Оглавление

Введение

TODO

1 Анализ предметной области

В данном разделе будут введены основные определения и будет обоснована актуальность задачи внесения изменений в ядро операционной системы Linux.

1.1 Актуальность

Программное обеспечение с открытым исходным кодом имеет такие преимущества, как скорость разработки и надежность. Причиной этого является большое число участников разработки. Каждый разработчик находит ошибки, предсавляет их решение и предлагает новый функционал. Так, осуществляется непрерывный процесс внесения изменений.

Ядро Linux - программное обеспечение с открытым исходным кодом, поэтому непрерывно разрабатывается специалистами со всего мира. Добавление новых функций, внесение усовершенствований, исправление ошибок делают актуальной проблему внесения изменений в ядро операционной системы Linux.

1.2 Основные определения

Ядро операционной системы - это программное обеспечение, которое предоставляет базовые функции для всех остальных частей операционной системы, управляет аппаратным обеспечением и распределяет системные ресурсы [1].

Одной из характеристик ядра Linux является динамическая загрузка модулей ядра. Другими словами, при необходимости существует возможность динамической загрузки и выгрузки исполняемого кода во время работы системы. Так, можно переопределять или дополнять функции ядра. Файлы с исправлениями встраиваются в виде загружаемых модулей ядра.

Технику внесения изменений в код или данные, позволяющую модифицировать поведение целевого алгоритма требуемым образом, называют патчингом [2]. Патчи - это небольшие добавочные изменения, вносимые в ядро.

Каждый патч содержит изменение ядра, реализующее одну независимую модификацию.

При применении патча к ядру операционной системы необходимо знать состояние функций ядра, используется ли функция в момент исправления или нет. Для этого используется метод обхода стека.

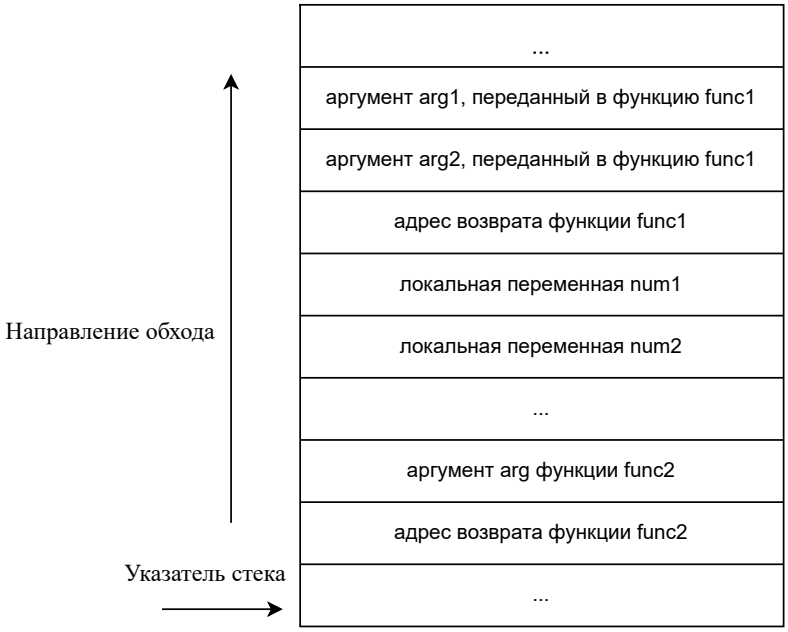

Рис. 1.1: Метод обхода стека

В методе обхода стека проверяется стек вызовов каждого потока ядра, показанный на рисунке 1.1. Необходимо определить, выполняется ли поток в функции, которую необходимо изменить. Для этого копия указателя стека уменьшается пока значение не достигнет нижней части стека. Функция используется, если адрес, принадлежащий этой функции можно найти в стеке.

1.3 Вывод

Были изучены основные определения и была обоснована важность проблемы изменения ядра операционной системы Linux.

2 Классификация существующих методов

В данном разделе будут описаны существующие методы решения, выделены критерии их оценки и будет проведено сравнение описанных методов по выделенным критериям.

2.1 Методы решения

Существуют следующие методы изменения ядра Linux:

- требующие перезагрузки системы;
- переноса;
- динамические.

2.1.1 Методы, требующие перезагрузки системы

Первые применения патчей к ядру происходили по следующему алгоритму:

- работающие приложения закрываются;
- происходит загрузка и иниациализация исправленного ядра;
- приложения перезагружаются.

Так, неисправленное ядро заменялось исправленным ядром.

В процессе внесения изменений в ядро операционной системы описанным способом возникает следующая проблема: появляется время простоя, которое состоит из времени простоя приложения и простоя ядра, что показано на рисунке 2.1.

Рис. 2.1: Метод, требующий перезагрузки системы

Для снижения времени простоя появились модификации метода, которые эффективно управляют перезагрузкой ядра. Одна из таких модификаций - бесшовный метод, который сокращает время простоя приложений.

При помощи системного вызова *kexec* происходит загрузка нового образа ядра. Чтобы применить патч необходимо дождаться момента, когда система примет состояние, в котором выполнены два условия:

- все потоки ядра остановлены;
- структуры данных ядра согласованны.

Выполнение этих двух условий позволяет сделать контрольную точку, которая позволяет сохранить состояние приложений. Код контрольной точки проходит через структуры данных ядра, связанные с приложениями, и преобразует их в высокоуровненый формат, который не зависит от версии ядра.

После сохранения контрольной точки выполняется инициализация нового ядра. Исправленное ядро считывает контрольную точку и перезапускает приложения, что требует перезагрузки системных вызовов.

Техника бесшовного метода показана на рисунке 2.2.

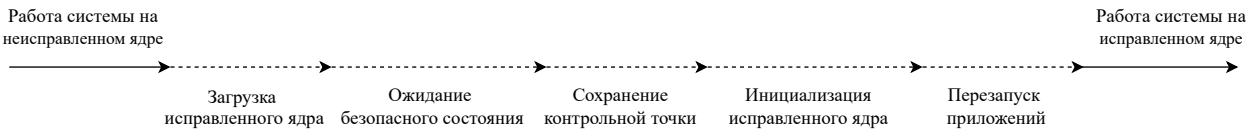

Рис. 2.2: Бесшовный метод

Существует метод, который сокращает время простоя путем одновременного выполнения приложений и перезагрузки системы - метод теневой перезагрузки. Перезагрузка операционной системы выполняется в фоновом режиме на виртуальной машине. Приложения могут продолжать выполняться на исходной машине.

После применения патча на выделенной виртуальной машине, делается его снимок, из которого восстанавливается файловая система.

TODO: рисунок для теневой перезагрузки.

Следующие методы решают проблему простоя исключением перезагрузки системы.

2.1.2 Метод переноса

Идея данного метода [4] заключается в следующем: на другом компьютере запускается измененное ядро, на него переносятся запущенные процессы старого ядра, и оно останавливается. В существующих решениях в качестве другого компьютера используется виртуальная машина, установленная на физической машине, требующей обновления ядра. Необходимо общее хранилище(сервер), которое подключено и к старому, и к новому ядрам. Патч применяется в три этапа.

На первом этапе собирается информация о состоянии операционной системы: подсчитывается число потоков, выполняемых в исправляемом коде, вызывается функция запуска, выполняется инициализация перед передачей управления виртуальной машине.

На втором этапе сервер начинается исправление структур данных и функций. Если измененный модуль не находится в состоянии покоя, обе версии структур данных должны существовать во время процесса исправления. Для обеспечения согласованности структур данных, страницы исходных данных и новых данных защищены. Перехват доступа к ним контролируется виртуальной машиной. То есть, при попытке изменить отслеживаемую страницу управление будет передано виртуальной машине. В этот момент сравнивается содержимое двух версий и вызывается функцию передачи состояния.

На последнем этапе отключается контроль исходных структур данных. Для этого используется метод обхода стека. В случае, если соответствующая исходные структуры используются, адрес возврата исходной функции заменяется адресом функции-заглушки, которые позволяют определить, используется ли еще обновленная функция. Затем выполняется очистка кода старой версии и устанавливается флаг завершения патчинга.

При применении данного метода время простоя невелико. Главным минусом является высокое потребление ресурсов центрального процессора, сети и объема памяти.

TODO: рисунок для переноса.

2.1.3 Динамические методы

Данные методы [4] позволяют применять патчи во время выполнения процессов. Решение состоит из двух этапов.

Для того, чтобы создать измененный код, проводится анализ обновленной и старой версий. Для этого собирается два варианта ядра: сборка неизмененного кода и сборка измененного кода. В отличие от поиска различий в исходном коде, анализ объектного кода позволяет понять, какие функции были изменены в патче. Большинство функций ядра, которые не были изменены патчем, будут иметь одинаковые объектные коды. Обнаруженные измененные функции помещаются в основной модуль для загрузки в ядро.

Следующий этап необходим для обнаружения встраиваемых функций и неуникальных символов. Для этого проводится сравнение работающего кода ядра и скомпилированного кода. Этот процесс называется предварительным сопоставлением.

Для применения данного метода необходимо определить состояние ядра, когда каждая заменяемая функция будет находиться в состоянии покоя, что является условием безопасного внесения изменений. В этом случае модуль может быть загружен в ядро. Проверка условия безопасного внесения изменений в случае неудачи возобновится через некоторое время. После нескольких неудачных попыток достичь безопасного состояния для применения патча процесс прерывается. Новые инструкции будут вставлены в необновленные функции путем бинарной перезаписи. Бинарная перезапись - это перенаправление вызова функции из исходной в исправленную, для чего используется прыжок к первым пяти или шести байтам функции.

Некоторые решения на основе динамических методов не поддерживают семантические изменения. Кроме того, возникают сложности с изменением типов и структур данных, нестабильных типов данных и функций ядра, которые всегда находятся в стеке вызовов потоков ядра.

Данные методы решают проблему с временем простоя.

TODO: рисунок

2.2 Критерии оценки методов

TODO

2.3 Сравнение методов

TODO

2.4 Вывод

TODO

Выводы

TODO

Литература

[1] Love Robert. Linux System Programming. 2013.

[2] Overview of Dynamic Operating System's Kernel Hooking Methods (Study Case of Linux Kernel). Режим доступа: https://www.flow3d.com/modeling-capabilities/waves/ (дата обращения: 04.10.2021).

[3] Shortening Downtime of Reboot-Based Kernel Updates Using Dwarf. Режим доступа: https://www.flow3d.com/modeling-capabilities/waves/ (дата обращения: 04.10.2021).

[4] Rebootless Security Patches for the Linux Kernel. Режим доступа: https://www.flow3d.com/modeling-capabilities/waves/ (дата обращения: 04.10.2021).