**Этапы преодоления систем защиты программного обеспечения**

С.А.Середа

В статье описывается обобщённая процедура анализа и деактивации систем защиты программного обеспечения (ПО) от несанкционированного использования и копирования. Знание методик, которые используются злоумышленниками ('crackers') для преодоления систем программной защиты, позволяет более точно определить слабые места существующих систем, а так же проектировать новые, более устойчивые к атакам 1. Нами предлагается описание унифицированной методики анализа и преодоления систем программной защиты, являющейся результатом обобщения и систематизации многочисленных приёмов "взлома" программ, публикуемых в современной литературе [ 1], а также, доступных в глобальной сети. По итогам проведённого исследования предложен комплексный критерий оценки устойчивости систем защиты программного продукта.

В настоящее время наиболее популярным средством борьбы с нелегальным распространением коммерческих программных продуктов остаётся программная защита их двоичного кода. Существуют системы защиты программного обеспечения разных типов [ 4, 6], все они постоянно развиваются. В то же время, имеются и средства, позволяющие исследовать защищённые программы и отключать системы их защиты [ 5]. В условиях такого динамического равновесия важным фактором, влияющим на стойкость систем защиты, является методическое обеспечение как специалистов по защите ПО, так и злоумышленников.

Мы считаем, что изучению методов, используемых для анализа и преодоления систем защиты ПО, не уделялось достаточного внимания, в то время как их знание позволяет в значительной мере сократить количество уязвимых мест в системах защиты.

Нами было проведено исследование современных подходов к программно-технической защите программных продуктов, а также подходов к преодолению такой защиты. В результате можно сделать вывод, что как с одной (защита), так и с другой ("взлом") стороны основное внимание уделялось и до сих пор уделяется прикладным приёмам защиты ПО или её преодоления [ 3]. В то же время, нам не удалось отыскать ни обобщённых методик проектирования и реализации систем программной защиты, ни аналогичных методик анализа и преодоления таких систем.

Анализ приёмов преодоления систем программной защиты различных типов позволил выявить ряд общих закономерностей. В данном случае, положительную роль сыграло многообразие описываемых злоумышленниками методик, различный уровень их сложности, а также их принадлежность к различным стадиям анализа систем защиты ПО. Таким образом, стало возможным систематизировать информацию по данному вопросу и описать обобщённую процедуру анализа и преодоления систем защиты программных продуктов.

По нашему мнению, любые приёмы и методы анализа и преодоления систем защиты ПО можно свести к ряду стандартных этапов. (См. блок-схему на Рис.1)

Первый этап - определение цели атаки. В первую очередь злоумышленнику необходимо определить цель, для достижения которой он будет атаковать систему защиты программного продукта. Среди возможных целей можно выделить следующие три: личное использование программного продукта; распространение средств, отключающих систему защиты продукта ('crack-files'); несанкционированное распространение самого программного продукта ('warez'). В зависимости от перечисленных целей подход к "взлому" системы защиты того или иного программного продукта может варьироваться. В частности, если злоумышленник планирует использовать программный продукт в личных целях, он может воспользоваться методами, требующими высокой квалификации пользователя, необходимой для работы с отключенной системой защиты. Если предполагается распространение средств отключения системы защиты конкретного продукта злоумышленнику необходимо ориентироваться на неквалифицированного в вопросах защиты ПО пользователя. Это может потребовать дополнительных усилий и затрат времени. Наконец, если злоумышленник планирует распространять "взломанный" программный продукт, ему, как правило, необходимо осуществить прямое отключение системы защиты, что значительно сложнее, чем просто её обход.

Второй этап - поиск проявлений системы защиты. Проявления работы систем защиты могут иметь следующий вид:

ограничение времени использования продукта ('trial/evaluation');

ограничение функциональности продукта ('demo/crippled');

регулярные напоминания о необходимости регистрации ('nag screens');

запрос регистрационного кода ('regcode');

генерация системных ошибок ('crash/GPF');

сбор и передача персональных данных ('spyware');

вредоносные действия ('malware');

внесение ошибок в обработку данных и др.

Возможны и комбинации перечисленных проявлений.

В зависимости от этих проявлений злоумышленник может применять различные подходы к исследованию системы защиты, а так же затрачивать разное время и усилия на её преодоление. Например, напоминания о необходимости регистрации направлены на чисто психологическое воздействие и их отключение обычно не представляет большой сложности. Программные продукты с ограничениями времени использования всегда уязвимы для атак. Продукты с ограничениями функциональности уязвимы, если есть возможность снять ограничения с помощью регистрационного кода, и отключённые подпрограммы не зашифрованы. Системы, отключающие защиту при вводе верного регистрационного номера, можно преодолеть при отсутствии шифрования кода ПО стойкими криптоалгоритмами (DES, RSA, IDEA, etc.). Что же касается систем, генерирующих внутренние или системные ошибки, собирающих персональные данные или осуществляющих вредоносные действия, их анализ и преодоление, как правило, требуют дополнительных затрат времени, так как подобные проявления систем защиты носят неявный и нелокализованный характер.

Третий этап - предварительный анализ работы защищённого продукта. На начальном уровне исследования системы защиты злоумышленник отслеживает активность программы, связанную с созданием и удалением обычных и скрытых файлов; обращением к системным файлам, портам ввода/вывода. Также контролируется строка запуска программы, используемые сервисы операционной системы и другие параметры. При помощи подобного мониторинга работы программного продукта злоумышленник получает общее представление об используемом механизме защиты и возможных путях её преодоления. Как правило, системы защиты используют какой-то один из указанных видов активности программы для реализации своих функций. Например, для реализации ограничения времени использования продукта используется либо создание скрытых файлов, либо запись данных в системные файлы. В этом случае мониторинг работы защищённого приложения с различными файлами даёт важную информацию для дальнейшего исследования системы защиты. Некоторые системы защиты записывают данные в область ППЗУ, в неиспользуемые участки дорожек жёсткого диска или используют электронный ключ. Мониторинг работы приложения с портами ввода/вывода даёт возможность обнаружить подобные факты. Аналогичным образом, системы защиты, использующие регистрационный код, как правило, сохраняют его в собственных или системных файлах. Отслеживание неудачных попыток найти файл или найти запись в системном файле также даёт злоумышленникам предварительную информацию о механизме защиты программного продукта.

Четвёртый этап - предварительный анализ кода программного продукта, определение типа защиты и локализация системы защиты. Проводя поверхностное исследование кода программного продукта, в большинстве случаев злоумышленник определяет тип, а иногда и производителя системы защиты. Становится возможным установить наиболее вероятное физическое расположение кода системы защиты в коде программного продукта. Для подобного исследования кода программы обычно используется шестнадцатеричный редактор с возможностью дизассемблирования машинных инструкций, поиска заданных байтовых последовательностей и др. Но, кроме средств просмотра машинного кода, злоумышленники используют и автоматические анализаторы исполняемых файлов. Эти программные средства по заложенным в них сигнатурам способны распознавать большинство популярных систем защиты ПО, а также определять производителя и версию компилятора, при помощи которого программа была собрана.

Полученная в результате информация, вкупе с данными мониторинга работы программного продукта, даёт злоумышленнику полную обзорную информацию об используемой в программном продукте системе защиты. Нередки случаи, когда дальнейший анализ системы защиты уже на данном этапе становится излишним и собранной информации достаточно для обхода или преодоления защиты.

Пятый этап - оценка стойкости и слабых мест системы защиты ПО для выбора оптимального способа её преодоления. После сбора информации о системе защиты производится оценка её стойкости и слабых мест, а также возможности применения стандартных средств и приёмов преодоления систем защиты данного типа. Злоумышленник, используя доступную информацию об устройстве и стойкости различных систем программной защиты, определяет стойкость системы защиты, реализованной в атакуемом программном продукте. Например, если в атакуемом продукте используется система защиты, ограничивающая время его использования, то уязвимостью является необходимость получения системного времени и даты. В системах с запросом регистрационного кода и/или ограничением функциональности продукта слабым местом является блок проверки корректности регистрационного кода. Системы, использующие ключевые файлы, диски или электронные ключи могут иметь слабые места в блоках проверки корректности ключевых данных или в блоках обмена данными. Другой уязвимостью для любых типов защиты является использование стандартных библиотек подпрограмм, при помощи которых реализуется система защиты ПО.

Оценив стойкость системы защиты, злоумышленник затем выбирает один или несколько эффективных способов её преодоления. На этот выбор влияет и исходная постановка цели, как это было указано выше. В частности, для случая с ограничением времени функционирования продукта злоумышленник может избрать три различных пути:

если он планирует самостоятельно использовать продукт, возможно, будет достаточно вручную изменять системную дату или переустанавливать программный продукт по истечении разрешённого срока его использования, стирая скрытые файлы и/или записи в системных файлах, сделанные системой защиты;

при желании в дальнейшем распространять средство снятия защиты с продукта злоумышленнику потребуется исследовать алгоритмы системы защиты и разработать программное средство её статического или динамического преодоления;

если же планируется "пиратская" реализация программного продукта, злоумышленник должен будет либо распространять его вместе со средством снятия защиты либо произвести полное статическое отключение системы защиты в программном продукте.

В зависимости от выбранного способа преодоления защиты изменяется и дальнейшее её исследование.

Шестой этап - направленное исследование системы защиты. На данной стадии злоумышленник выполняет статический и/или динамический анализ системы защиты атакуемого продукта. Статический анализ, в зависимости от языка программирования, на котором написана программа, может осуществляться при помощи дизассемблирования объектного кода или при помощи его декомпиляции. Применение дизассемблера даёт злоумышленнику возможность работы с последовательностью инструкций целевого процессора, в которую исходный текст программы был превращён при компиляции. В случае же декомпиляции объектного кода злоумышленник получает текст на языке высокого уровня в максимально близком к исходному виде. Получив текст программы на языке ассемблера или языке высокого уровня, злоумышленник анализирует его и определяет алгоритм преодоления защиты продукта.

В случае затруднений, связанных со статическим анализом, например, когда объектный код зашифрован или динамически изменяется, прибегают к его динамическому анализу. Как правило, динамический анализ систем защиты выполняется при помощи программного отладчика. При этом исследуемое приложение запускается в среде программы-отладчика и выполняется в пошаговом режиме или с использованием заданных точек останова. В режиме отладки становится возможным отслеживать изменения состояния программы, происходящие в процессе её выполнения (например, изменение содержимого регистров процессора, срабатывание команд условного перехода, параметры вызова подпрограмм, получаемые или передаваемые через порты ввода/вывода данные и др.). В этом же режиме выполняется прохождение алгоритмов, например, проверки/генерации корректного регистрационного кода.

Динамический анализ может также выполняться при помощи уже упоминавшихся выше мониторов активности приложения. Если на этапе предварительного анализа при помощи этих средств отслеживается общая направленность работы программы, то на этапе динамического анализа системы защиты отслеживается не только факты обращения к файлам, портам и системным сервисам, но и все параметры этих обращений: данные, записываемые/считываемые из скрытых и системных файлов, параметры вызова системных сервисов, протоколы обмена с портами ввода/вывода и др. В случае применения мониторов активности не имеет значения, реализованы ли в рамках системы защиты блоки противодействия отладке приложения или нет.

По завершении данного этапа анализа системы защиты злоумышленник обладает полной информацией о том, как технически осуществить обход или преодоление системы защиты программного продукта.

Седьмой этап - преодоление системы защиты. На финальном этапе, учитывая поставленную цель, выбранный способ преодоления защиты и найденную техническую процедуру её обхода или преодоления, злоумышленник реализует преодоление защиты на практике. Под обходом системы защиты понимаются действия, напрямую не относящиеся к противодействию системе защиты атакуемого программного продукта. В качестве примера можно привести периодическую реинсталляцию программных продуктов с ограниченным сроком использования; изменение системной даты до запуска программы и установка корректной даты по завершении её работы; удаление/замену скрытых файлов со счётчиками запуска; удаление/замену соответствующих строк в системных файлах; написание и использование генераторов регистрационных кодов; отслеживание и автоматическое завершение диалогов с напоминанием о необходимости регистрации продукта и т.п.

Преодоление системы защиты может осуществляться тремя основными путями:

Статическая модификация кода программного продукта, приводящая к отключению системы защиты.

Динамическая модификация кода программного продукта во время его выполнения.

Эмуляция ключевого файла, диска или электронного ключа защиты.

В первом случае в объектный код программного продукта вносятся изменения, дезактивирующие систему его защиты. Как правило, они касаются команд условного перехода типа "зарегистрированная версия/незарегистрированная версия" или "верный регистрационный код/неверный регистрационный код". Иногда модифицируются элементы данных программы, содержащие определённые флаги, по которым система защиты судит о наличии регистрации программного продукта. Нередко, для статической модификации кода требуется его предварительная дешифрация или восстановление по образу в оперативной памяти. Такая операция часто обладает значительной трудоёмкостью и требует дополнительного исследования системы защиты. Поэтому злоумышленники прибегают к статической модификации объектного кода атакуемого продукта либо при условии, что код ПО не зашифрован, либо когда преследуют цель "пиратского" распространения ПО. При этом распространяться может как программное средство, выполняющее статическую модификацию продукта, так и просто данные, позволяющие выполнить это вручную.

К динамической модификации кода злоумышленники прибегают в случаях, когда дешифрация или восстановление объектного кода программы требует слишком высоких затрат. Под динамической модификацией понимается изменение кода программы в оперативной памяти во время выполнения. Подобная модификация должна производиться при каждом новом запуске программы. Для реализации этого процесса злоумышленниками используются специальные программные средства, осуществляющие загрузку целевого приложения как своего дочернего процесса. Такая загрузка даёт доступ к адресному пространству приложения в оперативной памяти, а соответственно и возможность динамического изменения его кода. Как правило, программные средства, ориентированные на конкретный программный продукт, распространяются злоумышленниками в глобальной сети отдельно или вместе с самим продуктом.

Эмуляция используется, в основном, если система защиты включает в себя электронный ключ, реже, если присутствует ключевой диск или ключевой файл. Суть данного метода заключается в подделке ответов на запросы защищённого приложения к отсутствующему ключу, диску или файлу таким образом, что система защиты не обнаруживает его отсутствия. С одной стороны, использование эмуляции требует значительных усилий, связанных с исследованием протоколов обмена данными между блоками системы защиты, и программированием эмулятора (нередко в виде драйвера). С другой же стороны, эмуляция, как правило, не требует модификации кода ПО, а следовательно избавляет злоумышленника от необходимости дешифрации или исправления значительных участков программного кода. Чаще всего эмуляторы распространяются отдельно либо вместе с "пиратскими" копиями ПО.

Преодоление системы защиты атакуемого продукта является последним этапом процесса анализа и преодоления систем защиты. После этого злоумышленник начинает использование программного продукта, распространение средства отключения системы защиты или самого продукта.

Знание последовательности действий злоумышленника позволяет разрабатывать гибкую политику программно-технической защиты программных продуктов. Значительная доля современных систем защиты ориентирована на затруднение лишь части из описанных этапов их анализа и преодоления. Например, нередки случаи, когда система защиты обладает мощными механизмами противодействия дизассемблированию кода защищённого приложения и его отладке, но не способна противостоять мониторингу. Существуют, также, системы защиты, базирующиеся на шифровании объектного кода приложения, но обладающие чрезвычайно простой логикой, что позволяет злоумышленникам легко осуществлять динамическую модификацию кода. Кроме того, львиная доля систем защиты не только не маскирует своего присутствия, но и активно информирует о нём пользователя. Подобная стратегия в значительной мере облегчает злоумышленникам предварительный анализ системы защиты и её локализацию.

Таким образом, можно заключить, что правильно спроектированная система защиты, по крайней мере, должна затруднять своё первичное обнаружение, противодействовать мониторингу работы приложения, его отладке, дизассемблированию и/или декомпиляции, обладать сложной логикой работы, трудно поддающейся анализу, а так же создавать значительные трудности для динамической и/или статической модификации кода защищаемого программного продукта.

На основе высказанных выше соображений можно сформулировать частные критерии устойчивости защищаемого продукта к атакам. По нашему мнению, можно предложить следующие критерии:

Трудность распространения продукта (требуется пользовательская документация, необходимо специальное оборудование, объём продукта затрудняет его распространение по сети и т.п.)

Устойчивость к поиску проявлений системы защиты (маскировка факта срабатывания системы защиты);

Устойчивость к предварительному анализу защищённого продукта (маскировка функциональности системы защиты);

Устойчивость к предварительному анализу программного кода (маскировка физического расположения системы защиты в коде продукта);

Наличие уязвимостей в системе защиты;

Устойчивость к статическому и динамическому анализу кода (противодействие анализу алгоритмов системы защиты);

Устойчивость к обходу системы защиты (наличие логических ошибок в алгоритмах защиты);

Устойчивость к преодолению системы защиты (трудность статической и динамической модификации кода продукта).

Исходя из данного набора частных критериев, можно построить комплексный критерий устойчивости программного продукта к атакам. Этот критерий формулируется следующим образом:

Kкомпл. = n ∑ i=1 αi ki,

где ki - частный критерий устойчивости, αi - весовой коэффициент частного критерия устойчивости, определяющий его важность, а n - количество частных критериев устойчивости.

Таким образом, с помощью метода экспертных оценок становится возможным сравнивать между собой различные варианты защиты одного программного продукта либо сравнивать реализации защиты различных программных продуктов одного класса. Сама реализация метода экспертных оценок может быть осуществлена двумя основными способами:

Если отсутствует статистика предыдущих оценок, возможно лишь сравнение различных вариантов между собой. В этом случае по единой методике оцениваются различные варианты реализации защиты программных продуктов. Лучшим считается вариант с максимальным значением комплексного критерия.

При наличии статистики предыдущих оценок возможно выделение интервалов значений частных и совокупного критериев. Тогда варианты, не попадающие в необходимый интервал значений отбрасываются. Хотя и в данном случае лучшим считается вариант с максимальным значением комплексного критерия, данный способ позволяет оценить единственный вариант реализации защиты, не рассматривая его альтернатив.

Среди возможных направлений дальнейших исследований в данной сфере, по нашему мнению, следует отметить подробное изучение отдельных этапов преодоления систем программной защиты; выявление зависимости между сроком преодоления систем защиты разных типов и набором используемого злоумышленником программного инструментария; сбор статистических данных по стойкости систем защиты разных типов; разработку методик оценки стойкости систем защиты ПО и сроков гарантированной устойчивости подобных систем к атакам.

**Список литературы**

Касперски Крис. Техника и философия хакерских атак. - М.: Солон-Р, 1999.

Майерс Гленфорд Дж. Искусство тестирования программ. - М.: Финансы и статистика, 1982.

Расторгуев С.П., Дмитриевский Н.Н. Искусство защиты и раздевания программ. - М.: Совмаркет, 1991. - 94 с.

Семьянов П.В., Зегжда Д.П. Анализ средств противодействия исследованию программного обеспечения и методы их преодоления. // КомпьютерПресс. - 1993. №11.

Середа С.А. Анализ средств преодоления систем защиты программного обеспечения. // ИНФОРМОСТ: Радиоэлектроника и Телекоммуникации. - 2002. №4(22). С. 11-16.

Середа С.А. Оценка эффективности систем защиты программного обеспечения. // КомпьЛог. - 2000. №2.

1 Под атакой нами понимается комплекс действий, направленных на нарушение защиты ПО от копирования и использования.