Guardant®

Система защиты от компьютерного пиратства

Эффективная
защита приложений

Урок 1.5: автозащита
с использованием профайлера

Содержание

Извлечение инструкций из тела приложения 3

Установка автозащиты 4

Шаг 0. Подготовка приложения 4
Шаг 1. Создание проекта автозащиты 4
Шаг 2. Способ выбора защищаемых функций 5
Шаг 3. Выбор функций для защиты 7
Шаг 4. Сохранение файла описания защиты 8

Контрольные испытания 9

Заключение 10

Дополнительные источники информации 11

WWW: http://www.guardant.ru 11
Служба технической поддержки: 11

Извлечение инструкций из тела приложения

Автоматическая защита Guardant объединяет множество технологий. Одной из наиболее интересных является извлечение (по выбору разработчика) инструкций из кода приложения и трансляция их в код некой виртуальной машины (опция автозащиты **/RIP CODE**). Благодаря этому функции приложения получают защиту от анализа, а создание автоматических инструментов взлома автозащиты значительно затрудняется

При всех положительных качествах, использование технологий, подобных **/RIP CODE**, может негативно сказываться на производительности приложения, так как в процессе защиты инструкции подвергаются виртуализации. Кроме того, необходимо учитывать, что **не все** участки кода могут (и должны) быть защищены с использованием технологий, обладающих такими свойствами. Поэтому для повышения эффективности защиты необходимо **вручную определять набор функций**, подлежащих виртуализации.

Для этого служит **профилирование приложений**: с помощью специальных инструментов приложение анализируется (статически – при помощи дизассемблера, и динамически – в процессе исполнения), и, по результатам анализа, создается конфигурационный файл, содержащий информацию о функциях для защиты.

Следует отметить, что для выбора функций, подлежащих защите, с использованием профайлера разработчик должен хорошо представлять архитектуру защищаемого приложения.

В этом уроке будет рассматриваться методика работы с профайлером Guardant при установке автоматической защиты приложения.

Используемые термины и обозначения

Профайлер – совокупность программных инструментов для анализа скорости исполнения отдельных участков кода защищаемого приложения.

Базовый блок – набор последовательно идущих инструкций исполняемого файла, удовлетворяющих ряду требований и подлежащих защите при помощи технологии **RIP CODE**.

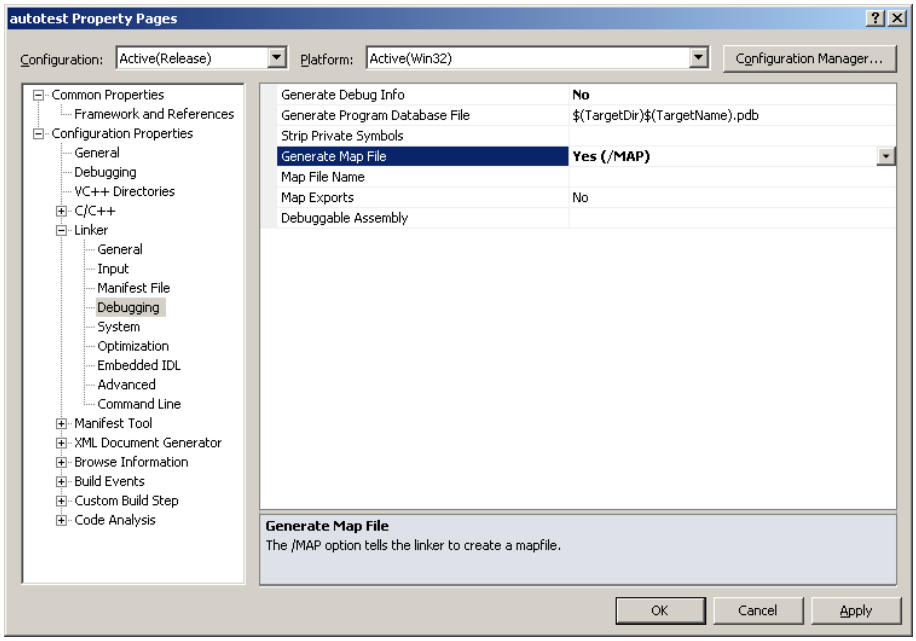
В рамках данного урока под **проектом автозащиты** понимается и проект лицензирования, и проект автозащиты (см. термины и обозначения урока 1.4).

Установка автозащиты

Будем рассматривать процедуру установки автозащиты с профилированием приложения на примере произвольного приложения, исходный код которого доступен.

Шаг 0. Подготовка приложения

Прежде всего, необходимо сгенерировать **MAP-файл** для защищаемого приложения. Сделать это можно, установив соответствующую опцию линкера (на примере Microsoft Visual Studio):



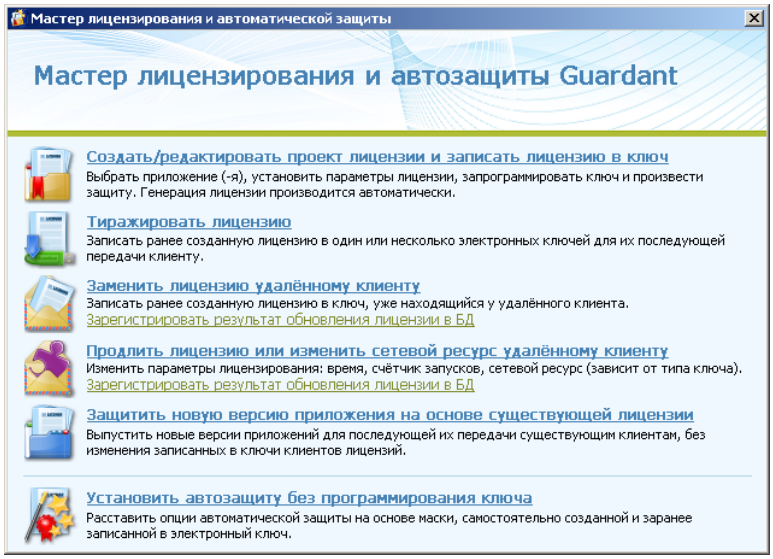
После очередной сборки проекта MAP-файл будет находиться в директории с исполняемым файлом (если не указано иное местоположение).

Примечание

1) При отсутствии MAP-файла будут защищаться случайные функции – без возможности их выбора.
2) В случае перекомпиляции приложения необходимо использовать новый MAP-файл.

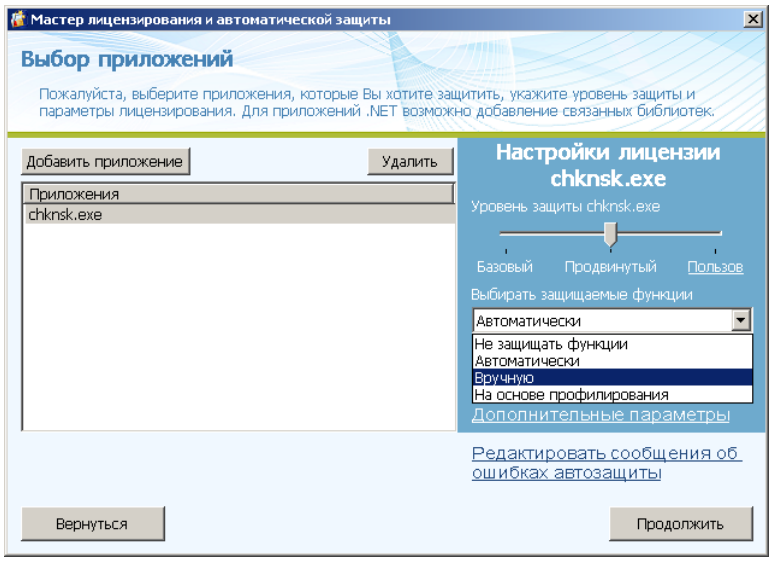
Шаг 1. Создание проекта автозащиты

Проект автозащиты создается любым из доступных способов (подойдет **1, 5 или 6 пункт** Мастера лицензирования и автозащиты). Также можно открыть любой ранее сохраненный проект:



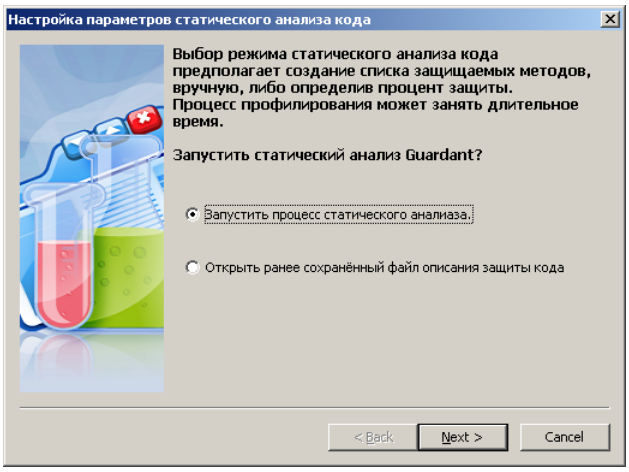
Шаг 2. Способ выбора защищаемых функций

Чтобы запустить профайлер, необходимо в диалоге **Выбор приложений** установить параметр **Выбирать защищаемые функции** в состояние **Вручную** или **На основе профилирования**:



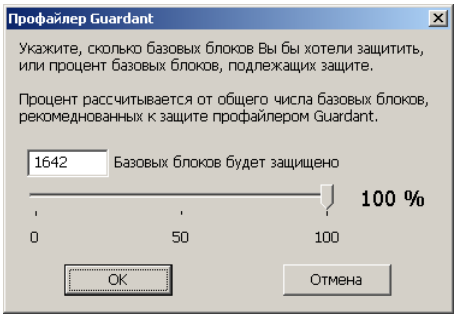
А. Вручную

Установка опции **Вручную** приведет к запуску профайлера и предложению начать процесс статического анализа приложения (или открыть ранее сохраненный файл описания защиты кода):



Так как используется Мастер автозащиты, пути к профилируемому файлу и результирующему файлу описания защиты кода (на следующих этапах диалога) заполняются им автоматически и недоступны для изменения.

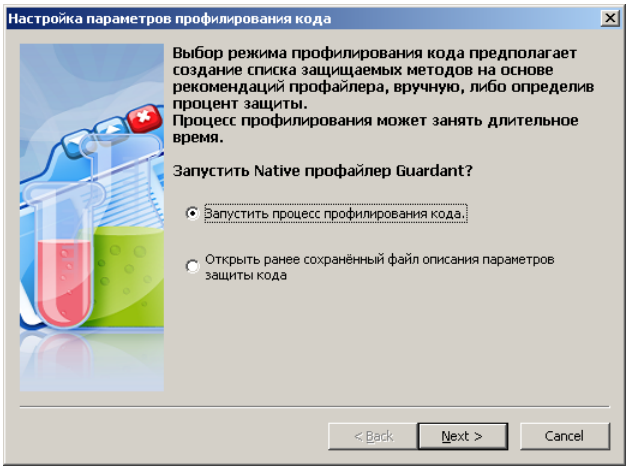
После успешного окончания процесса статического анализа кода приложения будет предложено выбрать **процент от числа базовых блоков, рекомендованных Профайлером** для защиты. В качестве примера **выберем 100%**:



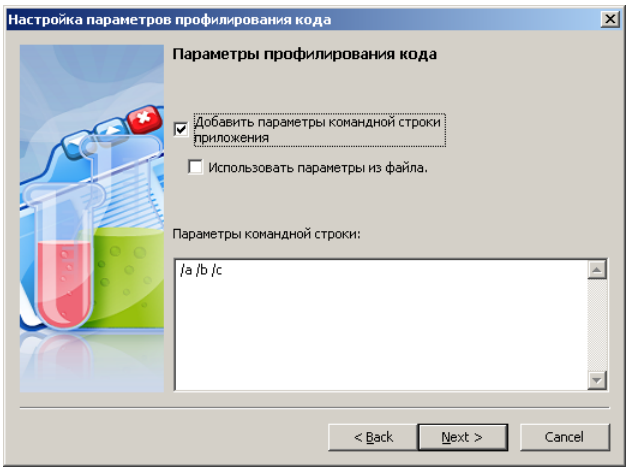
В результате будут автоматически отмечены для защиты все функции, **замедление работы которых будет наименее значительным** после установки автозащиты (к ним относятся функции без циклов и рекурсий).

Б. На основе профилирования

Установка опции **На основе профилирования** приведет к запуску профайлера с предложением **запустить процесс профилирования кода** (или открыть раннее сохраненный файл описания параметров защиты кода):



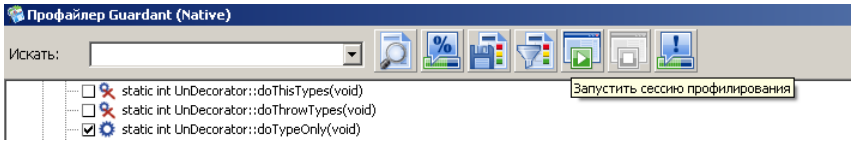
В одном из следующих окон диалога будет предложено задать **параметры командной строки** для **запуска профилируемого приложения**.



Пути к профилируемому исполняемому файлу и файлу описания параметров защиты останутся недоступными для изменения. Они будут автоматически заполнены Мастером лицензирования и автозащиты.

По завершении диалога будет автоматически выполнен статический анализ приложения.

Для запуска процесса **динамического анализа (профилирования)**, необходимо выбрать соответствующий элемент панели инструментов профайлера.

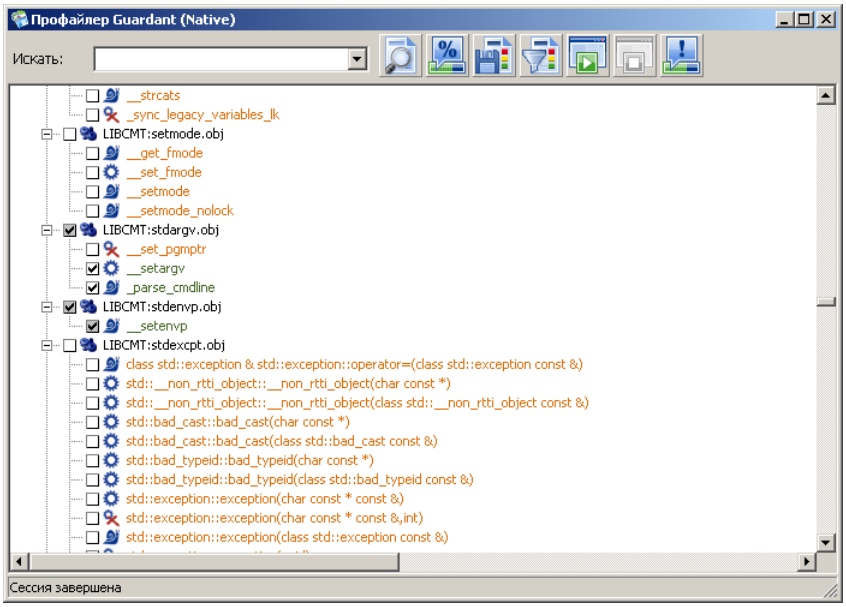


При профилировании происходит фактический запуск защищаемого приложения. Разработчику предлагается в течение некоторого времени поработать с приложением, используя наиболее востребованные функции и сценарии работы. В процессе работы с приложением профилировщик Guardant измеряет скорость работы отдельных функций и по результатам выдает рекомендации: какие функции вызывались достаточно часто, чтобы их имело смысл защищать, но недостаточно часто, чтобы их защита вызвала проблемы с производительностью.

Процесс профилирования останавливается нажатием соответствующей кнопки в окне профайлера или при завершении работы профилируемого приложения.

Шаг 3. Выбор функций для защиты

В результате выполнения предыдущего шага (после окончания статического или динамического анализа) в основном окне профайлера будут отображены обнаруженные функции защищаемого приложения.



Необходимо отметить, что в процессе установки автозащиты виртуализации подвергаются не функции целиком, а **базовые блоки** (наборы инструкции, удовлетворяющие определенным требованиям). Тем не менее, для удобства выбор базовых блоков для защиты основан на выборе функций, в которых они содержатся.

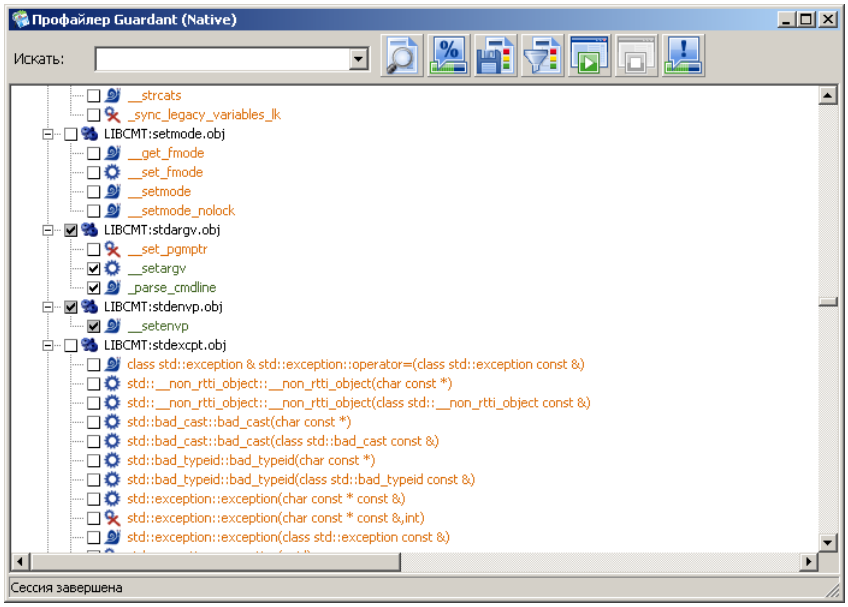
В защищаемой функции может быть от одного базового блока до нескольких тысяч, в зависимости от ее размера. Если базовых блоков в функции не найдено, защите она не подлежит.

Приняты следующие обозначения функций защищаемого приложения:

SUB_00	Функция не содержит ни одного базового блока подходящего для защиты
SUB_00	Есть базовые блоки для защиты, но один или более из них содержат циклы и рекурсии
SUB_00	Есть базовые блоки для защиты, нет циклов и рекурсий

Отметка на белом фоне означает, что все базовые блоки в данной функции выбраны и будут защищены. **Отметка на сером фоне** означает, что выбрана только часть базовых блоков.

Если на предыдущем шаге, помимо статического анализа, проводилось профилирование приложения (**шаг 2Б**), названия функций будут дополнительно окрашены различными цветами:



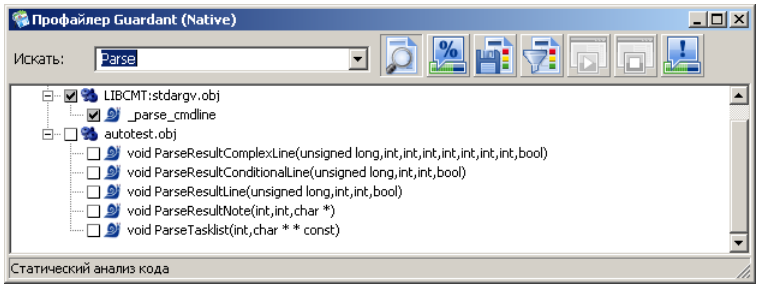
Зеленым цветом отмечаются функции, которые вызывались в ходе работы с приложением. В зависимости от скорости их работы и частоты вызова профайлер автоматически предложит, защищать их или нет.

Оранжевым цветом отмечены функции, которые ни разу не вызывались в ходе работы. При необходимости их можно отметить для защиты вручную.

Далее необходимо уточнить множество функций, подлежащих защите. При выборе функций рекомендуется

- Защищать функции, содержащие интеллектуальную собственность
- Защищать функции, изменившиеся в данном релизе
- Не защищать стандартные библиотечные функции, прилинкованные статически
- Не защищать «медленные» базовые блоки в функциях, отмеченных символом улитки

Если после защиты приложение работает медленно, необходимо найти функции вызывающие проблемы с производительностью (к примеру, при помощи **строки поиска**), и изменить параметры их защиты.



Примечание

После перекомпиляции приложения существующий **файл описания защиты кода** становится недействительным, и необходимо повторное проведение статического/динамического анализа и выбор функций.

Шаг 4. Сохранение файла описания защиты

Как только выбор функций для защиты будет закончен, **перед закрытием** окна профайлера, необходимо **сохранить файл описания защиты**:

Контрольные испытания

После окончания процесса защиты исполняемого файла, до передачи его конечному пользователю, рекомендуется провести ряд контрольных испытаний защищенного приложения, в ходе которых проверить:

- Корректность работы приложения в целом
- Корректность работы защищенного функционала приложения
- Устойчивость работы в различных средах и при различных нагрузках
- Производительность приложения в критичных, в смысле требований к производительности, местах

После успешного проведения контрольных испытаний будет получен готовый для передачи конечному пользователю программный продукт, защищенный на лицензию, находящуюся в ключе конечного пользователя.

Заключение

В данном уроке был рассмотрен порядок установки автозащиты с использованием технологии профилирования, которая позволяет выбрать функции, подлежащие защите.

Профилирование призвано разрешить сразу несколько проблем, возникающих при защите программного обеспечения автоматизированными средствами:

- Снижение производительности приложения в результате установки автозащиты
- Неопределенность покрытия кода автозащитой (позволяет гарантированно защитить участки кода, представляющие интеллектуальную собственность разработчика)

Дополнительные источники информации

При возникновении вопросов, на которые вам не удалось найти ответа в этом пособии, рекомендуем обратиться к следующим дополнительным источникам информации:

WWW: http://www.guardant.ru

Web-сайт разработчика содержит большой объем справочной информации об электронных ключах Guardant.

Служба технической поддержки:

e-mail: hotline@guardant.ru

тел. +7(495)925-77-90