A. P. Durakovskiy, S. V. Ponomarev

The Model of System to Counter to Software Tools Learning Code

Key words: debugger, disassembler, protection software

The article describes the main types of software used for the study of the code, the shortcomings of existing systems of protection programs and proposed a new model of protection system of code.

А. П. Дураковский, С. В. Пономарев

МОДЕЛЬ СИСТЕМЫ ПРОТИВОДЕЙСТВИЯ ПРОГРАММНЫМ СРЕДСТВАМ ИЗУЧЕНИЯ КОДА

В статье рассмотрены основные типы программных средств, применяемых для изучения программного кода, недостатки существующих систем защиты программ и предложена новая модель системы защиты программного кода.

Введение

В настоящее время существует огромное количество программ, которые помогают специалистам во всех сферах их деятельности. Например, текстовые редакторы, которые упрощают работу с текстом, или средства для разработки программного обеспечения, которые позволяют программистам создавать новые программы. Каждая из программ содержит уникальную наработку или нововведение, которое отличает ее от других. Многие из разработчиков старались сохранить в секрете свою интеллектуальную собственность и получать прибыль от разработок своего программного обеспечения. С этой целью и были созданы первые системы защиты, которые встраивались в программы на этапе проектирования и представляли собой простые системы идентификации и аутентификации. Они основывались на знании пользователем ключа (серийного номера), который поставлялся вместе с программным обеспечением. Пользователь при запуске программы вводил известный ему ключ, затем система защиты сверяла его с эталонным, и если ключи совпадали, то пользователь мог работать с программой, если же нет, то выводилось сообщение о том, что ключи неодинаковые и программа не запускалась. В первое время развитию методов и систем защит уделялось мало внимания, так как не было способов обойти существующие. Но с развитием информационных технологий стали появляться программные средства, позволяющие злоумышленникам изучать работу программ без знания исходных кодов. Это были первые отладчики, с помощью которых пользователи стали обходить существующие методы защиты, в результате чего многие разработчики стали нести убытки. С появлением отладчиков стали активно развиваться методы и системы защиты, которые затрудняют или делают невозможным работу отладчика. Несмотря на множество разработанных методов защиты программ, программные средства для изучения кода также активно развивались, появлялись новые средства, такие как дизассемблер, декомпилятор и другие, которые уже позволяли лучше понимать логику работы программы и получать ее исходный текст.

На данный момент развитие методов и техник защиты отстает от развития программных средств для изучения кода. Разработчики для защиты своего программного обеспечения используют или устаревшие методы и техники защиты, или коммерческие системы защиты. Некоторые из этих систем предоставляют эффективную защиту и затрудняют изучение программ, другие же спроектированы с использованием устаревших методов и техник, которые быстро обходятся квалифицированными программистами. Поэтому необходимо создание новой системы защиты

программного обеспечения, которая должна учитывать особенности программных средств для изучения кода и существующие способы противодействия им других систем защит или примеры их обхода, так как это позволит не допустить подобные ошибки в проектировании и разработать более устойчивую систему.

Обзор основных типов программных средств изучения кода

Для создания эффективной системы защиты необходимо тщательно исследовать средства, которые будет применять злоумышленник для ее изучения. Ведь в каждом программном средстве есть свои достоинства и недостатки, которые необходимо учитывать. Одним из программных средств, которое применяется для изучения работы программы, является отладчик [1. С. 188–216]. Он позволяет выполнять динамический анализ, в ходе которого злоумышленник получает больше информации о работе программы. Это становится возможным, потому что отладчик создает бесконечный цикл внутри программы и ожидает системного события отладки. Когда такое сообщение генерируется, цикл прерывается и вызывается соответствующий обработчик событий. Событиями, которые служат командой для отладчика, являются точки останова, позволяющие останавливать отлаживаемый процесс. Таким образом, можно изучить каждую инструкцию, выполняемую программой, проверить переменные, аргументы стека и узнать данные, находящиеся в этот момент в памяти, до того, как они будут перезаписаны. Это позволяет программистам получать больше информации о работе программы, а также подробно изучать механизмы защиты.

Существуют три основных вида точек останова [2]:

1. Программные точки останова — часто используемые, так как позволяют останавливать процессор при выполнении инструкций программы, представляют собой однобитную инструкцию, которая останавливает процесс и передает управление обработчику исключений. Количество устанавливаемых программных точек в программе неограниченно. Но при их установке модифицируется код программы, которую может отследить механизм защиты целостности и прекратить работу программы.

2. Аппаратные точки останова — данный тип точек устанавливается на уровне процессора в специальных регистрах и не модифицирует изучаемую программу. Всего можно установить до четырех точек останова.

3. Точки останова памяти — позволяют менять разрешения на странице памяти, в которой могут содержаться важные данные. Благодаря этому такие точки позволяют подробно изучать механизмы защиты, основанные на сравнении введенного пароля с эталонным, так как эталонный пароль во время сравнения находится в памяти в открытом виде.

Становится очевидно, что отладчик представляет собой сильное программное средство для изучения программ, так как позволяет получать много ценной информации об их работе.

Следующий тип программных средств, так называемый дизассемблер [3], позволяет преобразовывать машинный код в текст программы на языке ассемблера, который дает возможность программистам узнать архитектуру программы, а также определить место размещения защитных механизмов.

Еще один тип программных средств для изучения кода называется шестнадцатеричным редактором [4]. Он позволяет редактировать данные, такие как образ диска, содержимое оперативной памяти и другие. Стоит отметить, что данные представлены в виде цепочек байтов в шестнадцатеричной системе счисления. Благодаря этому типу средств становится возможным найти в изучаемой программе внедренный эталонный ключ без использования отладчика или дизассемблера. Однако в настоящий момент ключи генерируются в самой программе и в открытом виде не хранятся.

По этой причине в существующих системах защиты применяется несколько методов и техник, которые повышают устойчивость к изучению программного обеспечения, но вместе с

тем имеют свои недостатки. Например, многие системы защиты спроектированы по общему шаблону, раскрытие которого приводит к нейтрализации защиты. Под шаблоном понимается применение однотипных методов и техник противодействия программным средствам. По этой причине многие из систем защиты являются неэффективными, так как алгоритм противодействия им, в большинстве случаев, является похожим.

Модель системы противодействия программным средствам изучения кода

Для устранения приведенных недостатков в системах защиты была разработана новая модель системы, которая позволит создать уникальную систему защиты и более эффективно противодействовать программным средствам для изучения кода.

Система представляет собой систему внедрения защитных модулей в защищаемую программу. Система внедрения на основе известных алгоритмов внедрения кода встраивает защитные модули в программу. Защитные модули создаются самим разработчиком и представляют собой объектные файлы, написанные на языках программирования C/C++, ассемблера и др., которые содержат в себе свои методы и техники или применяют существующие, что позволит устранить однотипность защиты, так как в случае изучения злоумышленником защитного модуля и разработки алгоритма противодействия данный модуль может быть быстро заменен другим, а изучение системы внедрения с применением программных средств также не предоставит злоумышленнику никакой полезной информации. Еще одним достоинством такой системы является тесная интеграция защитных механизмов с самой программой, что позволяет устранять критические ошибки на этапе проектирования. К тому же такой подход позволяет создавать несколько защитных модулей, которые можно будет внедрять в программу постепенно, по мере изучения и обхода существующих механизмов защит.

Стоит отметить, что модули могут содержать разнообразные защитные техники, например текст программы может быть зашифрован и расшифровываться с помощью системы внедрения или иметь собственный расшифровщик. Поэтому система спроектирована с учетом возможных модификаций под нужды разработчиков, позволит им самостоятельно создавать более эффективную защиту и сохранять в секрете свою интеллектуальную собственность.

СПИСОК ЛИТЕРАТУРЫ:

1. *Dang B., Gazet A., Bachaalany E.* Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation. Indianapolis, 2014 — 383 p.
2. Общие сведения об отладке: точки останова. URL: http://msdn.microsoft.com/ru-ru/library/vstudio/4607yxb0%28v=vs.100%29.aspx (дата обращения: 20.12.2014).
3. Disassembler // Wikipedia, the free encyclopedia. URL: en.wikipedia.org/Disassembler (дата обращения: 18.12.2014).
4. Hex editor // Wikipedia, the free encyclopedia. URL: en.wikipedia.org/Hex_editor (дата обращения: 18.12.2014).

REFERENCES:

1. *Dang B., Gazet A., Bachaalany E.* Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation. Indianapolis, 2014— 383 p.
2. Obshchie svedeniya ob otladke: tochki ostanova. URL: http://msdn.microsoft.com/ru-ru/library/vstudio/4607yxb0%28v=vs.100%29.aspx (date of access: 20.12.2014).
3. Disassembler // Wikipedia, the free encyclopedia. URL: en.wikipedia.org/Disassembler (date of access: 18.12.2014).
4. Hex editor // Wikipedia, the free encyclopedia. URL: en.wikipedia.org/Hex_editor (date of access: 18.12.2014).