УДК 004.451

Очистка освобождаемых областей оперативной памяти в процессе освобождения страниц в GNU/Linux

В. С. Прокопов
ЗАО «ОКБ САПР», Москва, Россия

Рассмотрена актуальность очистки освобождаемых областей оперативной памяти. Предлагается метод, который позволяет очищать всю освобождаемую оперативную память в момент освобождения ядром страниц оперативной памяти.

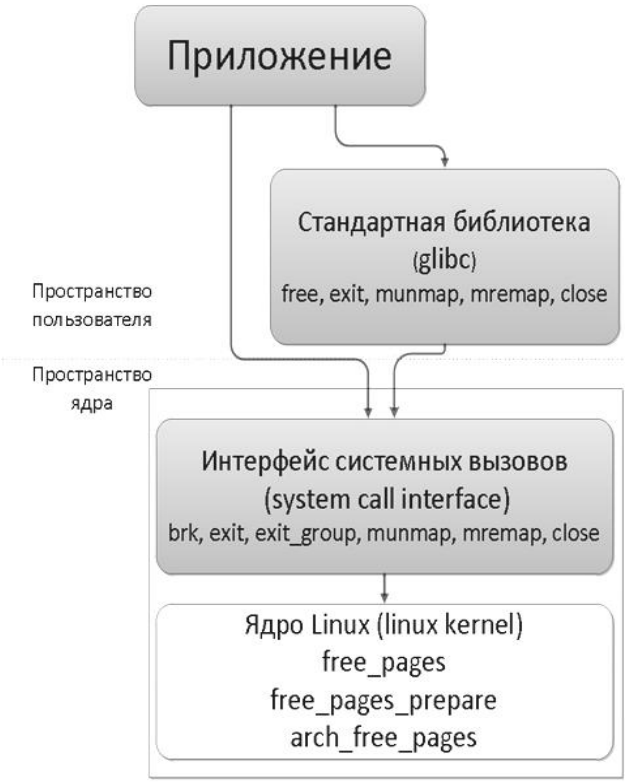
Ключевые слова: Linux, очистка памяти, free_pages, brk, free_pages_prepare.

В настоящее время ОС на базе ядра Linux все больше используются, в том числе и в государственных организациях, а, следовательно, и появляется потребность в сертифицированных СЗИ НСД для Linux. Согласно руководящим документам РФ для успешного прохождения процедуры сертификации на определенные классы (5-й класс СВТ и выше) средствам защиты от НСД необходим механизм очистки освобождаемых областей оперативной памяти.

На рисунке изображена схема освобождения областей оперативной памяти — приложение может очищать память, либо используя стандартную библиотеку glibc, либо напрямую вызывая системные вызовы. Поскольку приложение может и не использовать стандартную библиотеку, подсистему очистки освобождаемой оперативной памяти можно реализовывать либо на уровне интерфейса системных вызовов, либо на уровне освобождения страниц.

В статье [1] был рассмотрен способ реализации очистки освобождаемой оперативной памяти в ОС Linux на уровне интерфейса системных вызовов. Данный подход основывался на перехвате системных вызовов — brk, munmap, mremap, exit, exit_group, close и имел определенные недостатки: невозможность очищать разделяемые области памяти, области памяти, принадлежащие ядру ОС и приводил к заметному снижению производительности.

Второй подход — реализация подсистемы очистки освобождаемой оперативной памяти на уровне освобождения страниц — лишен этих недостатков. Данный подход заключается в том,



Освобождение областей оперативной памяти

чтобы модифицировать поведение функции/ функций, освобождающих страницы оперативной памяти.

Есть два пути изменить поведение ядра ОС — изменить исходный код и пересобрать ядро Linux [2] либо загрузить модуль ядра, перехватывающий функции, поведение которых нужно поменять. Примером следования первому пути является патч к ядру Linux — PaX. В рамках своих функций он может производить очистку освобождаемых страниц оперативной памяти. Однако использование патча к ядру создает трудности в установке и эксплуатации комплекса СЗИ НСД. Для установки комплекса на дистрибутив Linux

Прокопов Вячеслав Сергеевич, инженер-программист.
E-mail: vprokopov@okbsapr.ru.

Статья поступила в редакцию 14 июня 2014 г.

необходимо найти исходники ядра, затем приложить патч и пересобрать. При обновлении ядра Linux необходимо будет повторить процедуру. Кроме того, затрудняется сертификация — необходимо сертифицировать все ядро, так как оно в этом случае является частью комплекса. Среди плюсов данного пути можно выделить отсутствие любых ограничений на изменение поведения ядра. Из-за всего вышеперечисленного для создания подсистемы комплекса СЗИ НСД лучше подходит второй путь, который и рассмотрен в данной статье.

Рассмотрена реализация только для конкретной версии ядра (2.6.35), однако она с небольшими исправлениями может быть применима для более современных версий ядра. В ядре Linux рассматриваемой версии функцией, которая вызывается для освобождения страниц, является free_pages. Однако эта функция является всего лишь интерфейсом к функциям, которые производят реальное освобождение страниц ядром ОС и процессором (если архитектура процессора это допускает). В итоге вызова этой функции всегда вызывается функция free_pages_prepare. Данная функция принимает среди параметров дескриптор страницы, производит определенные проверки и вызывает функцию arch_free_pages, которая должна производить окончательное освобождение страниц. Но на большинстве архитектур, в том числе и на x86, данная функция является пустой и не производит никаких действий и ее нельзя перехватить из-за отсутствия ее адреса в таблице символов ядра, т. е. в зависимости от архитектуры необходимо перехватывать либо arch_free_pages, либо free_pages_prepare.

Перехват проще всего производить путем записи команды ud2 в начало кода функции, предварительно сохранив поврежденный участок начала функции. Данная команда вызывает обрабатываемое исключение, и в качестве обработчика этого исключения можно назначить функцию-перехватчик, и в функции перехватчике возвращать управление на сохраненный пролог, а затем в прологе — на оставшуюся часть функции. Для того чтобы определить длину перезаписанной команды, необходим код, вычисляющий длины команд (дизасемблер длин), что делает перехват функции ядра архитектурно-зависимым. Подробно этот процесс описан в [3].

В зависимости от архитектуры процессора, как было сказано выше, перехватывать нужно либо функцию arch_free_pages, либо free_pages_prepare. В последнем случае необходимо выполнить все проверки, которые проводит функция free_pages_prepare до вызова arch_free_pages, и в случае их успешного прохождения очищать группу освобождаемых страниц оперативной памяти. Одним из параметров этих функций является параметр order — степень, в которую необходимо возвести двойку, чтобы получить число страниц. Другим параметром является указатель на массив дескрипторов страниц, используя которые, можно очистить освобождаемые страницы.

Чтобы очистить страницу оперативной памяти, зная ее дескриптор, необходимо сначала отобразить ее в виртуальную память ядра. Это можно сделать при помощи двух функций — kmap и kmap_atomic. Первая функция использует внутри себя различные блокировки, и не может использоваться в атомарном контексте, в то время как функция free_pages_prepare может вызываться в таком контексте, и использование функции kmap может привести к панике ядра (kernel panic). Поэтому в данном случае можно использовать только функции kmap_atomic/kunmap_atomic. Кроме того, при очистке страницы необходимо отключить на время прерывания при помощи функции local_irq_save, соответственно после очистки включить функцией local_irq_restore.

Описанный подход обладает определенными преимуществами: он очищает всю память, освобождаемую ядром ОС, и практически не снижает производительность ОС. Недостатки — это сильная зависимость от архитектуры процессора и сильная зависимость от версии ядра.

Для создания средства или подсистемы очистки освобождаемых областей оперативной памяти, которое не замедляет работу ОС и очищает все освобождаемые участки оперативной памяти, лучше подходит способ, основанный на перехвате функции ядра Linux, освобождающей страницы оперативной памяти. Тем не менее, он не очищает сегмент кучи при уменьшении его размера посредством системного вызова brk. Таким образом, в данном подходе желательно перехватывать системный вызов brk (т. е. использовать оба подхода). Данный вопрос был подробно разобран в статье [1].

Литература

1. *Прокопов В. С., Каннер А. М.* Особенности реализации механизма очистки освобождаемых областей оперативной памяти в GNU/Linux //Комплексная защита информации. Электроника инфо. Материалы XVIII Международной конференции 21—24 мая 2013 года, — Брест (Республика Беларусь), 2013. С. 120—123.

2. *Бовет Д., Чезати М.* Ядро Linux. 3 изд. — СПб: БХВ-Петербург, 2007. — 1104 с.

3. Перехват функций ядра Linux с использованием исключений (kprobes своими руками) //http://habrahabr.ru/post/206778/

Cleaning of freed memory areas in the process of freeing pages in GNU / Linux

V. S. Prokopov
OKB SAPR JSC, Moscow, Russia

The article describes the topicality of cleaning freed memory areas. The approach based on cleaning of freed memory areas at the moment of freeing pages of random access memory is offered.

Keywords: Linux, memory cleaning, free_pages, brk, free_pages_prepare.

Bibliography —3 references. *Received June 14, 2014*