту от записи кода и данных привилегированного thread.

Список разрешённых syscall-ов:

- futex
- inotify_init1
- alarm
- stat
- fstat
- lstat
- open
- write
- close
- wait4
- unlink
- nanosleep
- tgkill
- clock_gettime
- rt_sigreturn
- brk
- mmap
- munmap
- wait4
- rmdir
- exit_group
- select
- read
- rt_sigprocmask
- rt_sigaction

security: problems of seccomp in clsync

Проблемы:

- Запрещено применение многопоточности синхронизаций [из-за запрета mprotect() не работает pthread_create()]

security: what is next?

- Сброс привилегий
- Изоляция в собственные namespace-ы
- Thread splitting
- seccomp
- → **cgroups**
- fork()

security: cgroups

На данный момент cgroups в clsync используется только для ограничения доступа к “устройствам” (в /dev).

На запись:

- /dev/console
- /dev/null

На чтение:

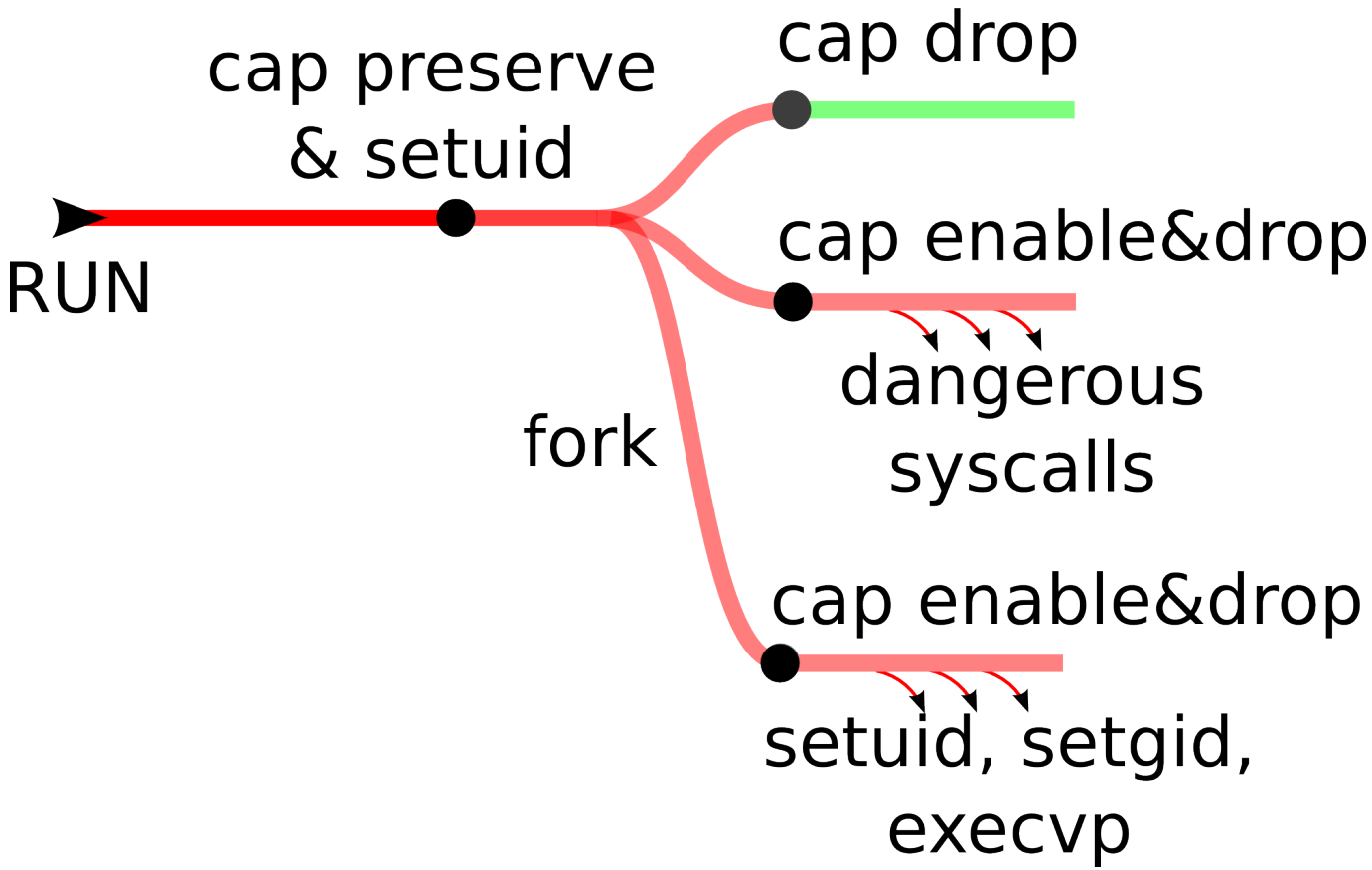
- /dev/console
- /dev/zero
- /dev/random
- /dev/urandom

security: what is next?

- Сброс привилегий
- Изоляция в собственные namespace-ы
- Thread splitting
- seccomp
- cgroups
- → **fork()**

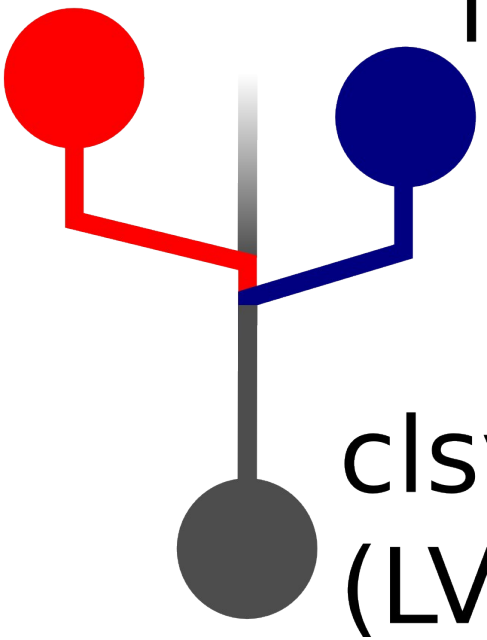
security: fork()

- fork() используется для запуска внешнего helper-а, выполняющего операции setuid()/setgid() и exec*(). Что позволяет сбросить capabilities CAP_SETUID и CAP_SETGID в атакуемом процессе.



porting to FreeBSD

security features



clsync v0.2
(LVEE 2014 winter)

porting to freebsd: problems

Задача по портированию включает в себя решение следующих проблем:

- Отсутствие поддержки inotify в ядре.
- Различные мелочи, вызванные использованием другого окружения: компилятор, shell, BSD make и т.п.
- Подготовка и публикация freebsd port.

porting to freebsd: inotify

В качестве альтернативы inotify во FreeBSD для наблюдения за ФС предлагается 3 backend-а:

- kqueue()/kevent()
- BSM API
- dtrace

porting to freebsd: kqueue()/kevent()

Проблемы kqueue()/kevent():

- Требование open() на каждый наблюдаемый объект.
- Недостаточность получаемой информации, необходимость пересканирования директорий и детального слежения за inode-ами.
- Большое количество сложноучитываемых проблем.

Но есть libinotify-kqueue...

porting to freebsd: BSM API

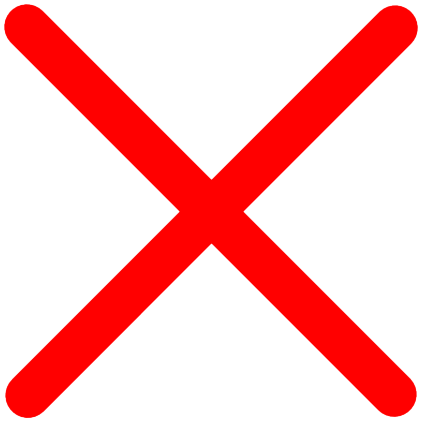
Проблемы BSD API:

- Необходимость глобальной переконфигурации auditd.
- Использование системы не по своему назначению.

porting to freebsd: dtrace

Проблема dtrace:

- dtrace во FreeBSD реализован в “урезанном” варианте (без поддержки некоторых built-in переменных), в результате становится невозможным получать полные пути объектов, соответствующих событиям.



conclusions

- Комбинирование различных Linux API позволяет создать многослойные препятствия для злоумышленника.
- Для FreeBSD не найдено удовлетворительного интерфейса для наблюдения за событиями ФС. Однако рекомендуется использовать libinotify-kqueue.

clsync progress: security and porting to freebsd

Q&A