Экспериментируем с уязвимостями и средствами защиты ядра Linux с помощью kernel-hack-drill

Александр Попов
Positive Technologies
alex.popov@linux.com

Введение

На предстоящей конференции OS DAY 2026 я хотел бы представить свой открытый проект kernel-hack-drill, который предоставляет экспериментальный полигон для эксплуатации уязвимостей в ядре Linux: https://github.com/a13xp0p0v/kernel-hack-drill.

Набор инструментов kernel-hack-drill позволяет исследователям безопасности экспериментировать с различными ошибками повреждения памяти в ядре Linux, разрабатывать методы их эксплуатации и изучать работу средств самозащиты ядра.

Я создал данный проект в 2017 году для своих студентов. Идея оказалась очень удачной, и с помощью kernel-hack-drill я также выполнил ряд собственных исследований:

- разработал прототип эксплойта для сложнейшего состояния гонки CVE-2024-50264 в ядре Linux: https://habr.com/ru/companies/pt/articles/953314/
- обнаружил ряд особенностей объекта pipe_buffer в ядре Linux, которые интересны для атакующего https://a13xp0p0v.github.io/2026/04/20/pipe-buffer-experiments.html

Технические подробности

kernel-hack-drill — это открытый проект, опубликованный под лицензией GPL-3.0. В его составе:

- `drill_mod.c` — исходный код небольшого модуля ядра Linux, который предоставляет простой интерфейс в пользовательском пространстве через файл `/proc/drill_act`. Этот модуль содержит синтетические уязвимости, с которыми удобно экспериментировать.
- `drill.h` — заголовочный файл, описывающий интерфейс модуля `drill_mod.ko`:

```
enum drill_act_t {
        DRILL_ACT_NONE = 0,
        DRILL_ACT_ALLOC = 1,
        DRILL_ACT_CALLBACK = 2,
        DRILL_ACT_SAVE_VAL = 3,
        DRILL_ACT_FREE = 4,
        DRILL_ACT_RESET = 5
};

#define DRILL_ITEM_SIZE 95

struct drill_item_t {
        unsigned long foobar;
        void (*callback)(void);
        char data[]; /* C99 flexible array */
};
```

- `drill_test.c` — тест для `drill_mod.ko` с примерами использования `/proc/drill_act`, который запускается из пользовательского пространства. Этот тест аккуратно

взаимодействует с `drill_mod.ko` без повреждения ядерной памяти и успешно завершается в том числе при `CONFIG_KASAN=y`.

- `README.md` — подробное пошаговое руководство по настройке и использованию `kernel-hack-drill` (спасибо контрибьюторам проекта, которые поучаствовали в его написании).

Проект kernel-hack-drill немного похож на KRWX, но гораздо проще. Кроме того, он содержит целый набор готовых прототипов эксплойтов для синтетических уязвимостей в `drill_mod.ko` (это главное отличие проекта от аналогов):

- `drill_uaf_callback.c` — UAF-эксплойт (use-after-free, использование памяти после освобождения), вызывающий функцию `callback` из освобожденной структуры `drill_item_t`. Он перехватывает поток управления в ядре и выполняет локальное повышение привилегий.
- `drill_uaf_callback_rop_smep.c` — доработанная версия `drill_uaf_callback.c` с ROP-цепочкой для обхода средств защиты SMEP и Page Table Isolation (PTI) на `x86_64`.
- `drill_uaf_callback_rop_smap.c` — усовершенствованная версия предыдущего эксплойта, размещающая ROP-цепочку в пространстве ядра для нейтрализации аппаратного средства защиты SMAP на `x86_64`.
- `drill_uaf_w_msg_msg.c` — UAF-эксплойт, выполняющий запись в освобожденную структуру `drill_item_t`. Он выполняет кросс-кэш-атаку и перезаписывает размер ядерного объекта `msg_msg`, что позволяет читать память ядра за границей буфера. Я написал этот PoC-эксплойт во время исследования уязвимости CVE-2024-50264.
- `drill_uaf_w_pipe_buffer.c` — UAF-эксплойт, также выполняющий запись в освобожденную структуру `drill_item_t`. Он выполняет кросс-кэш-атаку и перезаписывает `pipe_buffer.flags`, чтобы реализовать технику Dirty Pipe и добиться локального повышения привилегий. Этот PoC-эксплойт тоже был разработан во время моих экспериментов с CVE-2024-50264.
- `drill_uaf_w_pte.c` — UAF-эксплойт, выполняющий запись в освобожденную структуру `drill_item_t`. Он выполняет кросс-аллокаторную атаку и модифицирует запись в таблице страниц (PTE), чтобы реализовать технику Dirty Page Table и выполнить локальное повышение привилегий на `x86_64`.
- `drill_uaf_w_pud.c` — улучшенная версия `drill_uaf_w_pte.c`, которая перезаписывает не PTE, а запись в Page Directory Pointer Table (PDPT), которая в ядре Linux называется Page Upper Directory (PUD). Это позволяет реализовать атаку Dirty Page Table через большие страницы (huge pages).
- `drill_oob_w_pipe_buffer.c` — прототип эксплойта для ошибки записи за границей массива (OOBW, out-of-bounds write) в ядре Linux. Он выполняет повреждение указателя `page` в объекте `pipe_buffer` для получения возможности произвольного чтения и записи ядерной памяти. В результате атаки выполняется локальное повышение привилегий в системе.

Что нового?

За последние несколько месяцев в инструмент kernel-hack-drill были добавлены новые прототипы эксплойтов:

- drill_uaf_callback_rop_smep.c и drill_uaf_callback_rop_smap.c, в которых перехват потока управления усовершенствован с помощью возвратно-ориентированного программирования (return oriented programming, ROP);
- drill_oob_w_pipe_buffer.c, в котором эксплуатируется ошибка записи за границей массива (OOBW, out-of-bounds write) в ядре Linux.

А также за последние 3 недели я переосмыслил и усовершенствовал технику кросс-кэш атаки в ядре Linux. Были переработаны следующие прототипы эксплойтов:

- drill_uaf_w_msg_msg.c
- drill_uaf_w_pipe_buffer.c
- drill_uaf_w_pte.c
- drill_uaf_w_pud.c

Эта техника атаки особенно актуальна для свежих версий ядра Linux со включенными механизмами защиты ядерного аллокатора CONFIG_RANDOM_KMALLOC_CACHES и CONFIG_SLAB_BUCKETS.

Об этой атаке и способах ее отладки в kernel-hack-drill я бы хотел подробнее рассказать своем выступлении.

Проект kernel-hack-drill — отличный инструмент в руках исследователя безопасности ядра Linux. Он позволяет получать ценные знания на практике. Я буду рад рассказать об этом проекте на конференции OS DAY 2026.

Спасибо!