Улучшения кода ранних этапов загрузки ядра ОС Linux

Басков Евгений Сергеевич

Студент

Факультет ВМК МГУ имени М. В. Ломоносова, Москва, Россия

E-mail: baskov@ispras.ru

Научный руководитель *— Хорошилов Алексей Владимирович*

Данный доклад посвящен проблеме безопасности ядра операционной системы (ОС) Linux на этапе её развёртывания и инициализации на платформе `x86_64`. В настоящее время в ОС, использующих ядро Linux, многие техники повышения защищенности кода применяются либо во время работы прошивки, до передачи управления ОС, либо при уже полностью инициализированной и работающей ОС. Поэтому устойчивость к возможным атакам на этапе ранней загрузки в ядре Linux может быть улучшена. Одной из простых, но при этом эффективных техник защиты, является политика `W^X` (Write-xor-Execute).

Она заключается в запрете на использование регионов памяти доступных одновременно на запись и исполнение. Её применение не позволяет атакующему записать необходимый код и сразу передать на него управление, усложняя задачу выполнения произвольного кода. Эта политика активно используется в других ОС, а также применяется в ядре Linux, но на более поздних этапах его работы.

Основная сложность реализации `W^X` заключается в корректном определении регионов памяти и установке требуемых ими атрибутов памяти, так как

- в процессе загрузки механизмы страничной адресации не всегда доступны и не во всех случаях управляются непосредственно ядром;
- память повторно используется для регионов с различными требуемыми атрибутами;
- регионы перемещаются на перекрывающиеся с другими регионами адреса.

Ядро Linux хранится на диске, как файл, содержащий ядро в сжатом виде и небольшой код ранней загрузки, задачей которого

является распаковка образа ядра и подготовка окружения для продолжения инициализации (см. Рис. 1). Этот код поддерживает множество интерфейсов загрузки, но все они передают управление на общую для всех интерфейсов часть пути исполнения. Этим вызвана необходимость повторной инициализации состояния процессора и перемещения ядра. Однако при загрузке с использованием интерфейсов 64-битных UEFI (узлы, выделенные пунктиром на Рис. 1) окружение известно, и, если избежать использования общей для всех путей исполнения части кода и распаковывать ядро напрямую из окружения, предоставляемого UEFI-совместимой прошивкой, можно значительно упростить код загрузки ядра и избежать необходимости копирования образа ядра.

Иллюстрации

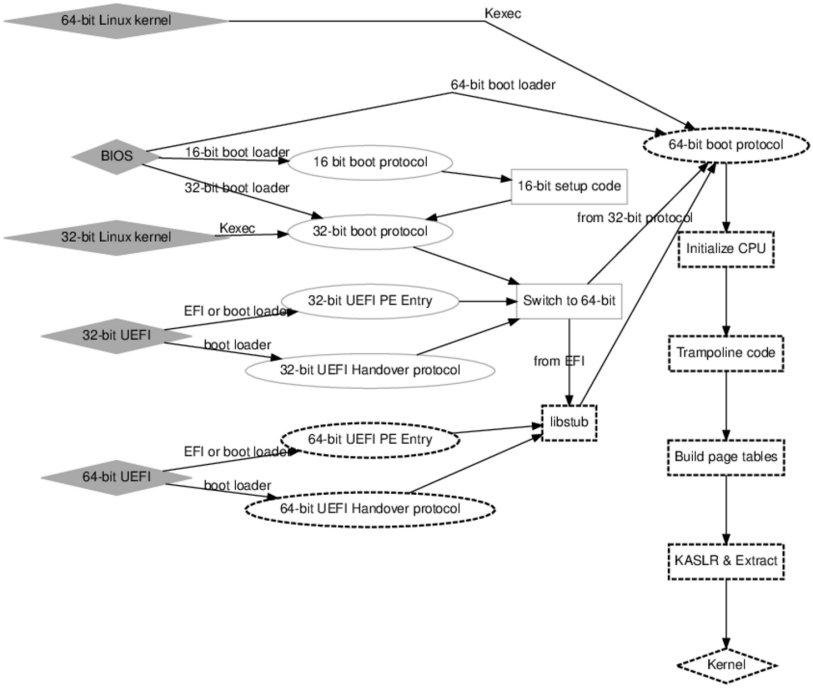

Рис. 1. Схема ранней загрузки

В рамках данной работы были исправлены некоторые проблемы совместимости кода ядра со стандартом UEFI, что позволяет повысить безопасность процесса загрузки за счёт возможности использовать прошивки, которые устанавливают более строгие строгие атрибуты памяти на регионы, выделяемые прошивкой. Была повышена безопасность кода общей части пути исполнения, реализацией поддержки политики `W^X` везде, где это применимо. А также был добавлен отдельный путь исполнения для загрузки с использованием UEFI-совместимых прошивок, убирающий перемещения ядра и повторное использование памяти, что позволяет реализовать `W^X` в полном объеме для этого способа загрузки.