# Как STACKLEAK улучшает безопасность ядра Linux

Александр Попов

Positive Technologies

# Кто я

- Александр Попов
- Разработчик ядра Linux с 2012 года
- Исследователь информационной безопасности в

# План доклада

- Миссия Kernel Self Protection Project
- Обзор STACKLEAK с упоминанием grsecurity/PaX
- Мои цель и тактика
- STACKLEAK как функция безопасности:
  - Какие уязвимости закрывает
  - Механизмы защиты
  - Влияние на производительность
- Текущее состояние работы:
  - Что сделано за год, что предстоит сделать
  - Опыт общения в сообществе разработчиков ядра Linux
  - Извлеченные уроки

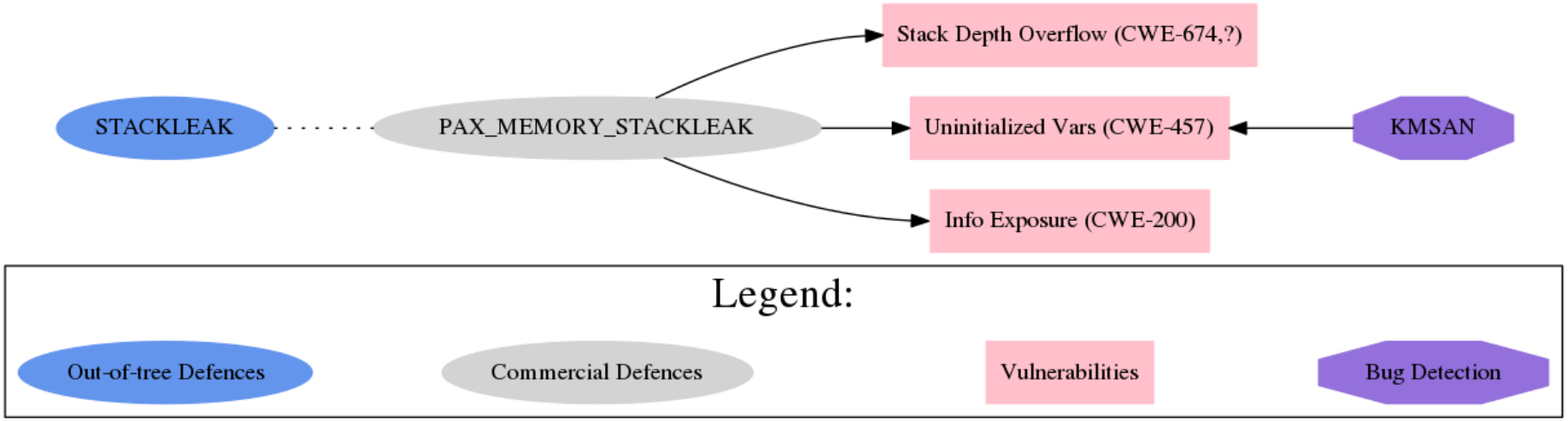
# Kernel Self Protection Project

- Безопасность - это больше, чем исправление ошибок
- Ядро Linux должно безопасно отрабатывать в ошибочной ситуации
- Цель: устранение классов уязвимостей и методов их эксплуатации
- Полезные ссылки:
  - KSPP wiki: http://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project
  - Обзор KSPP (Kees Cook): https://outflux.net/slides/2017/lss/kspp.pdf

# Linux Kernel Defence Map

https://github.com/a13xp0p0v/linux-kernel-defence-map

# Место STACKLEAK в общей картине



# Обзор STACKLEAK

- Замечательная функция безопасности ядра Linux
- Разработана PaX Team
- `PAX_MEMORY_STACKLEAK` в grsecurity/PaX патче
- Однако, grsecurity/PaX патч теперь закрыт для сообщества
- Последняя публичная версия для ядра 4.9 (04.2017)

# Моя цель

**Привнести STACKLEAK в ванильное ядро Linux**

# Моя тактика

- Выделить STACKLEAK из grsecurity/PaX патча

```
wc -l ../grsecurity-3.1-4.9.24-201704252333.patch
225976 ../grsecurity-3.1-4.9.24-201704252333.patch
```

- Тщательно изучить код и сформировать патч
- Отправить в LKML, получить обратную связь, улучшить, повторить
- /* Так повторяю уже год */

# STACKLEAK: свойства безопасности

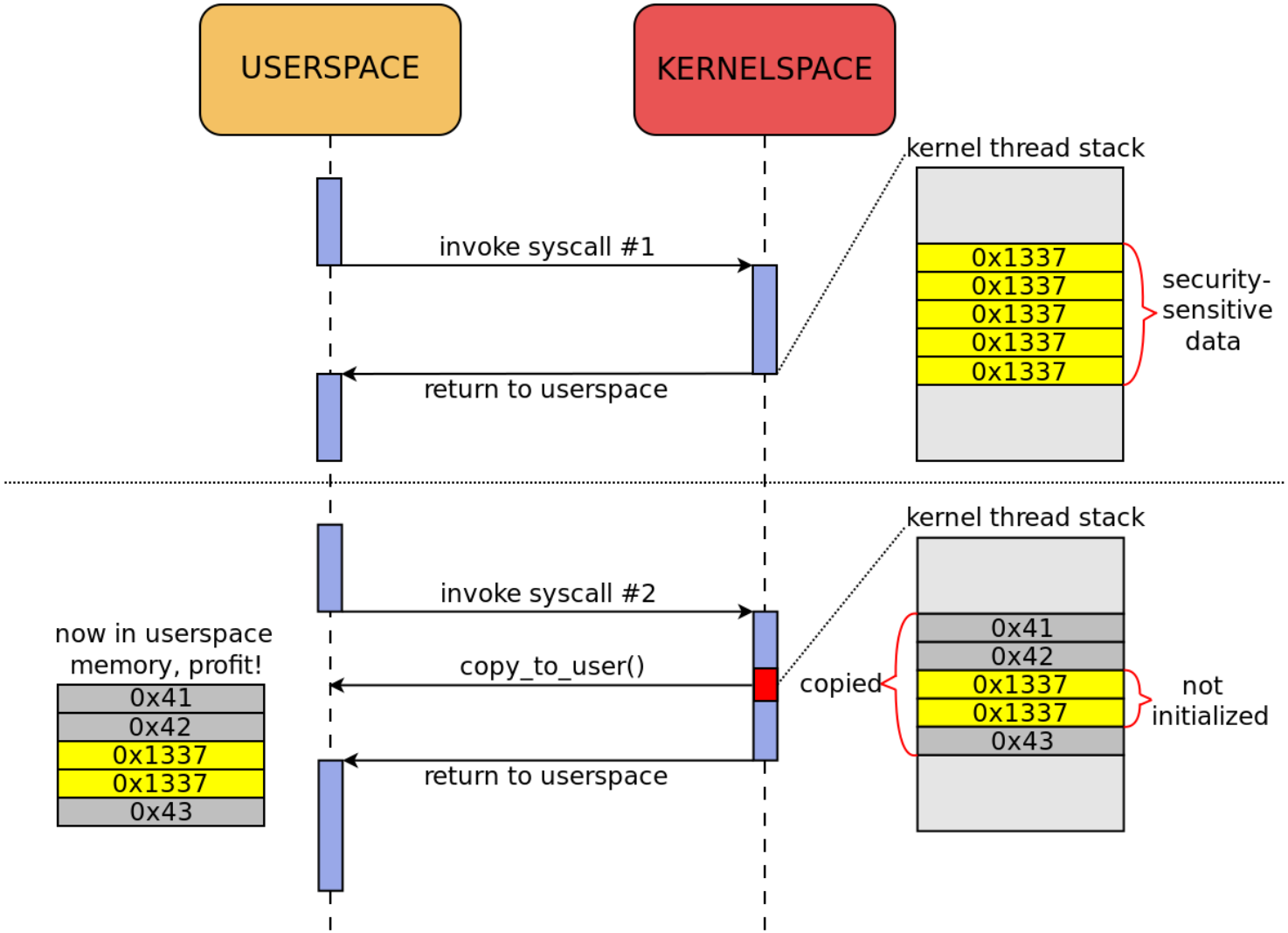
Кратко о свойствах безопасности,

предоставляемых STACKLEAK

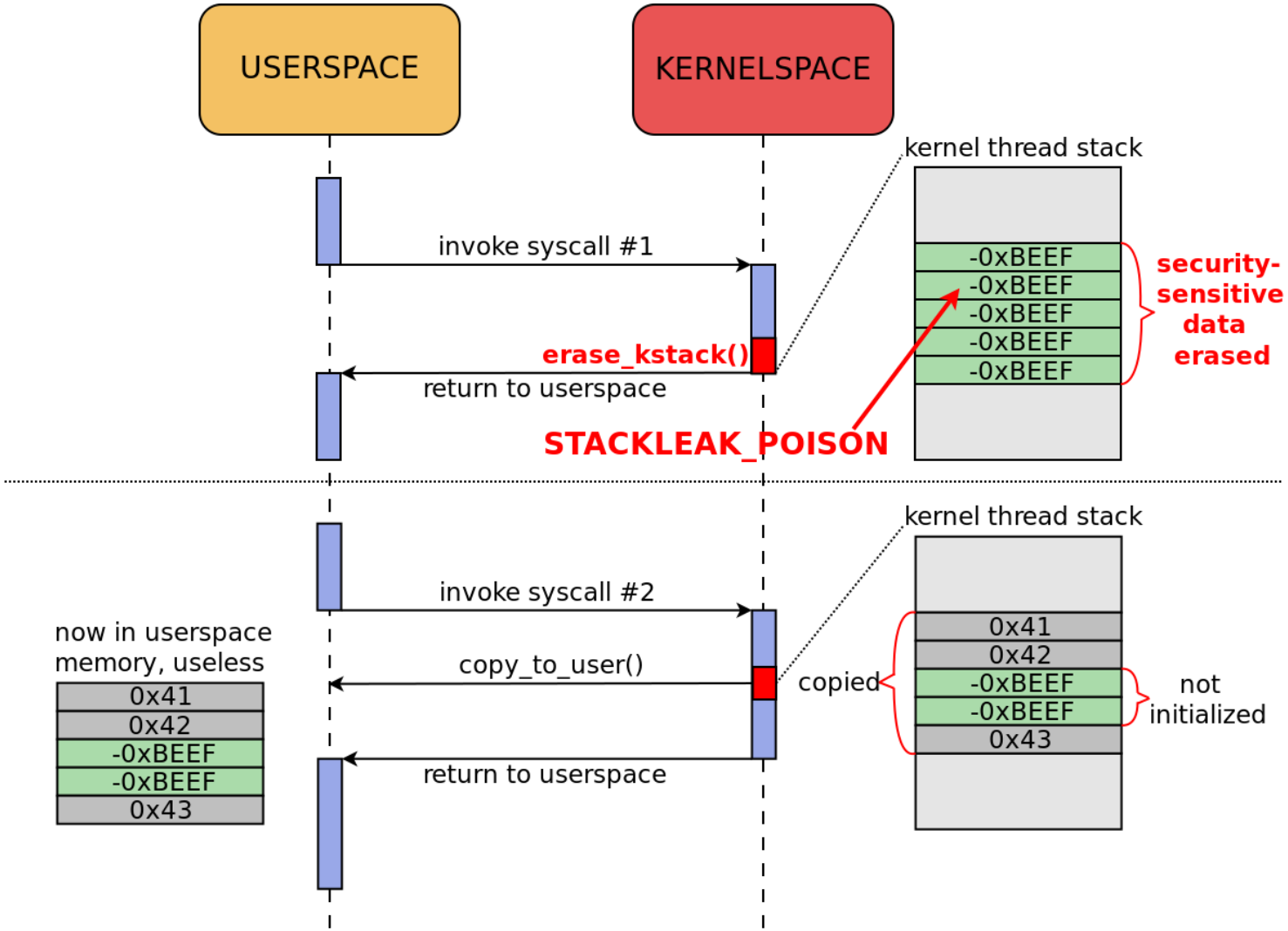
# STACKLEAK: свойство безопасности №1

- Очищает стек ядра в конце системного вызова
- Сокращает полезную информацию, которую могут выдать некоторые* утечки из ядерного стека в пользовательское пространство

# Пример утечки информации из стека ядра



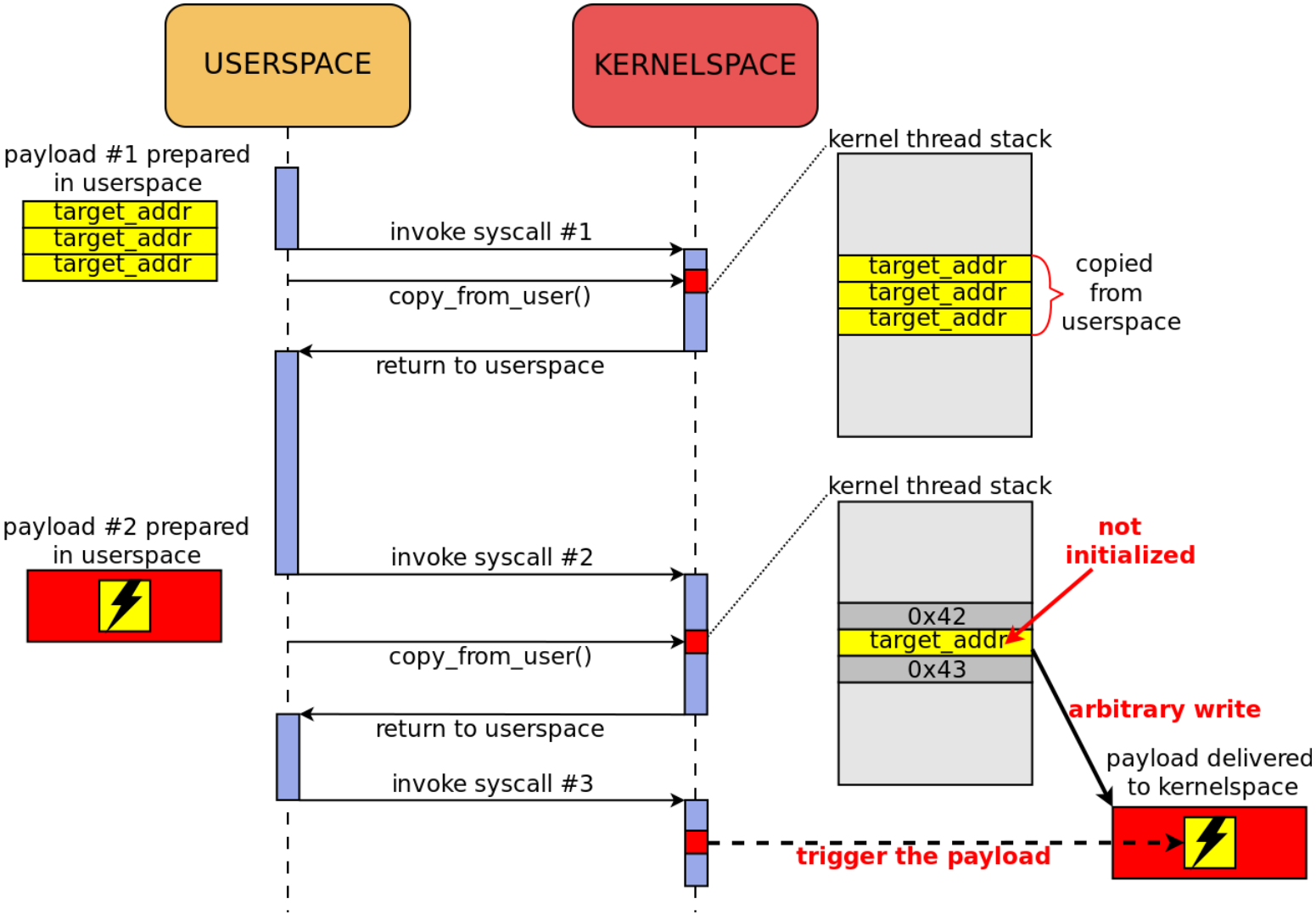
# Блокирование таких утечек информации



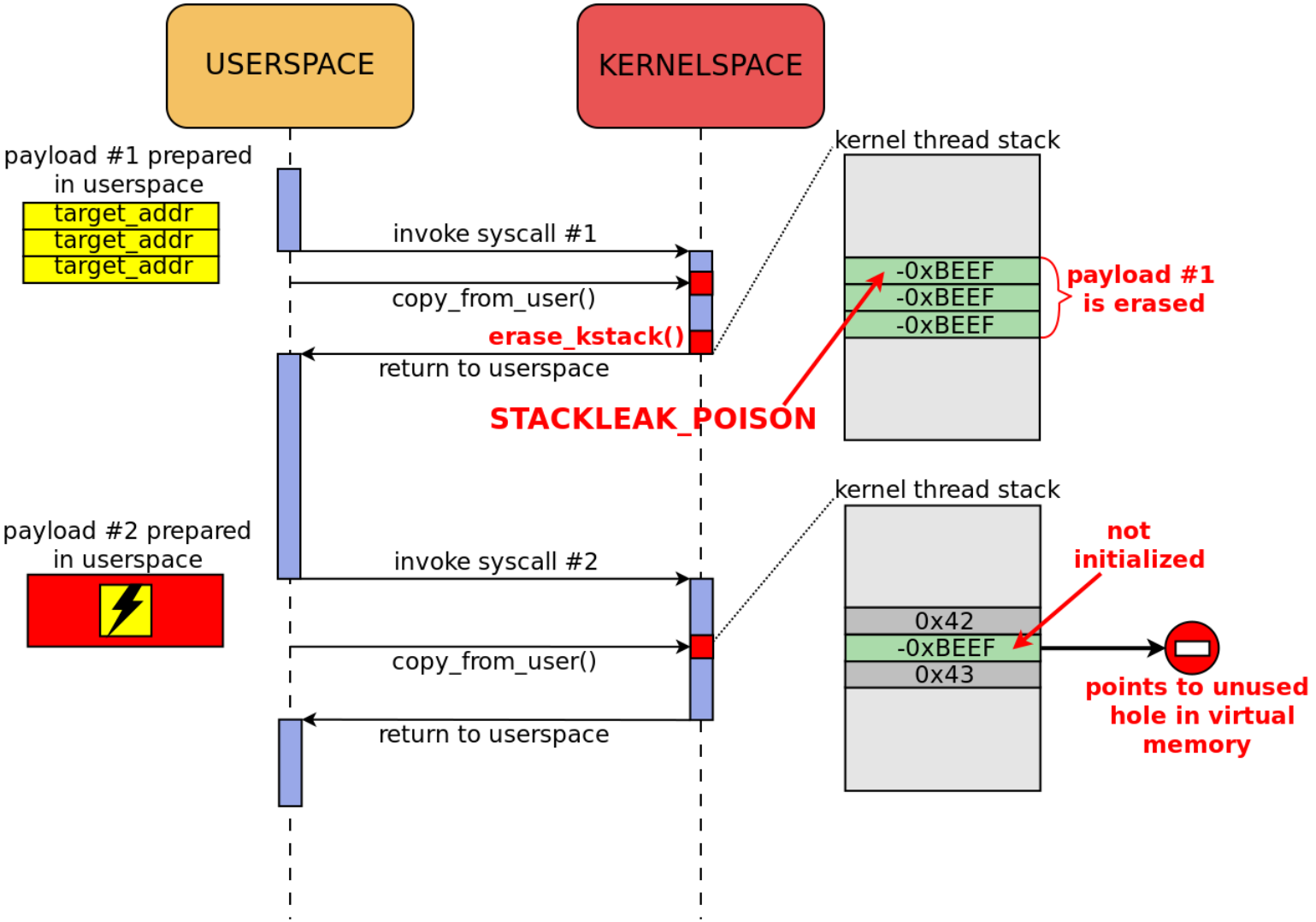
# STACKLEAK: свойство безопасности №2

- Блокирует некоторые* атаки на неинициализированные переменные в стеке ядра
- Примеры: эксплуатация CVE-2010-2963, CVE-2017-17712
- См. отличную статью Кейса Кука (Kees Cook): https://outflux.net/blog/archives/2010/10/19/cve-2010-2963-v4l-compat-exploit/

# Атака на неинициализированную переменную в стеке



# Блокирование атак такого типа



# STACKLEAK: свойство безопасности №3

Предоставляет динамические средства обнаружения переполнения стека ядра “в глубину” (kernel stack depth overflow)

# Взаимное влияние механизмов безопасности

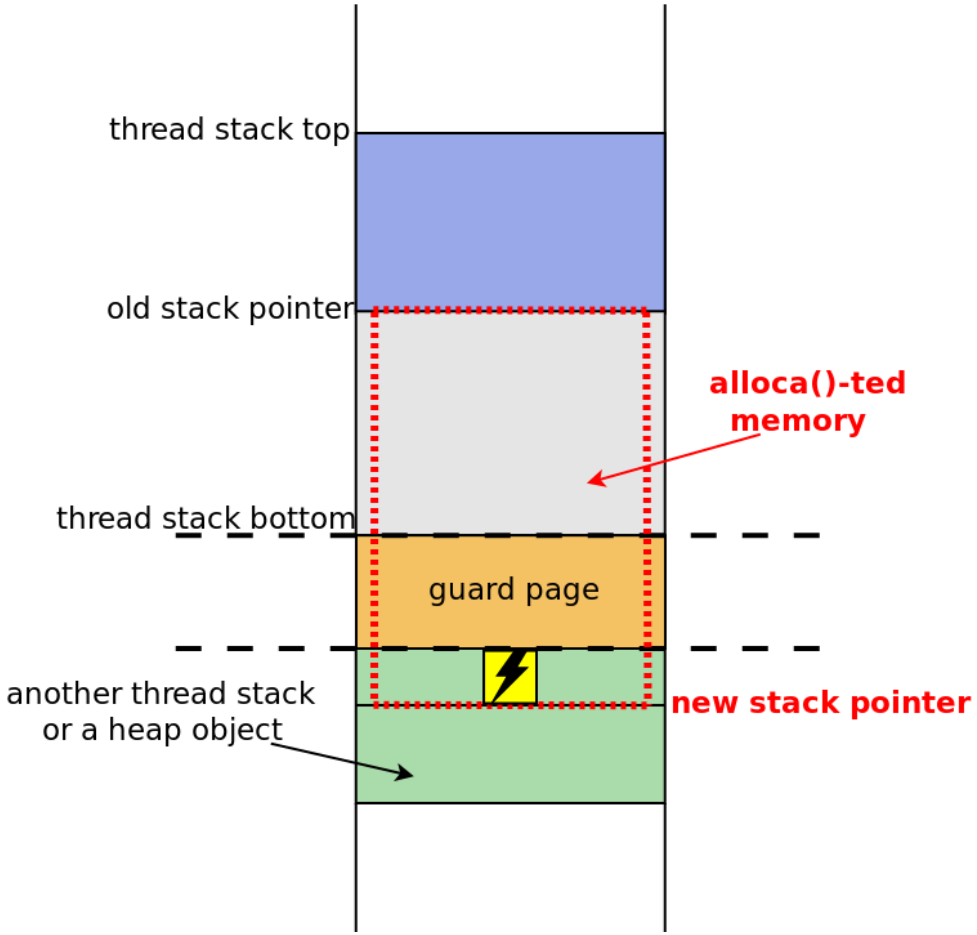
В ванильном ядре (Linux kernel mainline) STACKLEAK эффективен против переполнения стека “в глубину” только **в сочетании** с:

- `CONFIG_THREAD_INFO_IN_TASK`
- `CONFIG_VMAP_STACK` (оба внедрены Andy Lutomirski)

Виктор Васнецов, Богатыри (1898)

# Атака Stack Clash на ядерный стек

Идея: Gael Delalleau: "Large memory management vulnerabilities" (2005)
И еще раз: "The Stack Clash", Qualys Research Team (2017)



# STACKLEAK: защита от Stack Clash

- Статья о STACKLEAK и Stack Clash в блоге grsecurity: https://grsecurity.net/an_ancient_kernel_hole_is_not_closed.php
- Данный код исполняется перед каждым вызовом `alloca`:

```
    if (alloca_size >= stack_left) {
#if !defined(CONFIG_VMAP_STACK) && defined(CONFIG_SCHED_STACK_END_CHECK)
        panic("alloca over the kernel stack boundary\n");
#else
        BUG();
#endif
    }
```

# Здорово! А сколько это стоит? (1)

**Краткий** тест производительности на x86_64
**Оборудование:** Intel Core i7-4770, 16 GB RAM
**Тест №1, привлекательный:** сборка ядра Linux с Ubuntu config

```
time make -j9
```

```
Result on 4.11-rc8:

    real 32m14.893s
    user 237m30.886s
    sys 11m12.273s

Result on 4.11-rc8+stackleak:

    real 32m26.881s (+0.62%)
    user 238m38.926s (+0.48%)
    sys 11m36.426s (+3.59%)
```

# Здорово! А сколько это стоит? (2)

**Краткий** тест производительности на x86_64

**Оборудование:** Intel Core i7-4770, 16 GB RAM

**Тест №2, непривлекательный:**

```
hackbench -s 4096 -l 2000 -g 15 -f 25 -P
```

```
Average on 4.11-rc8: 8.71s

Average on 4.11-rc8+stackleak: 9.08s (+4.29%)
```

# Здорово! А сколько это стоит? (3)

## Выводы

1. Влияние STACKLEAK на производительность системы зависит от типа нагрузки
2. Оценивайте производительность STACKLEAK для планируемой нагрузки перед промышленной эксплуатацией

# Текущее состояние

Серия патчей v12 (16.05.2018) для x86_64 и x86_32

http://www.openwall.com/lists/kernel-hardening/2018/05/16/1

23 файла изменены, 949 строк добавлено, 19 удалено

Патчи:

1. gcc-plugins: Clean up the cgraph_create_edge* macros
2. x86/entry: Add STACKLEAK erasing the kernel stack at the end of syscalls
3. gcc-plugins: Add STACKLEAK plugin for tracking the kernel stack
4. lkdtm: Add a test for STACKLEAK (разработано вместе с Tycho Andersen)
5. fs/proc: Show STACKLEAK metrics in the /proc file system
6. doc: self-protection: Add information about STACKLEAK feature

# Уроки работы в сообществе

Расскажу несколько историй...

# Уроки работы в сообществе

Василий Перов, Охотники на привале (1871)

# Уроки работы в сообществе

- Начинай с объяснения цели работы
- Выдавай результаты чаще
- «Да, Брэд, ты абсолютно прав»
- Перечитывай их письма не менее 8 раз
- А свои ответы (до отправки) - не менее 16 раз
- Держи удар (Линус не церемонится)
- Постепенно расширяй список получателей
- Будь гибким и при этом настойчивым

# В заключение

- Мы сообщество разработчиков и пользователей GNU/Linux
- Давайте заниматься безопасностью нашей любимой ОС! Это очень интересно!

# Спасибо! Вопросы?

alex.popov@linux.com
@a13xp0p0v