

Guardant®

Система защиты от компьютерного пиратства

Профайлеры автозащиты Guardant

Методическое пособие

Содержание

Профилерование Native-приложений 3

Методы выбора функций для защиты	3
1. Не защищать функции.....	3
2. Автоматически.....	4
3. Вручную	4
4. На основе профилерования	6
Методика работы с Native-профайлером.....	8
Принципы выбора защищаемых функций	9
Примеры использования.....	9
1. Не защищать функции.....	9
2. Выбрать функции для защиты вручную.....	9
3. Перепрофилеровать приложение.....	10

Профилерование .NET-приложений..... 11

Методы выбора функций для защиты	11
1. Не защищать функции.....	11
2. Автоматически.....	12
3. Вручную	12
4. На основе профилерования	13
Принципы выбора функций	15
Автоматический выбор функций.....	15
Ручной выбор функций.....	16

Профилирование Native-приложений

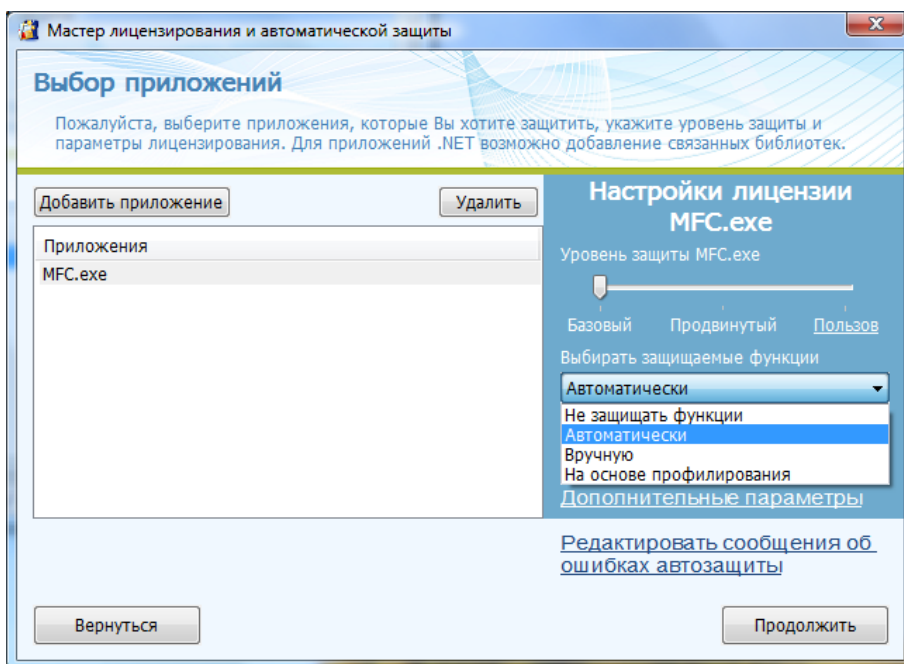
Одной из наиболее перспективных технологий автоматической защиты исполняемых Native-файлов является обработка инструкций, извлеченных из тела приложения (RIP CODE). При помощи мини-виртуальной машины некоторые наборы инструкций в защищаемом приложении преобразуются определенным образом. Это позволяет защитить функции приложения от изучения и анализа, а также значительно затрудняет создание автоматических инструментов снятия автозащиты.

Однако при использовании RIP CODE следует учитывать, что работа защищенного приложения может замедляться, вследствие того, что инструкции виртуализируются. Чтобы подобрать оптимальные функции для защиты применяется технология профилирования. С помощью специальных инструментов приложение анализируется (как статически, при помощи дизассемблера, так и динамически, в процессе исполнения), после чего создается конфигурационный файл, содержащий информацию о функциях, подлежащих защите.

Рассмотрим процесс защиты и профилирования тестового приложения, созданного в Visual Studio 2010 по шаблону **MFC Application**.

Методы выбора функций для защиты

В мастере лицензирования предусмотрены различные методы выбора защищаемых функций (т. е. тех функций, с которыми будет работать RIP CODE):



1. Не защищать функции

При выборе этой опции происходит отключение технологии RIP CODE. Соответственно, этот вариант следует выбирать лишь тогда, когда RIP CODE вызывает проблемы с работой защищенного приложения.

2. Автоматически

Данный вариант предлагается по умолчанию. В этом случае RIP CODE будет выбирать функции для защиты случайным образом, но в целях оптимизации производительности исключит все функции с циклами и рекурсиями.

Этот вариант рекомендуется использовать в случаях, когда защищенное приложение не вызывает проблем с производительностью.

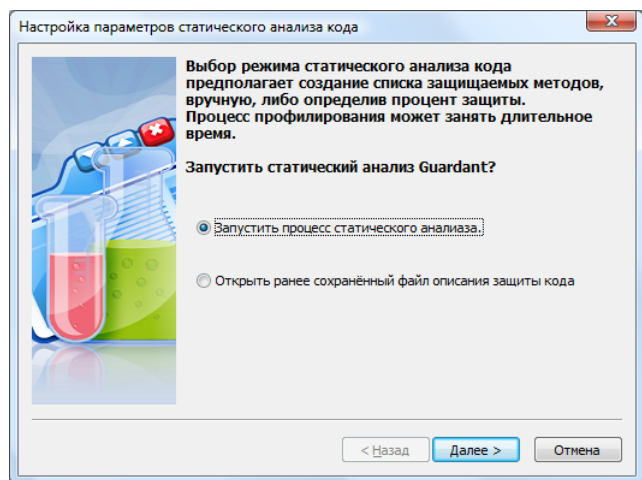
Процент функций, защищенных технологией RIP CODE, зависит от положения ползунка **Уровень защиты**, и показывается во всплывающей подсказке. На **Пользовательском** уровне защиты процент можно регулировать вручную.

3. Вручную

Можно выбрать функции для защиты вручную. Это рекомендуется делать, если защищенное приложение демонстрирует невысокую производительность (по сравнению с исходной), и разработчик хорошо знаком с внутренней структурой приложения, а значит, сможет указать автозащите, какие функции лучше не защищать.

При выборе данного метода запускается процесс статического анализа кода.

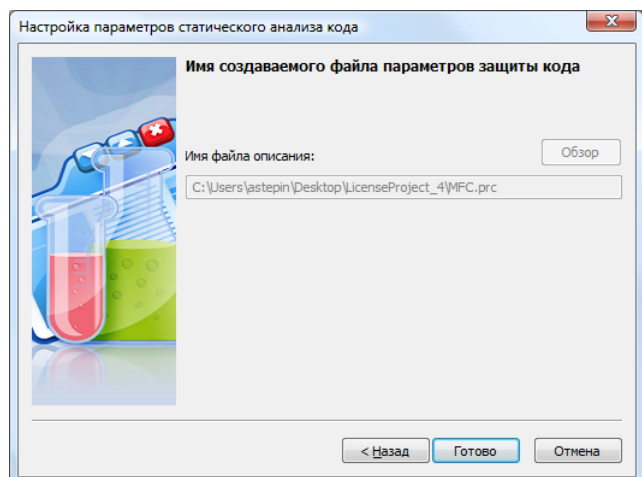
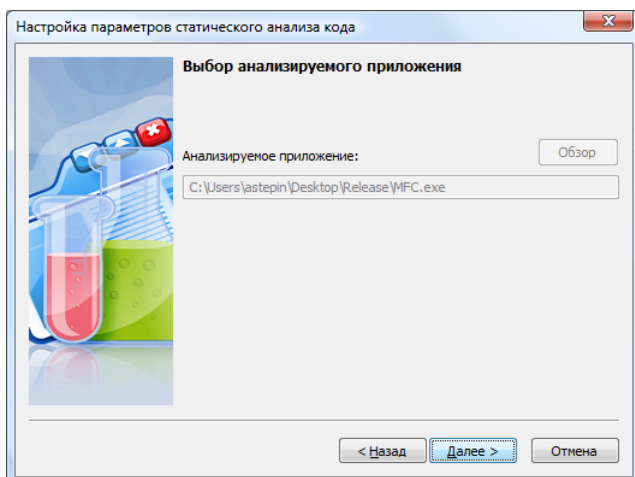
Если статический анализ уже проводился, и нужно лишь подкорректировать выбор защищаемых функций, то следует выбрать вариант **Открыть ранее сохраненный файл описания защиты кода**.



ВАЖНО!

При повторной компиляции приложения, и, соответственно, регенерации MAP-файла, существующий файл описания защиты кода становится недействительным, и необходимо повторное проведение статического анализа и повторный выбор функций!

Если же процесс статического анализа проводится впервые (или защищаемый файл изменился), то необходимо выбрать пункт **Запустить процесс статического анализа**, предварительно убедившись, что в одной папке с защищаемым файлом находится соответствующий ему MAP-файл:



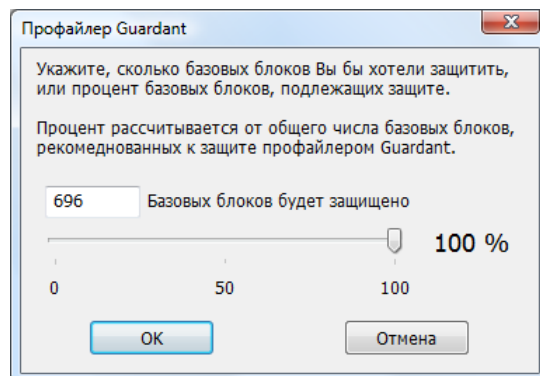
Когда профилирование запускается из среды мастера лицензирования, то изменение параметров в данном диалоговом окне будет недоступно. В случае, если мастер лицензирования не используется, то здесь необходимо указать непосредственно профилируемое приложение.

Аналогично, если профайлер запускается из мастера, то имя файла описания изменить нельзя – оно жестко привязано к проекту лицензии. В случае использования профайлера отдельно, здесь указывается имя выходного файла с настройками для опции **RIP CODE**.

По нажатию на кнопку **Готово** начинается процесс анализа и дизассемблирования защищаемого приложения. После этого в отдельном окне выводятся все базовые блоки, которые можно защитить.

Здесь можно указать первоначальный процент защищаемых базовых блоков.

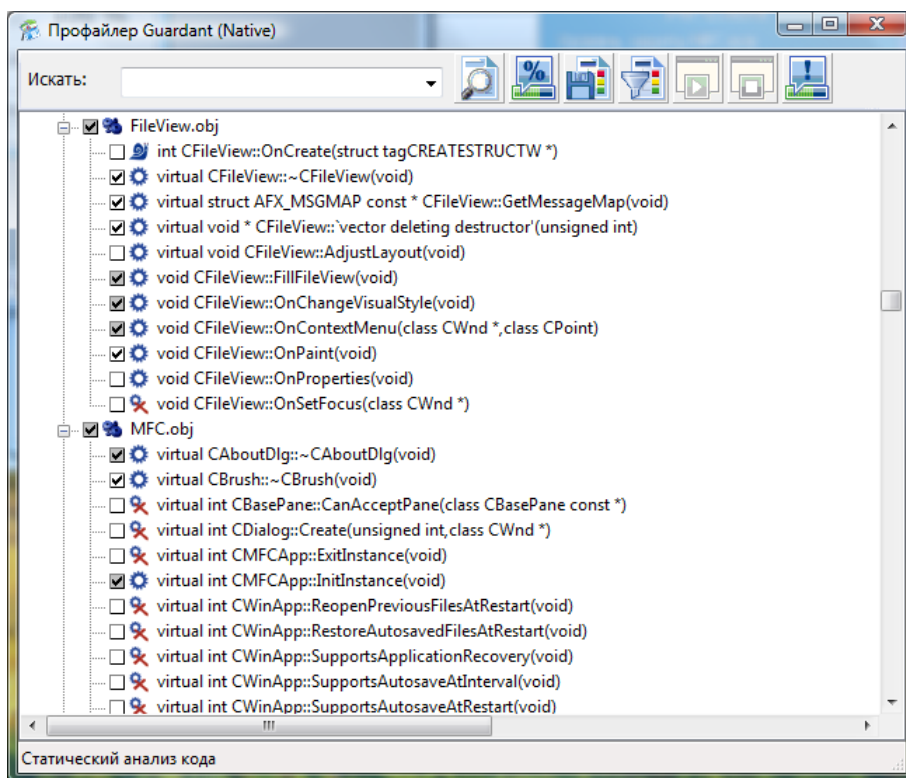
Чем меньше процент, тем быстрее будет работать защищенное приложение, но тем меньше оно будет защищено.



Примечание

При помощи RIP CODE виртуализируется не тело функции целиком, а только определенные наборы инструкций. Каждый такой набор называется базовым блоком. 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.

По нажатию на **OK** происходит переход в основное окно работы профайлера, методика работы с которым описана далее:

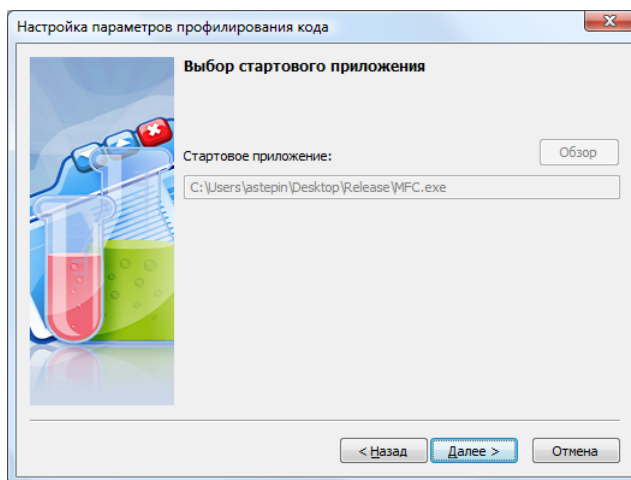
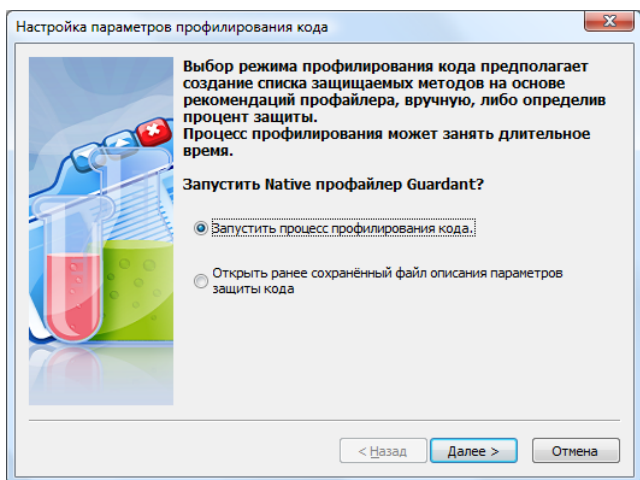


4. На основе профилирования

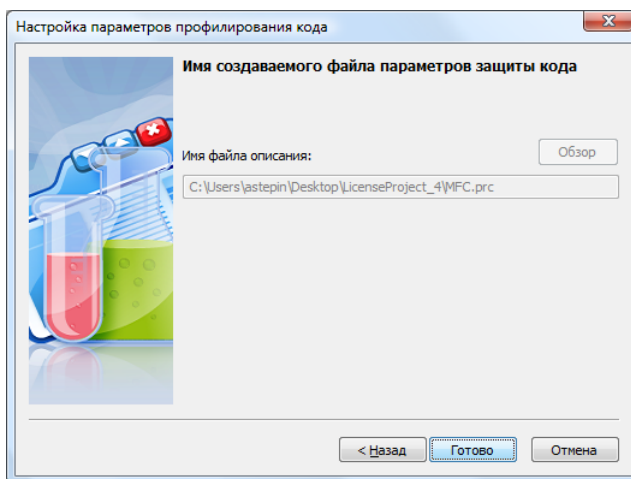
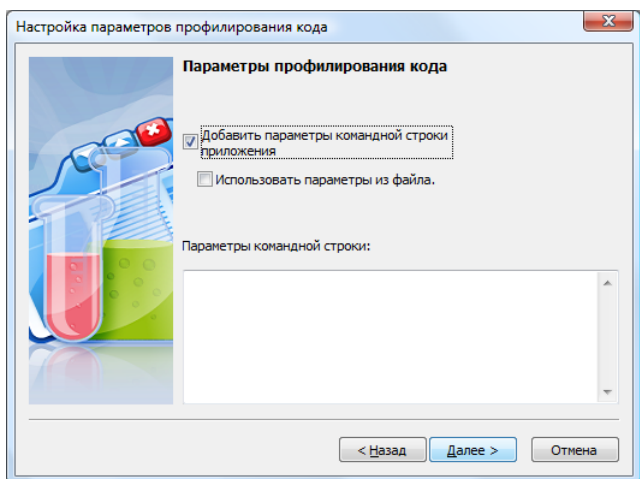
Выбор функций для защиты на основе профилирования – это наиболее сбалансированный вариант работы. С одной стороны, он обеспечивает высокую производительность защищенного приложения, с другой – не требует от разработчика досконального знания структуры приложения, а также особенностей работы компилятора.

При профилировании происходит фактический запуск защищаемого приложения. Разработчику предлагается некоторое время поработать с приложением, используя наиболее востребованные функции и сценарии. При этом профайлер 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.

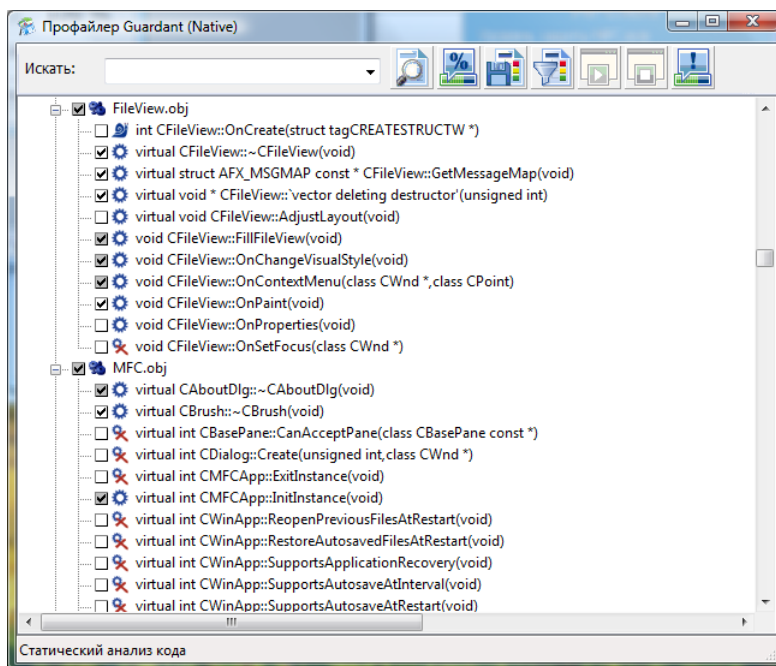
Аналогично статическому анализу, нужно начать процесс профилирования и выбрать профилируемое приложение (если вызов профайлера осуществляется из мастера лицензирования, то приложение задается автоматически):



Затем можно указать параметры командной строки, с которыми вызывается приложение, и задать файл для сохранения результатов работы профайлера:

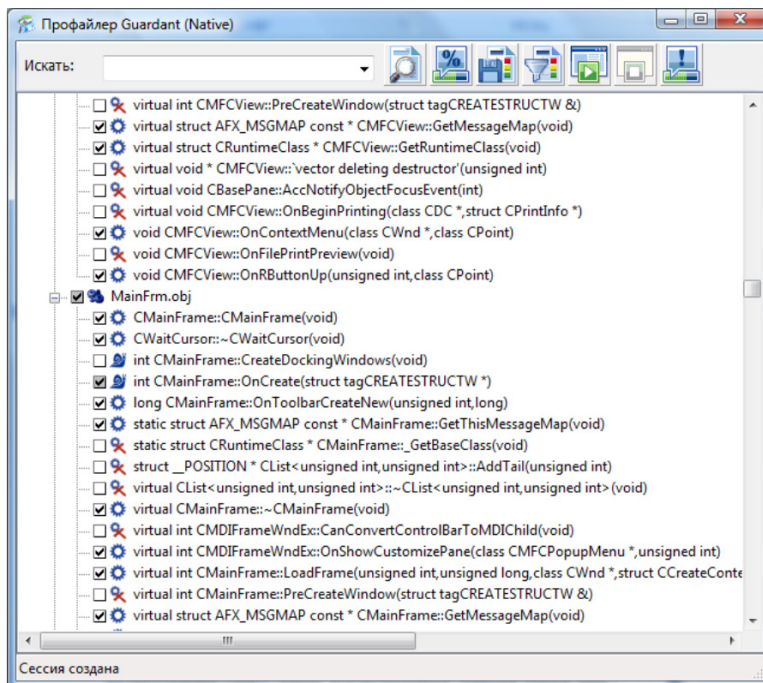


По нажатию на кнопку **Готово** начнется статический анализ приложения (как и в варианте **Вручную**), и будет предложен первоначальный набор функций для защиты:



Процесс профилирования запускается и останавливается нажатием на соответствующие кнопки в окне профайлера. После запуска приложения необходимо поработать с ним какое-то время, затем закрыть его любым способом.

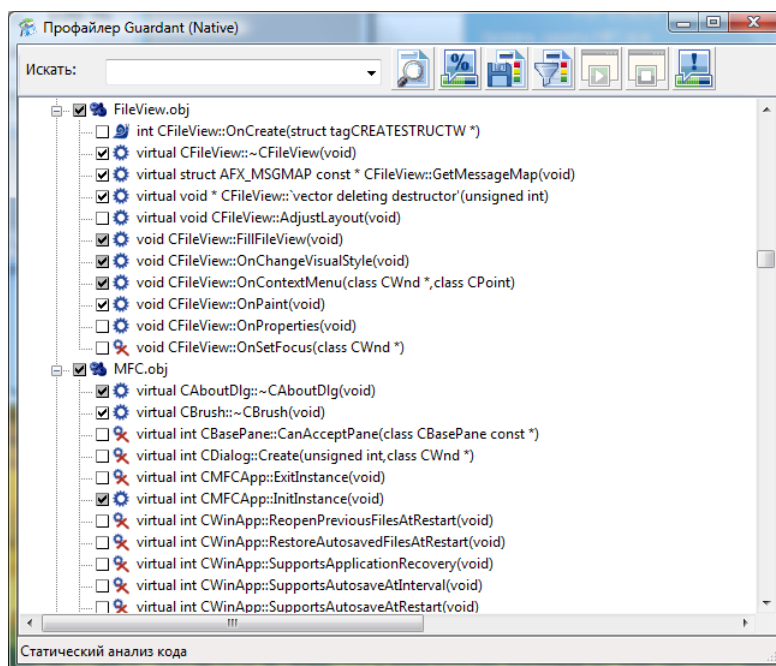
По результатам анализа повторно выдается окно профайлера со списком функций и отметками, какие из базовых блоков будут защищаться, сделанными на основе измерения скорости работы функций:






Условные обозначения	Пояснение
Оранжевый цвет шрифта	Функция ни разу не вызывалась профайлером
Зеленый цвет с галочкой	Функция вызывалась и выбрана для защиты
Зеленый цвет без галочки	Функция вызывалась, но не выбрана для защиты

Методика работы с Native-профайлером

Главное окно профайлера предназначено для выбора функций, базовые блоки в которых будут защищаться при помощи RIP CODE:



Любая функция после статического анализа находится в одном из трех состояний:

-  SUB_00 ни одного базового блока подходящего для защиты не найдено
-  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

Галочка на белом фоне означает, что все базовые блоки в данной функции выбраны и будут защищены.

Галочка на сером фоне означает, что выбрана только часть базовых блоков.

Выбор базовых блоков происходит автоматически при установке процента и может быть изменен вручную.

В любой момент можно:

- Сохранить текущие результаты профилирования и выйти из приложения
- Найти функцию в приложении по ее названию или его части
- Отфильтровать для показа только защищенные функции
- Вызвать меню настроек профайлера
- Запустить повторно процесс профилирования

Принципы выбора защищаемых функций

Общая рекомендация: если приложение после защиты работает медленно, необходимо найти функции, вызывающие проблемы с производительностью, и изменить параметры их защиты.

Кроме того, нужно:

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

Примеры использования

Пусть после защиты приложения обнаружилось, что часть функционала стала работать значительно медленнее.

Например:

- Смена шрифтов интерфейса занимает более 10 секунд, хотя до защиты она выполнялась за 1 - 2 секунды
- При создании новой папки в некоторых случаях приложение зависает или аварийно завершается
- В приложении окно состоит из нескольких панелей, и при изменении их размера оно надолго подвисает

Данные проблемы можно решить следующими методами:

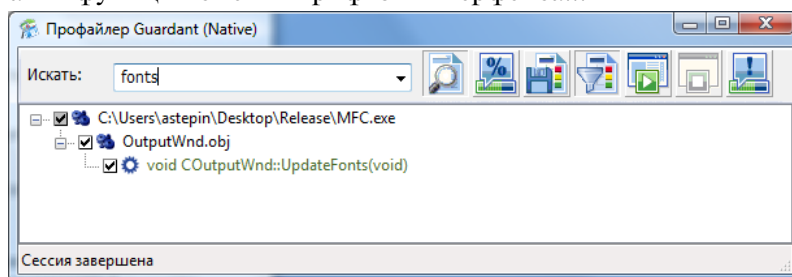
1. Не защищать функции

Отключить RIP CODE, т.е. в главном окне мастера лицензирования выбрать пункт **Не защищать функции**. Данный метод наиболее радикален, он помогает преодолеть большинство проблем с производительностью, но снижает защищенность приложения.

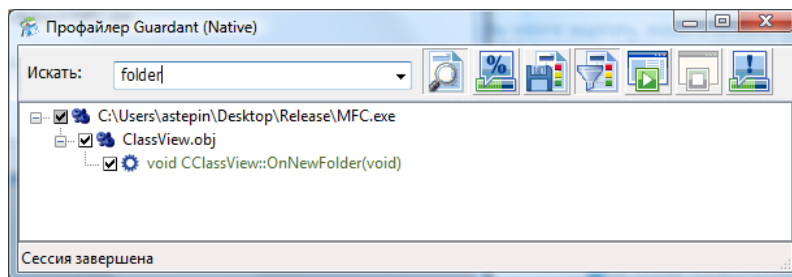
Рекомендуется использовать, если в приложение встроена защита на Guardant API, или когда любые попытки использования RIP CODE вызывают проблемы.

2. Выбрать функции для защиты вручную

С помощью окна поиска найти функции смены шрифтов интерфейса..:



..и создания новой папки:



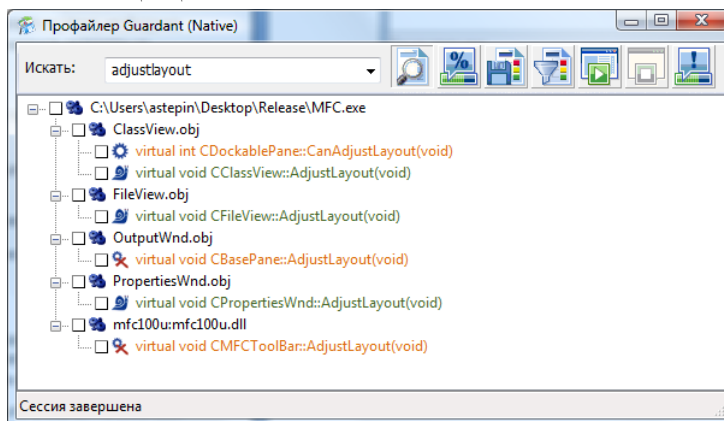
Снять с них отметки, т.е. не защищать эти функции. Проблемы с производительностью и зависаниями должны исчезнуть.

3. Перепрофилировать приложение

Наиболее простой для разработчика вариант решения проблем с низкой производительностью, это повторное профилирование приложений.

Необходимо при профилировании приложения сделать упор на использование именно тех функций, с которыми наблюдается проблема.

Например, при перепрофилировании тестового приложения несколько раз перемещали его панели в главном окне. По результатам профайлер обнаружил медлительность данной функции и более не предлагает ее защищать:



Профилерование .NET-приложений

Основным механизмом защиты .NET-приложений является шифрование тел защищаемых функций и их динамическое расшифрование во время работы приложения. При совместном использовании символьного обфускатора и шифратора строковых констант на электронном ключе защита становится беспрецедентно стойкой.

Однако в ряде случаев за это приходится платить снижением скорости работы:

- Если функция зашифрована, то при ее вызове на расшифрование тратится определенное время
- Даже если функция уже расшифрована, она, тем не менее, вызывается с использованием защитных механизмов, что может вносить задержку в ее выполнении на доли секунды

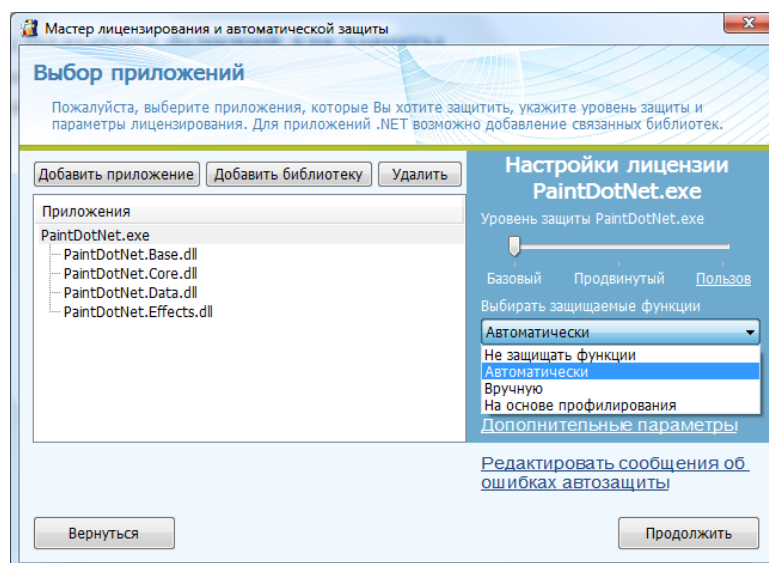
В среднестатистическом случае увеличение времени запуска приложения и понижение производительности его работы на несколько процентов не вызывает особых проблем. Однако если функция в процессе работы приложения вызывается сотни и тысячи раз, то накладные расходы по работе защитных механизмов именно этой функции могут серьезно снижать производительность приложения. В таком случае, необходима технология выбора функций для защиты. Именно для этого существует .NET-профайлер.

* * *

Рассмотрим процесс защиты тестового приложения Paint.NET.

Методы выбора функций для защиты

Для любого защищаемого приложения существует четыре варианта выбора функций для защиты:



1. Не защищать функции

Это самый кардинальный вариант, при котором не используется шифратор кода (модуль **CodeProtect.exe**), и функции не шифруются и не защищаются. Использовать его следует только при непреодолимых проблемах с производительностью защищенного приложения, а также в экзотических случаях, когда необходима лишь символьная обфускация с шифрованием строк.

2. Автоматически

Функции для шифрования будут выбираться случайным образом. Если необходимо произвести защиту быстро, и производительность защищенного приложения не вызывает нареканий, то использование автоматического выбора оправданно.

В этом случае будет защищен некоторый процент от общего числа функций, зависящий от положения ползунка **Уровень защиты**. В случае с **Пользовательским** уровнем процент защиты может быть установлен самостоятельно разработчиком.

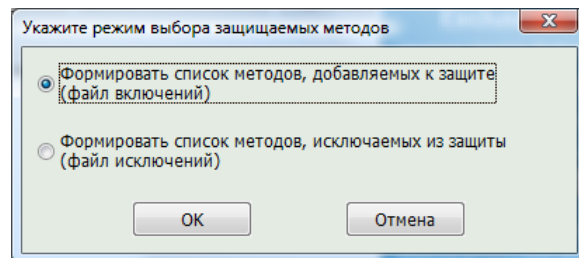
3. Вручную

Если разработчик хорошо знаком со структурой защищаемого приложения, то он может самостоятельно задать функции для защиты.

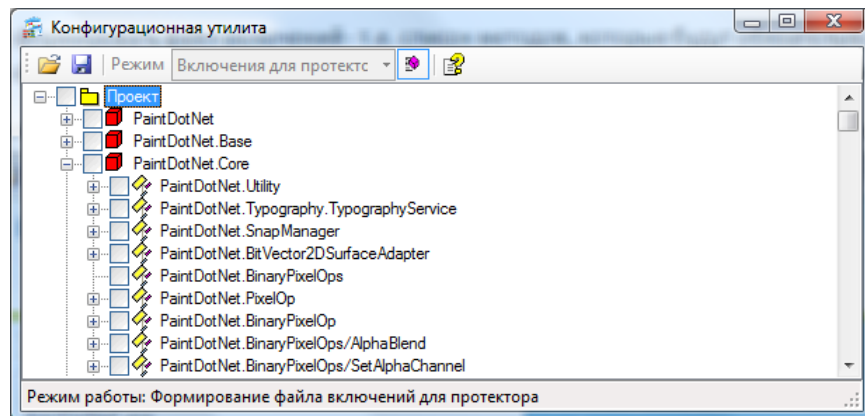
При использовании данного варианта предоставляются две возможности:

- Формирование файла включений (списка методов, которые защищаются)
- Формирование файла исключений (списка методов, которые не защищаются)

В зависимости от технологий .NET, использованных в приложении, а также от его специфики и структуры, может понадобиться сформировать как один из конфигурационных файлов, указанных выше, так и оба:



В любом случае, запускается утилита формирования конфигурационных файлов:



Во всех случаях работа с утилитой происходит одинаково:

- Галочками отмечаются методы, подлежащие защите (или исключению из нее)
- Для упрощения стоит отключить отображение т. н. accessor-методов, которые редко защищаются
- В любой момент можно сохранить результат работы и выйти из приложения
- При необходимости можно редактировать ранее созданный файл включений/исключений

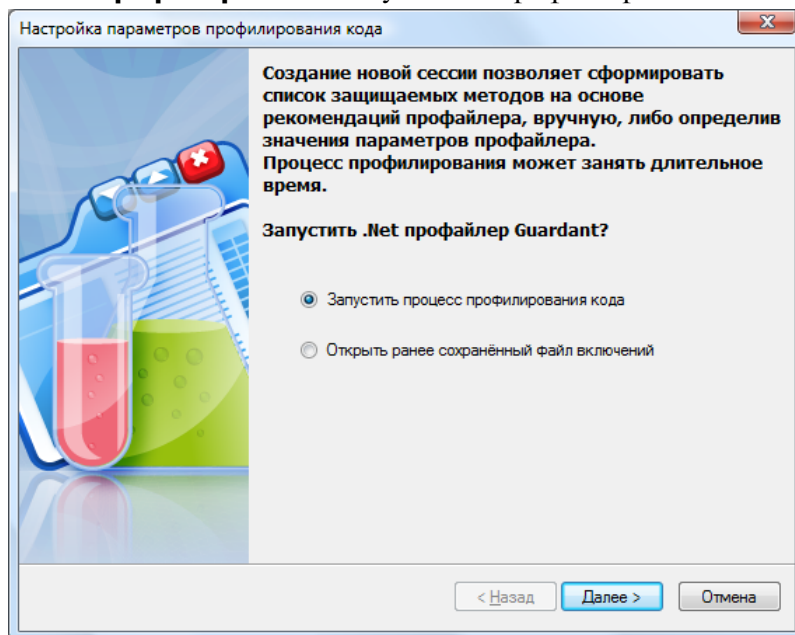
Получившийся конфигурационный файл автоматически передается мастеру лицензирования и используется для защиты.

Файлы включений и исключений применяются независимо друг от друга. При этом, если метод указан в обоих файлах, то исключение имеет приоритет.

4. На основе профилирования

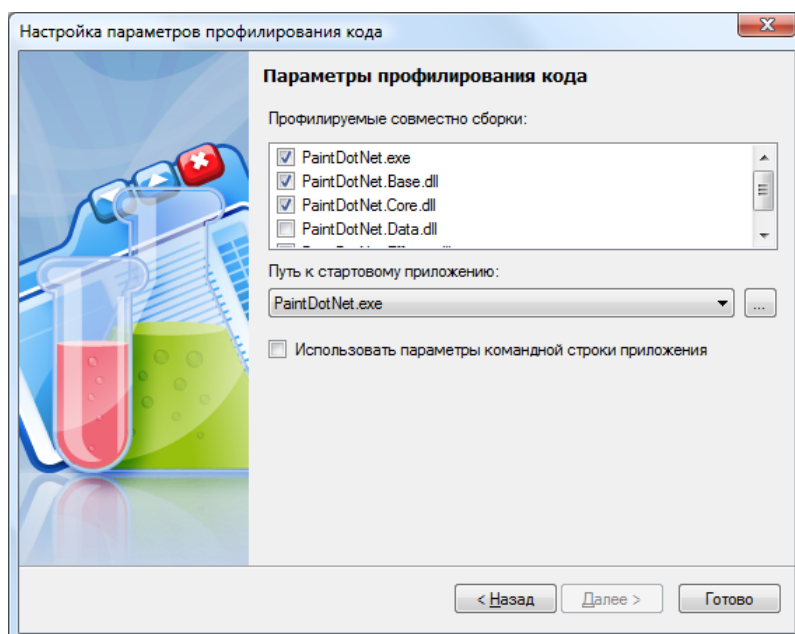
Если производительность работы защищенного приложения вызывает нарекания, можно поработать с ним с использованием профайлера и автоматически измерить время работы отдельных функций, частоту и интенсивность их вызова, и на основании полученных данных сформировать файл включений для защиты.

При выборе варианта **На основе профилирования** запускается профайлер .NET:



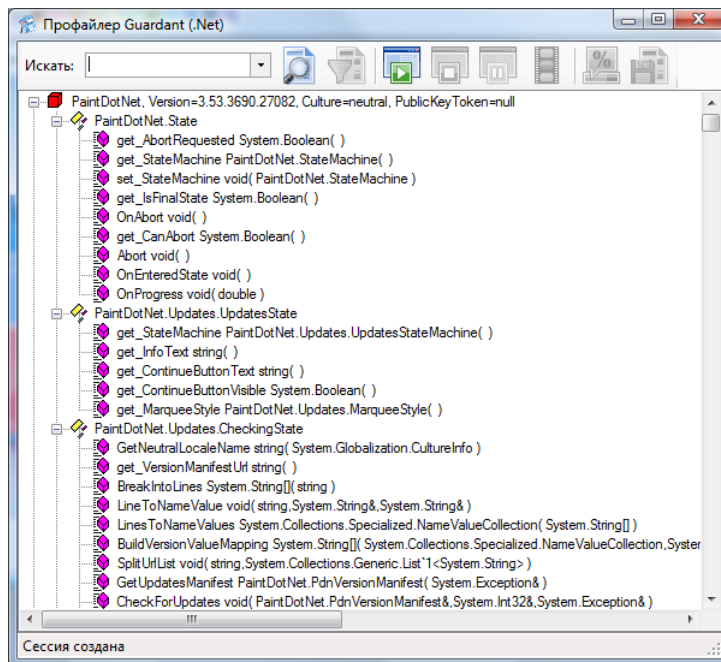
Здесь можно запустить процесс профилирования, или открыть для редактирования ранее созданный файл включений.

На следующем экране профайлера выбрать сборки для профилирования:

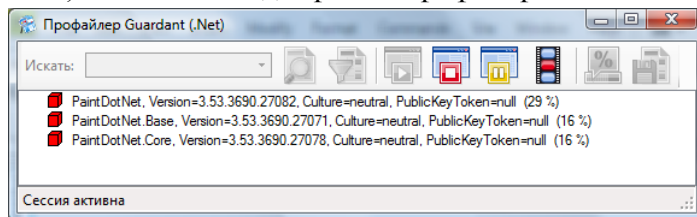


Профилировать можно любое число связанных сборок, при этом одна из них должна быть исполняемым файлом.

По нажатию на кнопку **Готово** происходит анализ всех выбранных сборок приложения и вывод основного окна профайлера:

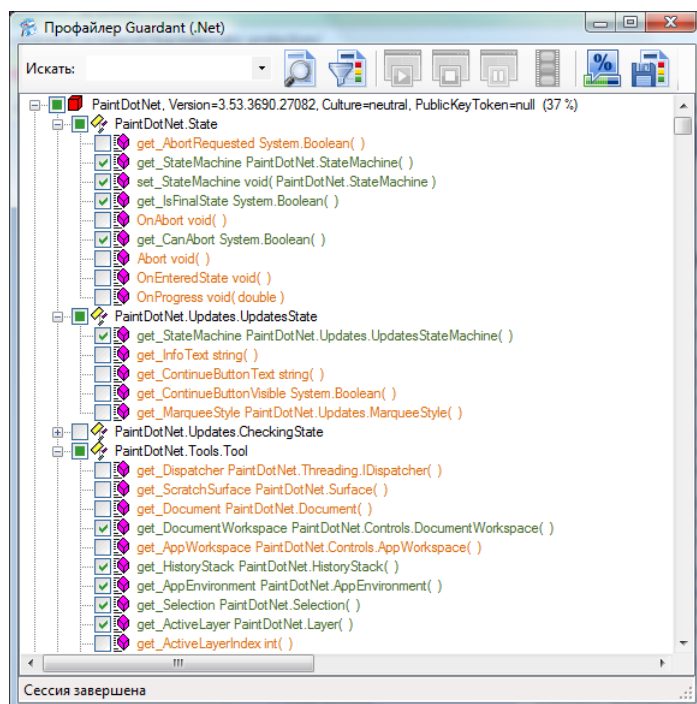


По нажатию на кнопку **Начать сессию профилирования** выбранное приложение запускается и начинается процесс измерения времени работы и частоты вызова функций. В ходе работы в главном окне профайлера отображаются данные о профилируемом приложении и связанным сборкам. Главный параметр здесь – процент уже вызванных функций. Чем ближе это число к 100%, тем больше кода прошло профилирование:



В любой момент процесс профилирования можно остановить, или получить «снимок» – информацию о функциях, которые уже проанализированы и выбраны для использования в автоматической защите.

По завершении процесса в главном окне профайлера выводится подробная информация о приложении:



Условные обозначения	Пояснение
Оранжевый цвет шрифта	Функция ни разу не вызывалась профайлером
Зеленый цвет с галочкой	Функция вызывалась и выбрана для защиты
Зеленый цвет без галочки	Функция вызывалась, но не выбрана для защиты

Принципы выбора функций

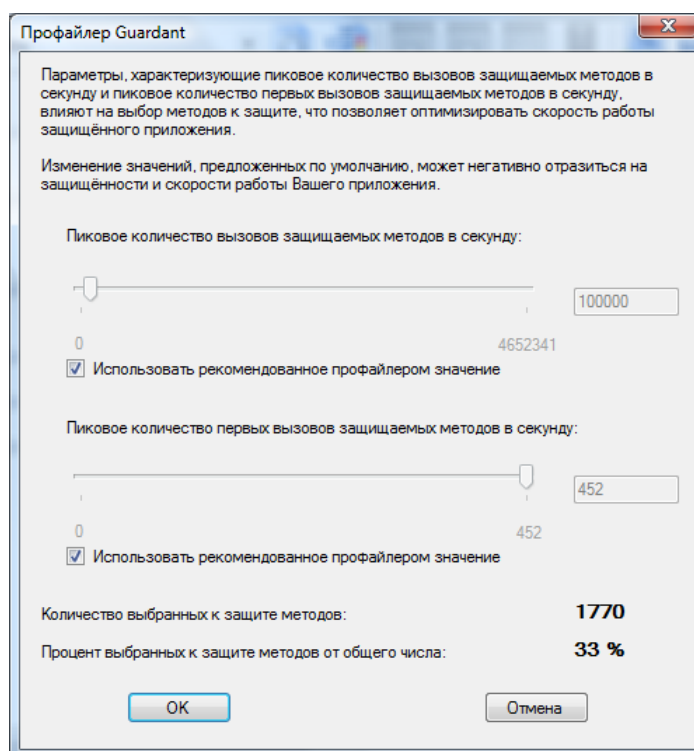
После профилирования приложения можно:

- Изменить критерии автоматического выбора функций для защиты
- Вручную изменить параметры защиты каждой функции по отдельности

Для тонкой настройки производительности приложения можно использовать оба метода, но надо учитывать, что каждый раз при автоматическом выборе все пользовательские функции сбрасываются.

Автоматический выбор функций

При нажатии на кнопку **Процент защищаемых функций** появляется следующее диалоговое окно:



Здесь необходимы небольшие пояснения.

В процессе работы защищенного приложения вызов функции может быть **первый** или **обычный**.

При первом вызове функции происходит обращение к электронному ключу и расширение тела функции, для его последующего исполнения средой .NET. При обычном вызове уже расшифрованной функции обращения к ключу не происходит, но есть небольшой расход времени на работу защитных механизмов.

В связи с этим, существуют две типичные проблемы с производительностью защищенного приложения:

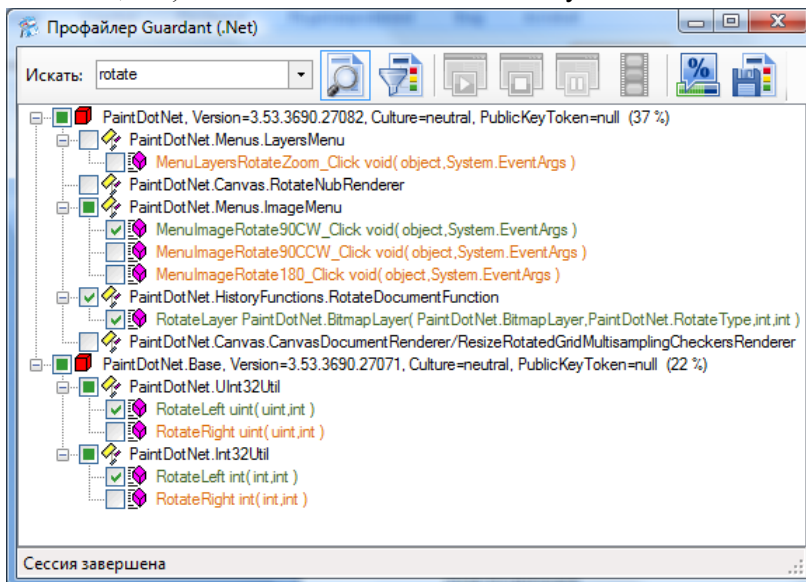
- **Обычный** вызов происходит слишком часто (десятки тысяч раз в секунду)
- Последовательно происходит много **первых** запусков функции, что особенно характерно при старте приложения

Именно эти критерии и регулируются при автоматическом выборе функций для защиты (см. скриншот выше).

Накладные расходы на вызов функции фиксированы и посчитаны с точностью до микросекунд, поэтому рекомендованные профайлером значения отлично подходят для большинства ситуаций.

Ручной выбор функций

Здесь все довольно просто. Если работа какой то конкретной функции вызывает проблемы после применения автозащиты, ее легко найти и снять отметку:



Например, на скриншоте выше показаны результаты поиска по слову **rotate**. Если после автозащиты с параметрами по умолчанию было замечено, что в Paint.NET функция **Повернуть изображение на 90 градусов** стала работать гораздо медленнее, то легко найти ее и снять отметку **Защитить**.