МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Гжельский государственный университет»** (ГГУ)

Колледж ГГУ

Специальность 09.02.07 Информационные системы и программирование

**Реферат**

по предмету «Моделирование и анализ ПО»

на тему «Обратное проектирование алгоритма»

ВЫПОЛНИЛ:

Студент группы ИСП-О-17

Скрябин С.И.

ПРОВЕРИЛА:

Прокуронова А. Ю.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

п. Электроизолятор

1. год

В современной конкурентной среде традиционный жизненный цикл изделия от разработки до производства часто не устраивает промышленные компании. В тех случаях, когда необходимо разработать и произвести изделие на основе существующего прототипа без готовой конструкторской документации и прочих исходных данных, тогда на помощь производству приходит обратное проектирование - создание виртуальной 3D-модели изделия и воссоздание конструкторской документации.

Основными угрозами для программного продукта, защищённого от несанкционированного использования, являются угроза нарушения функциональности модуля защиты и угроза раскрытия эталонных характеристик среды.

В любом случае, существует ряд задач, которые злоумышленник должен решить при реализации данных угроз.

Задача локализации кода защитного механизма в коде программы.

Следует отметить, что для современного ПО без использования специализированных программных средств эта задача злоумышленником не может быть решена за приемлемое время, что обусловлено следующими причинами.

* код защитного механизма занимает сравнительно малый объем в общем объеме кода программы. Задача ручного поиска блоков установки, сравнения характеристик среды, и ответной реакции, занимающих размер 100-200 байт в коде программы, занимающем сотни мегабайт, без использования специализированных средств не может быть решена за приемлемое для злоумышленника время.
* анализ кода программы злоумышленником в значительной степени затрудняется тем, что производится анализ не исходного текста на языке высокого уровня, а машинного кода, сформированного компилятором. На его разборку и понимание уходит значительное время даже у специалистов высокого класса. Также, при анализе машинного кода исследователю приходится увязывать в единую последовательность действий как минимум 10-20 команд, чтобы понять скрываемый за ними результат. Это усложняет анализ программы.

Злоумышленник должен понять, каким образом построена защита, где она хранит (если хранит) эталонные характеристики среды, где сохраняет (если сохраняет) свои метки и ключи, на каком этапе принимается решение о регистрации программы либо об отклонении регистрации. При этом злоумышленник сталкивается с проблемой анализа машинного кода, что приводит к аналогичным трудностям, представленным выше.

Однако необходимо отметить, что на самом деле сложности, перечисленные выше, для подготовленного злоумышленника в большинстве случаев таковыми не являются. Существует множество программных продуктов, облегчающих ему решение данных задач. Их реализация, в отдельных случаях доводится до автоматизма.

Одна из основных задач, решаемая злоумышленником при исследовании ПО – анализ логики работы программы, поиск в ней участков кода, отвечающих за реализацию защитных механизмов, детальное исследование принципов работы модулей защиты. При этом ставится задача представления машинного кода на как можно более высоком уровне абстракции с целью упрощения его понимания.

**Под обратным проектированием** (reverse engineering) понимают процесс исследования и анализа машинного кода, нацеленный на понимание общих механизмов функционирования программы, а также на его перевод на более высокий уровень абстракции (более высокий уровень языка программирования) вплоть до восстановления текста программы на исходном языке программирования.

Основными методами обратного проектирования являются отладка и дизассемблирование программ. При этом используются следующие средства (инструменты).

***Отладчики*** – программные средства, позволяющие выполнять программу в пошаговом режиме, контролировать ее выполнение, вносить изменения в ход выполнения. Данные средства позволяют проследить весь механизм работы программы и являются средствами динамического исследования работы программ. В качестве примера одного из наиболее мощных отладочных средств можно привести отладчик SoftIce.

***Дизассемблеры*** – программные средства, позволяющие получить листинг программы на языке ассемблера, с целью его дальнейшего статического изучения. Дизассемблеры являются средствами статического исследования.

По способу реализации интерфейса взаимодействия с пользователем, существующие дизассемблеры можно разделить на две категории – автономные и интерактивные.

Автономные дизассемблеры требуют от пользователя задания всех указаний до начала дизассемблирования и не позволяют вмешиваться непосредственно в сам процесс. Если же конечный результат окажется неудовлетворительным, пользователь либо вручную правит полученный листинг, либо указывает дизассемблеру на его ошибки и повторяет всю процедуру вновь и вновь, порой десятки раз. Такой способ общения человека с дизассемблером непроизводителен и неудобен, но его легче запрограммировать.

Интерактивные дизассемблеры обладают развитым пользовательским интерфейсом, благодаря которому приобретают значительную гибкость, позволяя человеку «вручную» управлять разбором программы, помогая автоматическому анализатору там, где ему самому не справиться – отличать адреса от констант, определять границы инструкций и т.д.

Примером автономного дизассемблера является SOURCER, а интерактивного – **IDA Pro.**