Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Операционные системы

Лабораторная работа №3

Группа: P3324

Выполнил: Маликов Глеб Игоревич

Проверил:

Клименков Сергей Викторович

Санкт-Петербург

2024г.

Оглавление

Задание

Вариант

- ОС: Linux
- ioctl: vm areas

Интерфейс передачи между программой пользователя и ядром: ioctl - передача параметров через управляющий вызов к файлу/устройству.

Целевая структура: Именем структуры в заголовочных файлах Linux.

Задание

Разработать комплекс программ на пользовательском уровне и уровне ярда, который собирает информацию на стороне ядра и передает информацию на уровень пользователя, и выводит ее в удобном для чтения человеком виде. Программа на уровне пользователя получает на вход аргумент(ы) командной строки (не адрес!), позволяющие идентифицировать из системных таблиц необходимый путь до целевой структуры, осуществляет передачу на уровень ядра, получает информацию из данной структуры и распечатывает структуру в стандартный вывод. Загружаемый модуль ядра принимает запрос через указанный в задании интерфейс, определяет путь до целевой структуры по переданному запросу и возвращает результат на уровень пользователя.

Интерфейс передачи может быть один из следующих:

1. syscall - интерфейс системных вызовов.
2. ioctl - передача параметров через управляющий вызов к файлу/устройству.
3. procfs - файловая система /proc, передача параметров через запись в файл.
4. debugfs - отладочная файловая система /sys/kernel/debug, передача параметров через запись в файл.

Целевая структура может быть задана двумя способами:

1. Именем структуры в заголовочных файлах Linux
2. Файлом в каталоге /proc. В этом случае необходимо определить целевую структуру по пути файла в /proc и выводимым данным.

Решение

Общая схема взаимодействия.

Комплекс программ (модуль ядра + утилита на уровне пользователя):

- Принимает на вход PID из аргументов командной строки в пользовательской программе.
- Посылает этот PID в модуль ядра через ioctl.
- Модуль ядра собирает информацию о VMA (vm_area_struct) целевого процесса.
- Модуль ядра возвращает информацию обратно в пользовательскую программу.
- Пользовательская программа выводит результат в удобном для чтения виде.

Последовательность запуска

Для сборки использовался CLion

1. Соберите vma_driver.c и vma_user.c
2. Напишите свой ключ `x509.genkey`. В текущем файле необходимо указать собственное значение CN.
3. Сгенерируйте пару ключей запуская `key_creation.sh`.
4. С готовой парой ключей запустите `key_installation.sh`.
5. Перезапустите систему. При перезагрузке если у вас Linux с UEFI Secure Boot, то выберите Enroll MOK > Continue > Yes > Введите пароль установленный в предыдущем шаге.
6. Для подтверждения установки ключа можете запустить `mokutil -l`.
7. Подпишите `vma_driver.ko` с помощью `sign_module.sh`.
8. Загрузите модуль в систему с помощью `load_driver.sh`.
9. Запустите `vma_user`. Пример: `sudo cmake-build-debug/user/vma_user 3640`, где 3640 - PID. Вставьте соответсвующий путь к `vma_user`.

Исходный код

Исходный код доступен по репозиторию:
https://github.com/glebmavi/OperatingSystems_Lab3

Заключение

В ходе выполнения работы был разработан комплекс программ, включающий модуль ядра и утилиту на уровне пользователя, который реализует передачу данных через интерфейс ioctl для сбора и отображения информации о виртуальных областях памяти (VMA) целевого процесса.