Н. Ю. Кульман, А. Г. Полянин

РЕВЕРСИНГ И ЗАЩИТА ПРОГРАММ ОТ ВЗЛОМА

Филиал «Протвино» университета «Дубна»
Кафедра информационных технологий

В настоящее время проблема защиты программ от анализа особенно актуальна. В работе рассматриваются методы и средства анализа программного кода и способы защиты от такого анализа приложений.

Защита программ

Анализ программных реализаций очень важен при решении целого ряда практических задач, например:

- экспертизы качества реализации программных средств обеспечения информационной безопасности;
- исследования программного обеспечения (ПО) на предмет наличия недокументированных возможностей;
- выявления уязвимостей ПО;
- выявления вредоносного ПО [1].

Необходимость в защите ПО возникает не только на государственном уровне или уровне корпораций, но и на уровне средних и малых предприятий, и даже — на индивидуальном. При разработке любой важной программы необходимо использовать системы защиты для предотвращения нелицензионного копирования и распространения. В случае пренебрежения к защите продукта компания или разработчик-индивидуал могут понести серьёзные убытки. Наблюдая в сети Интернет многочисленные пиратские версии продуктов типа *Photoshop, 3dMax* и др. можно только догадываться, какие потери понесли компании-разработчики. И это несмотря на то, что, скорее всего, деньги на защиту программ были выделены серьёзные.

Помощь в защите продукта даёт и законодательство России. Вступление России в ВТО также поможет защищать авторские права.

Реверсинг

«…Термин *Reverse engineering* (или *reversing*) определяется как анализ, разбор (расшифровка) конструкции, структуры, построения программного или аппаратного изделия; восстановление структурной схемы и алгоритма работы; проектирование по образцу; воспроизведение, обратная инженерия. Всё это обычно представляет собой процесс систематического разбора программы (восстановления её исходного текста и структуры) или микросхемы для изучения алгоритмов работы с целью имитации или повторения некоторых или всех её функций в другой форме или на более высоком уровне абстракции, снятия защиты, добавления новых возможностей, восстановления протоколов или исправления ошибок и др…»[2]

Реверсинг необходим и применяется для создания защиты программного обеспечения от нелицензионного и несогласованного с разработчиками изменения, несанкционированного копирования и распространения. При реверсинге исследуемая программа также проверяется на, зашифровку и запаковку. Осуществляются эти проверки при помощи соответствующих инструментов.

От реверсинга необходимо защищаться, поскольку такое исследование продукта приводит к нежелательным последствиям, например, созданию «кряков» программы. Существуют некоторые программы, которые не поддаются дизассемблированию, например *Skype*.

Инструменты для реверсинга

Существует множество различных инструментальных программ, с помощью которых можно исследовать приложения. Их обычно делят на различные категории, определяющие возможности, специфику и задачи конкретного инструмента.

- Дизассемблеры и отладчики. Традиционно оба эти инструмента используются в паре, поскольку дизассемблер выдаёт лишь "чистый код". Современные дизассемблеры способны на большее, например: распознать вызовы стандартных функций и выделить локальные переменные. Но работая с дизассемблером, нет возможности точно знать, какие данные получает та или иная

функция в качестве параметров и что они означают. Чтобы выяснить это, чаще всего требуется изучение если не всей программы, то довольно значительной ее части. Отладчики выполняют другие функции — они позволяют анализировать код в процессе его работы, отслеживать и изменять состояние регистров и стека, и с их помощью править код "на лету". Обратной стороной медали является "неинтеллектуальность" многих отладчиков. Например, *SoftICE* — один из лучших отладчиков — ничего не знает о типах данных. Сейчас появились отладчики, сочетающие высокое качество дизассемблирования и анализа кода с широкими возможностями по его отладке, например, *OllyDbg*.

- Декомпиляторы и специализированные отладчики. С ростом мощности ЭВМ широкое распространение получили компиляторы, создающие некий набор условных инструкций, выполняемый при помощи интерпретатора или виртуальной машины, а не "чистый" машинный код. Интерпретатор может как поставляться отдельно (*Java*), так и быть "прикреплённым" к самой программе. Для анализа таких программ используются специализированные утилиты, переводящие код, понятный лишь интерпретатору, в форму, более удобную для понимания человеком. Также некоторые декомпиляторы могут извлекать информацию об элементах интерфейса, созданных визуальными средствами (например, *DeDe*).
- Распаковщики и утилиты для снятия дампа процессов. Дизассемблировать запакованную или зашифрованную программу невозможно, но если нужно получить хоть какой-то листинг, можно попытаться извлечь из памяти компьютера "снимок" (дамп) программы в момент её работы. Этот дамп уже можно более или менее успешно дизассемблировать. На основе дампа можно воссоздать исполняемый файл программы, причем этот файл будет успешно загружаться, запускаться и работать. Именно на этом принципе и основана работа большинства современных распаковщиков: подопытная программа запускается под управлением распаковщика, который ждёт некоторого события, говорящего о том, что программа полностью распаковалась, тут же "замораживает" программу и сбрасывает ее дамп на диск. Защитные системы нередко пытаются противодействовать получению работоспособного дампа при помощи манипуляций с сегментами и таблицами импорта/экспорта. В этих случаях приходится применять *PE*-реконструкторы, то есть утилиты, обнаруживающие в дампе некорректные ссылки на функции и пытающиеся их восстановить. Многие продвинутые дамперы и распаковщики оснащены встроенными средствами восстановления секций импорта.
- Утилиты анализа файлов. Иногда при реверсинге требуется быстро узнать, каким упаковщиком или защитным ПО обработана та или иная программа, найти все текстовые строки в дампе памяти, просмотреть содержимое файла в виде текстовых строк, вывести в файл список импортируемых и экспортируемых программой функций и многое другое. Для всех этих целей существуют специализированные утилиты (например, *PEiD*), позволяющих быстро проанализировать файл на наличие тех или иных признаков.
- Шестнадцатеричные редакторы и редакторы ресурсов. Шестнадцатеричные редакторы — это наиболее старые из программистских инструментов. Они часто используются крекерами. Сейчас только крекеры занимаются тем, что исправляют байты в готовых программах, а редакторы ресурсов используются для изменения присоединенных к исполняемому файлу ресурсов. Именно при помощи редакторов ресурсов выполняется значительная часть работ по "независимой" русификации программ и доработке интерфейсов.
- Утилиты мониторинга. При исследовании программ также требуется знать, какие именно действия выполняет та или иная программа, откуда читает и куда записывает данные, какие стандартные функции она вызывает — и с какими параметрами. Получить эти знания помогают утилиты мониторинга. Такие утилиты делятся на две большие группы:
  - утилиты, которые отслеживают сам факт возникновения каких-либо событий;
  - утилиты, которые позволяют выявить один или несколько специфических типов изменений, произошедших в системе за некий промежуток времени.

К первой группе относятся *API*-шпионы, перехватывающие вызовы системных (а более продвинутые — и не только системных) функций, хорошо известные утилиты *RegMon, FileMon, ProcMon*. Эти утилиты предоставляют подробную информацию об отслеживаемых событиях, но генерируют объемные и неудобные для анализа лог-файлы, если отслеживаемые события происходят достаточно часто.

Вторая группа представлена программами, создающими снимки реестра, жёсткого диска, системных файлов и т. п. Эти программы позволяют сравнить состояние компьютера "до" и "после" некоего события, построить список различий между состояниями и, на основе этого

списка, сделать определенные выводы. Основная проблема при работе с такими утилитами: вы можете получить, кроме интересующей вас деятельности, программы, лог "жизнедеятельности" других, параллельно работающих программ, и операционной системы.

- Прочие утилиты. Есть множество утилит, не принадлежащих ни к одной из приведённых категорий или попадающих в несколько из них. Крекинг — занятие многогранное и столь же многогранны инструменты, в нём используемые. Более того, некоторые из утилит, которые могут быть полезны для крекера, создавались для совершенно иных целей (например, программа *ArtMoney*, которая была предназначена для того, чтобы продлевать жизнь в компьютерных играх).

Антиреверсинг

Для антиреверсинга применяются различные методы — от обнаружения отладчика и «родительского процесса» программы до определения использования виртуальной машины и подключения чужих программ к нашей с выполнением впоследствии несанкционированных действий.[4]

Достигнутые результаты

В результате проделанной работы получены первоначальные знания об инструментах исследования программ:

- *OllyDbg* — двоичный отладчик;
- *IDA Pro* — дизассемблер;
- *Kodovick* — HEX редактор.

С помощью *OllyDbg* и *IDA* можно исследовать структуру изучаемой программы и взаимодействие её элементов.

Полученные знания были применены в простом случае: была написана простая программа, выводящая приветствие определенному лицу — надо было изменить это приветствие на приветствие другому лицу, работая только с исполняемым файлом программы.

Перспективы изучения

В рамках дальнейшего изучения реверсинга и защиты программ нужно исследовать реализацию на ассемблере условных переходов, циклов, вызовов функций; ознакомиться с другими программами в данной области информационных технологий, исследовать разные специфические аспекты применения прикладного программного обеспечения.

Сфера защиты программ сегодня быстро развивается: постоянно разрабатываются новые технологии взлома, а следовательно — изучаются и совершенствуются и методы защиты.

Библиографический список

1. Проскурин, В. Г. Защита программ и данных. / В. Г. Проскурин — М.: Издательский центр «Академия», 2012. — 208 с.
2. Панов, А. С. Реверсинг и защита программ от взлома. / А. С. Панов — СПб.: БХВ-Петербург, 2006. — 256 с: ил.
3. http://reversing.do.am
4. http://www.anti-reversing.com