Изучение программного продукта может осуществляться методом статического и/или динамического анализа. При статическом анализе разработка алгоритма взлома защиты производится на основе анализа результатов дизассемблирования или декомпиляции взламываемой программы. Динамический анализ чаще всего применяют при взломе программных продуктов, исполняемый код которых зашифрован или динамически изменяется, так как статический анализ подобных программ сопряжен с определенными трудностями.  
  
Для динамического анализа взламываемую программу запускают в среде отладчика, при этом становится возможным контроль всех изменений, возникающих в процессе функционирования приложения. В процессе динамического анализа, используя режим отладки, производят пошаговое прохождение «защитных» алгоритмов программы, в частности алгоритмов проверки и генерации корректного регистрационного кода. Еще одним средством, применяемым при динамическом анализе, являются мониторы активности, отслеживающие обращение взламываемой программы к файлам, системным сервисам, портам и внешним устройствам.  
  
Основным инструментом, применяемым для защиты приложений от взлома, являются программы-протекторы. Защита, реализуемая большинством протекторов, заключается в том, что они упаковывают и/или шифруют исходный исполняемый файл, уделяя основное внимание защите процедуры распаковки/расшифровки файла.  
  
Подобный алгоритм работы протекторов очень часто оказывается причиной недостаточной степени защиты обеспечиваемой большинством из них. В случае если приложение защищено с использованием упаковки, по окончании работы распаковщика в распоряжении злоумышленника может оказаться исходный файл, который может быть получен при дампе определенной области памяти. Более того, для борьбы с самыми распространенными протекторами взломщики разработали множество программных инструментов, позволяющих взламывать защиту в автоматическом режиме. Аналогичный подход применяется при борьбе с шифрованием: после получения лицензионного ключа, который в том числе может быть легально приобретен, взломщик сможет расшифровать защищенные участки кода.  
  
Ряд программ-протекторов применяют различные антиотладочные приемы. Однако применение антиотладки значительно снижает быстродействие программы. Также следует помнить, что антиотладочные приемы действенны лишь против динамического анализа и совершенно не эффективны против статического. Более того, все антиотладочные приемы, применяемые в современных протекторах, уже давно изучены, а взломщиками написано множество утилит, которые их полностью нейтрализуют. На эффективность использования мониторов активности встроенные в приложение средства защиты от отладки не влияют.  
  
Более эффективными методами защиты приложения являются обфускация и виртуализация, усложняющие анализ кода защищаемого приложения. В общем случае эффективность данных методов достигается за счет использования особенностей человеческого фактора — чем сложнее исходный код, чем больше ресурсов использует приложение, тем человеку его анализирующему тяжелее понять логику работы программы, а следовательно, и взломать примененные средства защиты.  
  
При обфускации производится запутывание кода приложения за счет введения дополнительных инструкций. При виртуализации исходный код преобразуется в байт-код, выполнение которого осуществляется на специальном интерпретаторе, имитирующем виртуальную машину со специфической системой команд. Следовательно, применение виртуализации приводит к высокой и неснижаемой степени запутанности результирующего кода, а при определенном подходе к реализации этого метода защищенный код не содержит методов восстановления оригинального кода в явном виде. Таким образом, важнейшим преимуществом виртуализации является то, что в момент выполнения виртуализированного участка кода не происходит его обратного преобразования в машинные коды процессора, а это исключает возможность получения взломщиком оригинального кода приложения.  
  
Задача обратного инжиниринга виртуализированных фрагментов сводится к изучению архитектуры виртуальной машины, созданию дизассемблера, соответствующего архитектуре имитируемого виртуальной машиной процессора, и анализу дизассемблированного кода. При определенном подходе к реализации виртуальной машины задача создания дизассемблера виртуализированного кода становится достаточно трудоемкой. Единственным недостатком виртуализации является сравнительно низкая скорость работы, поэтому этот метод следует применять только для защиты участков кода, некритичных к скорости исполнения.  
  
В подавляющем большинстве современных протекторов методы обфускации и виртуализации играют второстепенную роль, а уровень их реализации недостаточен, что позволяет взломщикам выполнять снятие подобной защиты в автоматизированном или ручном режиме. Еще одним слабым местом многих современных протекторов является использование недокументированных функций Windows, что создает ограничения для использования приложения в новых версиях операционной системы или при включенном режиме DEP.

Что такое VMProtect?

Программа VMProtect относится к новому поколению средств защиты программного обеспечения. VMProtect поддерживает компиляторы Delphi, Borland C Builder, Visual C/C++, Visual Basic (native), Virtual Pascal, при этом VMProtect содержит встроенный дизассемблер, позволяющий работать с файлами форматов EXE, DLL, BPL, OCX, SYS и подключать MAP-файл, создаваемый компилятором, для быстрого выбора участков кода, которые следует защитить. Для автоматизации операций по защите приложения в VMProtect реализован встроенный скриптовый язык. Программа VMProtect обладает полной поддержкой всех 32/64-разрядных операционных систем семейства Windows: Windows 95/98/ME, Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Vista.  
  
Базовым принципом, на основе которого построен VMProtect, является обеспечение эффективной защиты кода приложения от изучения, так как именно максимальное усложнение понимания логики работы внутренних механизмов защиты приложения создает максимальные трудности при взломе программы. Основными методами защиты программного кода, применяемыми VMProtect, являются виртуализация, мутация и смешанный метод защиты, сочетающий мутацию кода приложения с его последующей виртуализацией.  
  
Одним из достоинств реализации метода виртуализации в программе VMProtect является то, что виртуальная машина, на которой выполняются виртуализированные фрагменты кода, встраивается в результирующий код защищаемого приложения. Следовательно, для функционирования приложения, защищенного с помощью VMProtect, нет необходимости использовать какие-либо дополнительные библиотеки или модули. VMProtect позволяет использовать несколько отличных друг от друга виртуальных машин для защиты разных участков кода одного приложения, что еще больше усложняет процесс взлома защиты, так как взломщику будет необходимо анализировать архитектуру уже нескольких виртуальных машин.  
  
Метод мутации кода приложения, реализованный в VMProtect, основан на обфускации, в процессе которой в код приложения добавляются «мусорные» команды, «мертвый» код, случайные условные переходы, выполняется мутация оригинальных команд, а также выполнение ряда операций переносится в стек.  
  
Ключевым отличием программы VMProtect от других протекторов является то, что с ее помощью можно защитить различные участки кода разными методами: часть кода виртуализировать, часть обфусцировать, а для самых критичных участков применить смешанный метод защиты.  
  
Еще одной уникальной возможностью программы VMProtect является включение в код приложения водяных знаков, позволяющих однозначно идентифицировать официального владельца взломанного экземпляра программы, а следовательно, принять к нему соответствующие меры.  
  
Программа VMProtect доступна в 4 версиях:

* Ultimate (имеет систему лицензирования);
* Lite (облегченная версия);
* Professional (базовая версия);
* SenseLock Edition (возможна привязка кода к аппаратными ключами SenseLock).

Отличия функциональных возможностей разных версий VMProtect приведены в таблице:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Возможности** | **Версия VMProtect** | | | | |
| Lite | Professional | SenseLock Edition | Ultimate |  |
| **Методы защиты** | | | | | |
| Мутация | + | + | + | + |  |
| Виртуализация | + | + | + | + |  |
| Ультра (мутация+виртуализация) | + | + | + | + |  |
| **Виды интерфейса** | | | | | |
| Режим «Мастер» | + | + | + | + |  |
| Режим «Эксперт» | — | + | + | + |  |
| Консольная версия | — | + | + | + |  |
| **Опции защиты** | | | | | |
| Защита памяти | + | + | + | + |  |
| Защита импорта | + | + | + | + |  |
| Упаковка | + | + | + | + |  |
| Определение отладчика | + | + | + | + |  |
| Определение средств виртуализации | + | + | + | + |  |
| **Сервисные функции** | | | | | |
| Водяные знаки | — | + | + | + |  |
| Скриптовый язык | — | + | + | + |  |
| Система лицензирования | — | — | + | + |  |
| Система администрирования ключей SenseLock | — | — | + | — |  |
| Менеджер лицензий | — | — | — | + |  |
| DLL Box | — | — | — | + |  |