УДК 004.451

Особенности переноса стандартной библиотеки языка Си в режим безопасных вычислений архитектуры «Эльбрус»

Мустафин Т.Р., Алёхин А.И.

ЗАО «МЦСТ»

Одной из особенностей микропроцессоров семейства «Эльбрус» выступает возможность выполнения программ в безопасном режиме. Безопасные вычисления Эльбрус созданы для защиты программ от ошибок памяти, таких как выход за границы массива и переполнение буфера. Безопасные вычисления Эльбрус достигаются использованием дескрипторов для обращений в память [1], [2]. Построение стека программного обеспечения, работающего в безопасном режиме архитектуры Эльбрус, требует переноса в безопасный режим стандартной библиотеки языка Си. В качестве стандартной библиотеки языка Си в работе [3] выбрана uclibc-ng. В данной работе описаны принципы реализации в режиме безопасных вычислений отдельных частей библиотеки uclibc-ng: системных вызовов, динамического загрузчика и механизма управления динамической памятью.

1. Системные вызовы.

Для обращений в память в безопасном режиме пользовательские приложения используют 128-ми битные дескрипторы. Ядро операционной системы использует 64-х битные указатели. Во время системных вызовов аргументы-дескрипторы преобразуются в указатели, а возвращаемые указатели переводятся в дескрипторы. Передача параметров организована через окно из 14 двойных регистров по 64 бита. Каждый параметр, будь то целое число или дескриптор, библиотекой размещается на отдельном квадрорегистре (128 бит). Для каждого системного вызова ядро хранит битовую маску. Битовая маска указывает номера квадрорегистров содержащих аргументы-дескрипторы. Благодаря большому окну передачи параметров и использованию битовых масок в ядре преобразование аргументов-дескрипторов выполняется единым для большинства системных вызовов кодом. Индивидуальная обработка параметров требуется только системным вызовам, принимающим дескрипторы внутри структур, а также системным вызовам, возвращающим указатели.

2. Динамический загрузчик.

Процесс загрузки модулей в память, а также разрешения зависимостей в безопасном режиме имеет особенности, которые нужно учесть при переносе динамического загрузчика в данный режим. Исполняемые файлы содержат в себе как минимум два загружаемых сегмента: исполняемый, предназначенный только для чтения сегмент кода, а также неисполняемый, предназначенный для чтения и записи сегмент данных. Поскольку права доступа различные, в безопасном режиме доступ к ним осуществляется через различные дескрипторы. В сегменте кода содержится исполняемая часть файла. В сегменте данных содержится глобальная таблица смещений (global offset table - got) и глобальные переменные. Для проведения динамического связывания необходима инициализация got. Содержащиеся в got адреса функций и глобальных переменных в безопасном режиме являются дескрипторами, которые можно получить только на этапе исполнения программы. Поэтому для инициализации глобальной таблицы смещений в безопасном режиме используется функция selfinit(), включаемая компилятором в исполняемые файлы. Для загрузки модулей в память в защищённом режиме динамический компоновщик использует системный вызов uselib, который размещает в отдельных областях памяти сегменты кода и данных модуля и возвращает дескриптор функции selfinit() и дескриптор сегмента данных. Далее динамический компоновщик производит вызов функции selfinit() для инициализации got. После инициализации got динамический загрузчик производит разрешение зависимостей между модулями, обращаясь к глобальной таблице смещений через дескриптор сегмента данных.

3. Механизм заказов и освобождения динамической памяти.

Механизм заказов и освобождения динамической памяти является одним из слабых мест языка Си. В частности, неправильной последовательностью вызовов malloc()/free() порождаются проблемы зависших ссылок. Использование зависших ссылок на ранее освобожденные массивы памяти приводит к порче новых данных. Проблему зависших ссылок в режиме безопасных вычислений можно решить программно-аппаратным методом. Для этого между вызовом free() и выделением заново освобождённой памяти все копии дескриптора описывающего освобождаемую память удаляются из пользовательской памяти. Удаление копий производит ядро операционной системы. Благодаря механизму внешних тегов ядро отличает копию дескриптора от просто числового значения. Функция очистки из пользовательского пространства запускается системным вызовом. Переключение контекста занимает много времени, поэтому очистка запускается один раз на несколько вызовов free(). Реальное освобождение памяти и его выделение заново производится только после выполнения функции очистки. Таким образом можно построить защиту от зависших ссылок в режиме безопасных вычислений.

Предложенные принципы реализованы при переносе библиотеки uclibc-ng в режим безопасных вычислений Эльбрус. Характеристики библиотеки uclibc-ng позволяют выполнять в безопасном режиме не только синтетические тесты, но и реальные программы написанные на языке Си с использованием разделяемых библиотек и многопоточности.

Литература

1. Волконский В.Ю. Безопасная реализация языков программирования на базе аппаратной и системной поддержки // Вопросы радиоэлектроники, 2008. Т. 4. № 2. С. 98–141.

2. Ким А.К., Перекатов В.И., Ермаков С.Г. Микропроцессоры и вычислительные комплексы семейства «Эльбрус». СПб.: Питер, 2013. 272 с.

3. Мустафин Т.Р., Алехин А.И., Куян А.С., Кравцунов Е.М., Семенихин С.В. Выбор дистрибутива программного обеспечения для индустриальных и бортовых систем, использующего технологию защищенных вычислений архитектуры «Эльбрус» // Вопросы радиоэлектроники. 2017. № 3. С. 44–47.