МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Защита программ от обратной разработки

АВТОРЕФЕРАТ
дипломной работы

студента 6 курса 631 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Хорева Михаила Олеговича

Научный руководитель
доцент _____ И. Ю. Юрин
22.01.2022 г.

Заведующий кафедрой
д. ф.-м. н., доцент _____ М. Б. Абросимов
22.01.2022 г.

Саратов 2022

ВВЕДЕНИЕ

В условиях активного развития компьютерных технологий и сети Интернет, рынок программного обеспечения растёт с огромной скоростью каждый год. Разрабатываются новые программные продукты для решения самых разнообразных задач. И в основе каждого из этих продуктов лежит интеллектуальная собственность программиста или компании. Поэтому желание обезопасить результаты своей работы от обратной разработки с целью копирования, модификации или взлома программ является естественной потребностью. Решить её могли бы полностью гомоморфные системы шифрования, которые предоставляют возможность исполнять программы без их предварительного дешифрования, но они всё ещё находятся на стадии разработки, а скорость работы существующих прототипов остается крайне низкой.

Как положительным, так и негативным можно считать тот факт, что область проектирование систем защиты от обратной разработки тесно связана с деструктивным программным обеспечением. Некоторые подходы и алгоритмы были изначально позаимствованы именно в результате изучения образцов вредоносных программ. Верно и обратное, наиболее удачные системы, разработанные для легального программного обеспечения, применяются для защиты деструктивных программ.

Существует множество методов, реализующих защиту программ от обратной разработки, но одним из наиболее перспективных на данный момент является HARES.

Целью данной работы является рассмотрение существующих методов защиты программ от обратной разработки, а также программная реализация метода HARES для 32 битных исполняемых файлов формата PE-EXE.

Задачи данной работы:

- Рассмотреть основные методы анализа исполняемых файлов, применяемые для обратной разработки;

- Проанализировать существующие программы, реализующие методы анализа исполняемых файлов;
- Рассмотреть основные методы защиты программного обеспечения от обратной разработки;
- Изучить метод защиты программ от обратной разработки HARES;
- Разработать собственную программу, реализующую данный метод.

Дипломная работа состоит из введения, 4 разделов, заключения, списка использованных источников и 3 приложений. Общий объем работы – 82 страницы, из них 35 страниц – основное содержание, включая 31 рисунок и 1 таблицу, список использованных источников из 14 наименований.

КРАТКОЕ СОДЕРЖАНИЕ

1 Обратная разработка

Обратная разработка – это исследование программного обеспечения с целью установления архитектуры, принципов и особенностей его работы. На сегодняшний день обратная разработка наиболее сконцентрирована на решении следующих задач:

1. Определение используемых в программе систем защиты;
2. Восстановление исходного алгоритма критически важных функций программы;
3. Выявление уязвимостей;
4. Поиск недокументированных возможностей.

Для решения этих задач используют следующие подходы и методики:

1. Статический анализ:
    a. Базовый статический анализ;
    b. Дизассемблирование.
2. Динамический анализ:
    a. Базовый динамический анализ;
    b. Отладка.

Базовый статический анализ заключается в исследовании всей информации, которую можно извлечь из исполняемого файла, не рассматривая его кодовые инструкции. К такой информации можно отнести:

К такой информации можно отнести:

- Метаинформацию о файле;
- Структуру исполняемого файла;
- Строковые данные;
- Несоответствия и отклонения от формата файла;
- Таблицы экспорта и импорта;
- Зависимости;
- Информацию, записанную в файл компилятором;

- И др.

В рамках данной методики можно выделить 2 основных метода:

- Анализ совокупности всей полученной из программы информации;
- Сравнение извлеченной информации с данными, полученными в рамках прошлых исследований или из открытых источников.

В данном разделе приводятся описание следующих программ, реализующих методы статического анализа:

- PE Explorer;
- PEiD;
- Resource Hacker;
- Maltego.

Дизассемблирование – это процесс преобразования двоичных машинных инструкций в код на языке ассемблера. Заметим, что ассемблерный код является самым высокоуровневым кодом, который можно восстановить, при отсутствии доступа к исходным текстам на языке более высокого уровня.

В данном разделе приводится описание программы IDA Pro – современного стандарта в области дизассемблирования.

Базовый динамический анализ заключается в исследовании изменений и взаимодействий, происходящих в контролируемой среде, в результате запуска программы.

В данном разделе приводятся описание следующих программ, применяемых для проведения базового динамического анализа:

- Process Monitor;
- Process Explorer;
- Regshot;
- INetSim;
- Wireshark;

Отладка – это процесс исследования программы путем создания динамического представления её кода во время выполнения. Существуют два

основных подхода к отладке программ. Во-первых, можно запустить программу при помощи отладчика. Тогда, после загрузки в память, она немедленно остановит свою работу, не переходя к точке входа. На этом моменте у нас будет полный контроль над её дальнейшим выполнением. Во-вторых, отладчик может быть подключен к программе непосредственно во время её работы. Все потоки программы будут остановлены, и будут готовы к отладке. Для прерывания исполнения программы используются точки останова. Они могут быть разделены на 2 типа:

- Программные;
- Аппаратные.

Программные точки останова прерывают исполнение программы при переходе к выбранным инструкциям или выполнении заданных условий. Аппаратные точки останова, предусмотренные архитектурой x86, используют для своей работы специальные регистры.

В данном разделе приводится описание программы OllyDbg – самого популярного отладчика на сегодняшний день.

Наилучшего результата в обратной разработке можно добиться, только последовательно используя каждую из представленных методик в указанном порядке, так как информация, полученная от каждой их них, может значительно упростить и ускорить проведение дальнейшего анализа.

2 Методы противодействия обратной разработке

Обфускация – это процесс приведения исполняемого файла к виду, сохраняющему его функциональные возможности, но затрудняющему анализ, дизассемблирование и понимание алгоритмов его работы. Основной метод обфускации исполняемых файлов – преобразование потока управления. Преобразование потока управления – это метод, который изменяет порядок выполнения программы таким образом, чтобы снизить ее удобочитаемость. Данный метод состоит из трех подходов:

1. Преобразования вычислений;

2. Преобразования агрегирования;

3. Преобразования упорядочивания.

Обфускация исполняемых файлов намного более эффективна, чем обфускация исходного кода, так как в первом случае отсутствует дополнительный шаг «противостояния» с компилятором. Современные компиляторы хорошо оптимизируют исходный код непосредственно перед его трансляцией, тем самым, все усилия по его обфускации могут быть напрасными.

Применение шифрования для защиты программ осуществляется следующими двумя подходами:

1. Весь код расшифровывается перед началом выполнения. После окончания работы программы, код будет снова зашифрован.

2. Зашифрованы лишь некоторые части программы, которые динамически дешифруются во время её выполнения, прямо перед тем, как к ним переходит поток управления.

Основной слабостью при использовании шифрования является секретный ключ, который, как правило, либо записан в самой программе, либо генерируется во время её выполнения. Таким образом для проведения обратной разработки необходимо только найти этот секретный ключ.

Антиотладка – это методика противодействия обратной разработке, заключающаяся в обнаружении самой программой того факта, что она выполняется под управлением отладчика. Выявить присутствие отладчика можно при помощи следующих функций и признаков, указывающих на его работу:

- *IsDebuggerPresent;*
- Сверка времени;
- *NtGlobalFlag;*
- *Process Heap;*
- *CheckRemoteDebuggerPresent;*
- *TLS Callbacks;*

- SeDebugPrivilege;
- *NtSetInformationThread;*

3 HARES

HARES – это метод противодействия обратной разработке программного обеспечения, разработанный Якобом Торри и представленный на симпозиуме по безопасности SyScan в Сингапуре в 2015 году. Основная идея метода заключается в комбинации следующих трех технологий для создания системы комплексного противодействия обратной разработке:

- Кольца привилегий;
- Дешифрование на уровне процессора;
- Десинхронизация TLB.

После запуска приложения, защищенного HARES, отправляется сообщение драйверу ядра, заблокировать страницы программы в невыгружаемой памяти. Это необходимо, поскольку операционная система будет выгружать неиспользуемые страницы памяти на диск, чтобы освободить оперативную память для других процессов. Следующим шагом станет использование десинхронизации TLB и дешифрования. Зашифрованные страницы будут расшифрованы процессором, используя секретный ключ с отладочных регистров, и благодаря десинхронизации TLB, они будут записаны в секцию памяти только для исполнения (execute-only), следовательно будут доступны только для дальнейшего исполнения содержащихся в этих страницах команд, всем остальным процессам, включая исходную программу, будет доступны только страницы с зашифрованным кодом, предоставляемые TLB. Во время завершения программы драйвер ядра так же, как и при запуске программы, отправляет обратный вызов HARES, который перезаписывает расшифрованные страницы кода в буфере нулями.

Основным недостатком метода, на данный момент, является как сложность безопасной доставки секретного ключа в отладочные регистры процессора, так и его замена без перезагрузки системы. Основной

рекомендацией по минимизации рисков компрометации ключа является его загрузка при старте операционной системы, когда число активных процессов в памяти минимально.

4 Программная реализация

В ходе реализации практической части были написаны две программы – программа, реализующая AES шифрование секций кода 32-битных исполняемых файлов формата PE-EXE. Реализация данной программы представлена в Приложении А.

Программная реализация метода HARES была выполнена на основе гипервизора MoRE, с включением в него подходов к дешифрованию, использованных в TRESOR. Реализация данной программы представлена в Приложении Б.

Было произведено тестирование программ, защищённых методом HARES, для анализа его влияния на их быстродействие. В качестве тестовой программы была использована GracefulTrees.exe. Результаты проведённого исследования можно свести в следующую таблицу (таблица 1).

Таблица 1 – Результаты тестирования программы

n	Время работы (мс)	Время работы программы защищенной методом HARES (мс)	Снижение быстродействия (%)
1	40	66	65,0
2	55	81	47,2
3	65	94	44,6
4	74	100	35,1
5	77	104	35,0
6	92	119	29,3
7	149	177	18,7
8	360	394	9,4
9	2112	2190	3,6
10	18648	19152	2,7
11	529332	543012	2,5
12	10940068	11222226	2,5

Большое снижение быстродействия для запусков с небольшим временем работы можно объяснить тем, что использование гипервизора в реализации

программы привносит дополнительное время работы, необходимое на его подготовку. Можно заметить, что при увеличении времени работы снижение быстродействия становится постоянным, на уровне 2,5 %. В открытом доступе нет программ, реализующих метод HARES, поэтому сравнить её с аналогами не удалось.

Так же была рассмотрена атака, основанная на снятии образа оперативной памяти, к которой данный метод уязвим.

ЗАКЛЮЧЕНИЕ

В современном мире защита программ от обратной разработки является одним из наиболее востребованных направлений в области информационной безопасности.

Защита интеллектуальной собственности, к которой относится исходный код программного обеспечения, одна из самых востребованных сфер применения данного направления.

Активное развитие методов обратной разработки, стимулируемое потребностью в анализе вредоносных программ, приводит к быстрой потере эффективности большинства известных методов защиты. Необходим комплексный подход к обеспечению защиты программ от обратной разработки.

В данной работе были рассмотрены основные методы анализа исполняемых файлов, применяемые для обратной разработки, проанализированы существующие программы, реализующие методы анализа исполняемых файлов, рассмотрены основные методы защиты программного обеспечения от обратной разработки, изучен метод защиты от обратной разработки HARES, а также была произведена его программная реализация для 32 битных исполняемых файлов формата PE-EXE.