ISSN 2225-9309 (Print)
2024. № 2 (31). С. 59–63

Молодежный
Вестник УГАТУ

http://journal.ugatu.su

УДК 330

Обзор методов защиты информации от обратной разработки (Reverse Engineering)

А. В. Давлетшин[1], А. А. Журавлев[2]

[1] artur.davletshin.v@yandex.ru, [2] zhuravlevaao@mail.ru

[1-2] ФГБОУ ВО «Уфимский университет науки и технологий» (УУНИТ)

Аннотация. В статье рассматривается процесс обратной разработки (reverse engineering) как метод анализа программного обеспечения, позволяющий понять его структуру, язык программирования и восстановить исходный код. Основное внимание уделяется инструментам, таким как дизассемблеры (IDA, OlyDbg, Ghidra) и отладчики, которые позволяют исследовать и модифицировать код приложений. Обсуждаются различные типы языков программирования (компилируемые, интерпретируемые и байт-кодовые) и методы взлома, включая дизассемблирование, изменение кода и установку хуков. Также рассматриваются способы защиты от обратной разработки, такие как обфускация, упаковка, защита от отладчиков и контроль целостности кода. Статья подчеркивает важность этих методов для обеспечения безопасности программного обеспечения и защиты интеллектуальной собственности.

Ключевые слова: Обратная разработка, дизассемблер, отладчик, взлом, обфускация, защита, код, пакеры, целостность, WinApi.

ВВЕДЕНИЕ

Reverse engineering с английского обратная разработка, способ с помощью которого можно понять как устроена программа, на каком языке написана, и возможность восстановить её исходный код. Основными приложениями в обратной разработке служат дизассемблеры, которые представляют код программы в виде псевдокода ассемблера или языка С, и отладчики, при подключении которых возможно пошагово просмотреть выполнение программы и изменить код приложения.

Существует три вида языков программирования это компилируемые, интерпретируемые и байт-кодовые языки. Компилируемые языки сразу преобразуются в машинный код без промежуточных стадий. В интерпретируемых языках код транслируется в компилируемый язык и только после этого уже в машинный код. Байт-кодовые языки схожи с интерпретируемыми языками, но вместо компилируемого языка транслируется в некоторый байт-код и после этого в машинный. Поэтому при декомпиляции интерпретируемых и байт-кодовых языков из-за дополнительного этапа в компиляции легко вытянуть исходный код программы.

Обратную разработку активно используется специалистами, которые обеспечивают безопасность в информационной сфере. Однако этот способ используется не только для обеспечения безопасности, но и для создания пиратский копий программ, обхода защиты антивирусных программ, организации фейкового общения с сервером, где под видом приложения общается программное обеспечение злоумышленников.

Целью работы является изучить методы взлома приложений и способы их защиты. Установить, какой метод обеспечивает наилучшую защиту.

АНАЛИЗ МЕТОДОВ ОБРАТНОЙ РАЗРАБОТКИ

Основным программным обеспечением в обратной разработке которые позволяют проводить отладку, являются: Interactive Disassembler (IDA), OlyDbg, Ghidra, Cheat Engine.

IDA – программа имеющая возможность представления программ в виде псевдокода на ассемблере или на языке С сплошным текстом или виде блок схем. Присутствуют инструменты, облегчающие работу по обратной разработке. Имеется возможность замены встроенного 32-битного отладчика.

OlyDbg – 32-битный дизассемблер со встроенным отладчиком, возможность просмотра псевдокода сплошным текстом только на ассемблере (Рис. 1).

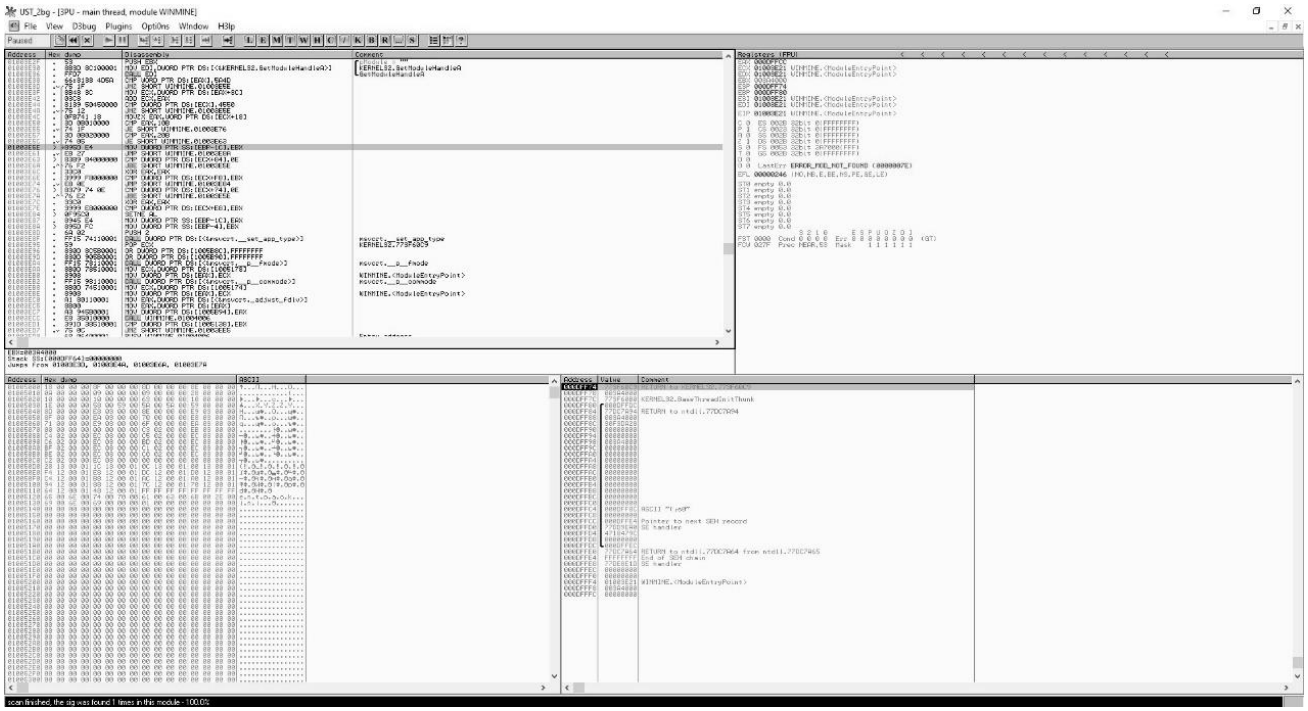

Рис. 1. Пример дизассемблера в программе OlyDbg

Cheat Engine – программа позволяет просматривать программы в виде псевдокода на ассемблере, имеет возможность подключения нескольких видов отладчиков.

Существует несколько основных видов взлома приложений с помощью обратной разработки:

Получение информации через дизассемблированный код – метод основывается на том, что мы загружаем программу в дизассемблер и читаем его код. Если нужно взломать лицензионный ключ к приложению, то нужно выяснить, где происходит проверка ключа в коде и какой ключ проверяется. Чтобы выяснить, где хранится нужная информация надо знать как устроены графические приложения в операционной системе Windows. Стоит отметить, что неважно какой язык использовался для написания программы в дизассемблере у всех будет одинаковая структура, которая основывается на WinApi. Зная структуру Windows (Рис. 2) у нас, есть некоторая основная функция WinMain которая общается с ОС, а она передает информацию функции WndProc. Поэтому мы находим в коде WinMain и WndProc и находим, где хранится пароль.

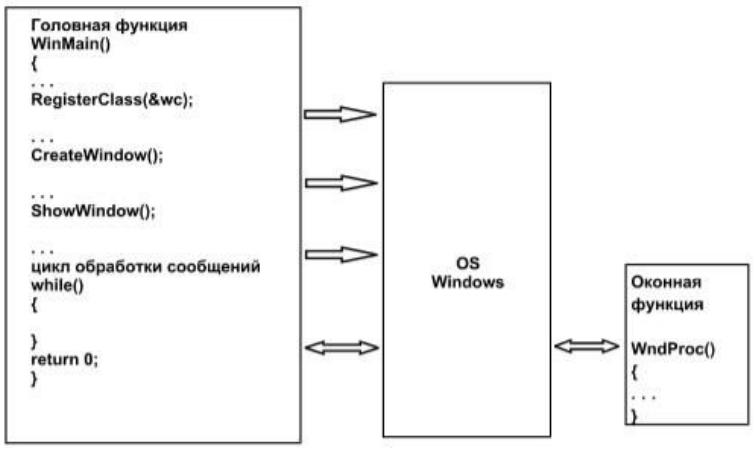

Рис. 2. Структура Windows приложения

Изменение кода через отладчик – основывается на изменении внутреннего кода программы что позволит убрать функции, отвечающие за проверку лицензии приложения или функции, отвечающие за защиту от факта самого взлома. Для этого нужно подключить отладчик, а остальные действия остаются без изменений, как поиск точки входа в программу и сам факт проверки ключа.

Выкачка приложения их памяти – используется если программа не может найти функции, такое происходит если наложены средства защиты приложения такие как обфускация. Windows позволяет создать дамп памяти с помощью диспетчера задач, но против хорошей защиты лучше использовать отладчик. Стоит отметить, что дамп программы может не работать из-за того, что не все модули выкачались из памяти, однако этого хватит чтобы понять, где хранится проверка ключа.

Метод поиска по значению – заключается в том, что мы ищем ячейку памяти по тому значению, которое оно принимает для этого, мы должны просканировать область программы в оперативной памяти и по нужному значению найти адрес нужной ячейки и туда записать свое значение.

Установка хуков – это внедрение стороннего кода в приложение, устанавливающего прыжок на нужную функцию для этого, ищут адрес функции и устанавливают прыжок на неё. Обычно хук внедряют в программу в виде динамической библиотеки (dynamic link library или сокращено dll).

АНАЛИЗ МЕТОДОВ ЗАЩИТЫ ОТ ОБРАТНОЙ РАЗРАБОТКИ

Чтобы защитить приложения от взлома обратной разработкой можно воспользоваться обфускацией кода, наложением пакеров, защитой от отладчиков или защитой на уровне ядра.

Обфускация кода – метод основан на мутации кода, при которой дизассемблер не сможет прочитать его. Существует алгоритмы обфускации Колберга и Chenxi Wang’s, а также несколько видов обфускации. К видам обфускации относятся: лексическая обфускация, обфускация структур данных. Лексическая обфускация является самым простым методом мутации кода, так как метод основывается на добавлении в код программы лишних ни на что не влияющих строк кода возможно изменение расположения частей программы, удаление комментариев или замена их. Обфускация структур данных трансформирует структуры данных программы. Данная обфускация также делится на три вида: обфускация хранения, соединения и переупорядочивания.

Обфускация хранения заключается в трансформации хранилищ данных, а также самих типов данных (например, создание и использование необычных типов данных). Первым методом осуществления данного типа обфускации будет изменение интерпретации данных определённого типа. Как известно сохранение, каких-либо данных в хранилищах определенного типа в процессе работы программы, очень распространенное явление. Например, для

перемещения по элементам массива очень часто используют переменную типа «целое число», которая выступает в роле индекса. Использование в данном случае переменных иного типа возможно, но это будет не тривиально и может быть менее эффективно. Также методом может послужить изменение срока использования хранилищ данных. Большинство программ, в процессе работы, выводят различную информацию, которая чаще всего в коде программы представляются в виде статических данных таких как строки, которые позволяют визуально ориентироваться в ее коде и определять выполняемые операции. Такие строки также желательно изменить, это можно сделать, просто записывая каждый символ строки, используя его код (к примеру, вместо символа «A» можно записать как 16-ричное число «0x41»).

Обфускация соединения запутывает соединения структур данных, содержащихся в коде программы, что усложняет воссоздание алгоритма работы программы.

Обфускация переупорядочивания заключается в изменении последовательности объявления переменных, внутреннего расположения хранилищ данных, а также переупорядочивании методов, массивов (использование нетривиального представления многомерных массивов) определенных полей в структурах и так далее.

Наложение пакера или упаковщик шифрует код программы путем упаковки его кода. Когда происходит запуск упакованной программы, код программы расшифровывается, и она исполняется. Процесс расшифровки начинается перед запуском основной функции программы.

Защита от отладчика суть метода состоит в том, что при подключении какого-либо отладчика к программе, она будет знать, что сейчас происходит исполнение не в обычном режиме, а в режиме отладки.

Защита по времени исполнения кода основывается на том, что на некотором участке кода фиксируется его время исполнения, в случае подключения отладчика, особенно в режиме пошаговой отладки время, в котором подсчитывается, насколько быстро происходит выполнение кода, будет идти и если оно превысит установленное значение, то программа поймет, что сейчас подключен отладчик.

Защита от отладчиков с использованием ловушек основывается на ловушках, при попадании в которые выявляется присутствие отладчика. Первый способ постановки ловушки помогает выяснить реальное содержимое трассировочного файла в условиях эмуляции его отладчиком. Ловушка основывается на реализации аппаратной особенности процессора Intel вследствие чего после исполнения инструкции pop ss прерывание int 1 не может быть вызвано, и отладчик «не замечает» и пропускает следующую за данной командой инструкцию. Таким образом, следующая за pop ss инструкция исполняется на реальном процессоре, а не под отладкой. Второй способ постановки ловушки перехват в защищаемой программе прерываний int 1 и использование его для собственных нужд либо для ссылки на механизм защиты. Тогда при попытке отладки программы она будет либо некорректно работать, либо будет передавать управление модулю защиты, который, например будет выводить сообщение о невозможности продолжения работы.

Использование защиты в режиме ядра позволяет заблокировать доступ на уровне пользователя к адресному пространству программы, а также подключения к нему различных отладчиков, а также чтение и запись адресного пространство программы. Этот метод реализуется при помощи некоторого драйвера, работа которого сравнима с антивирусом, так как его задача сводится к сканированию запущенных процессов и, если он обнаруживает запрос на подключение к адресному пространству нашего приложения он это блокирует. Просмотр драйвером запросов осуществляется путем перехвата WinApi вызовов. WinApi вызовы организовывают общение между пользовательским уровнем и уровнем ядра и их просмотр возможен только находясь в нулевом кольце операционной системе Windows.

Контроль целостности кода – еще один метод защиты приложений от взлома. Этот метод основывается на том, что при загрузке какой-либо сторонней динамической библиотеки или

же изменение кодовых инструкций нарушается целостность кода. Целостность кода можно обеспечить с помощью: циклично-избыточного кода и хэш суммы.

Таблица 1

Сравнительная таблица ПО и методов защиты.

Методы защиты / ПО	Защита от отладчика	Обфускация	Наложение пакера	Использование защиты в режиме ядра
CheatEngine	+	-	-	+
IDA	+-	+	+	+
OlyDbg	+-	+	+	+
Ghidra	+-	+	+	+

«+» – Метод полностью защищает информацию от приложения для взлома.
«+-» – Метод частично защищает информацию от приложения для взлома.
«-» – Метод не защищает информацию от приложения для взлома.

ЗАКЛЮЧЕНИЕ

Наиболее высокий уровень защиты обеспечивает защита в режиме ядра. Защита, работающая в ядре, блокирует запросы на чтение и запись в ОЗУ, а также подключение отладочного ПО, что не даст возможность взломать приложение.

Авторы выражает благодарность кандидату технических наук В. Е. Кладову за высказанные замечания и пожелания по улучшению статьи.

СПИСОК ЛИТЕРАТУРЫ

1. **Юричев Д.** Reverse Engineering для начинающих. 2017. 1045 с.

2. **Лин В.** PDP-11 и VAX-11. Архитектура ЭВМ и программирование на языке ассемблера / В. Лин; Пер. с англ. В. М. Северьянов. М.: Радио и связь, 1989. 315, [1] с.

ОБ АВТОРАХ

ДАВЛЕТШИН Артур Вадимович, студент направления ИВТ каф. АСУ.

ЖУРАВЛЕВ Алексей Александрович, студент направления ИВТ каф. АСУ.

METADATA

Title: Overview of Information Protection Methods Against Reverse Engineering

Authors: A. V. Davletshin[1], A. A. Zhuravlev[2]

Affiliation:

[1-2] Ufa University of Science and Technology (UUST), Russia

Email: [1]artur.davletshin.v@yandex.ru, [2]zhuravlevaao@mail.ru.

Language: Russian.

Source: Molodezhnyj Vestnik UGATU (scientific journal of Ufa University of Science and Technology), no. 2 (31), pp. 59-63, 2024. ISSN 2225-9309 (Print).

Abstract: This article examines the process of reverse engineering as a software analysis method that enables understanding of a program's structure, the programming language used, and reconstruction of source code. Emphasis is placed on tools like disassemblers (IDA, OlyDbg, Ghidra) and debuggers, which allow for the investigation and modification of application code. Different programming language types (compiled, interpreted, and byte-code) and hacking methods, including disassembly, code modification, and hook implementation, are discussed. Additionally, the article reviews protection techniques against reverse engineering, such as obfuscation, packing, anti-debugging measures, and code integrity checks, which are essential for ensuring software security and intellectual property protection.

Key words: Reverse engineering, disassembler, debugger, hacking, obfuscation, protection, code, packers, integrity, WinAPI

About authors:

DAVLETSHIN, Artur Vadimovich – Student, Information Technology (IT) program, Department of Automated Systems, UUST.

ZHURAVLEV, Alexey Alexandrovich – Student, Information Technology (IT) program, Department of Automated Systems, UUST.