**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Гжельский государственный университет»** (ГГУ)

Колледж ГГУ

Специальность 09.02.07. Информационные системы и программирование.

**Реферат**

**по предмету: «МАПО»**

**на тему: «Дизассемблер».**

ВЫПОЛНИЛ:

Студент группы ИСП-О-17

Попов Т.А

ПРОВЕРИЛА:

Прокуронова А. Ю.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

п. Электроизолятор

2019 г.

**Дизассе́мблер** (от англ. *disassembler* [ˌdɪsəˈsɛmblə(r)]) — транслятор, преобразующий машинный код, объектный файлили библиотечные модули в текст программы на языке ассемблера.

По режиму работы с пользователем делятся на автоматические и интерактивные. Примером автоматических дизассемблеров может служить Sourcer[[1]](https://www.wikiwand.com/ru/%D0%94%D0%B8%D0%B7%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80" \l "citenote1). Такие дизассемблеры генерируют готовый листинг, который можно затем править в текстовом редакторе. Пример интерактивного — IDA. Он позволяет изменять правила дизассемблирования и является весьма удобным инструментом для исследования программ.

Дизассемблеры бывают однопроходные и многопроходные. Основная трудность при работе дизассемблера — отличить данные от машинного кода, поэтому на первых проходах автоматически или интерактивно собирается информация о границах процедур и функций, а на последнем проходе формируется итоговый листинг. Интерактивность позволяет улучшить этот процесс, так как просматривая дамп дизассемблируемой области памяти, программист может сразу выделить строковые константы, дать содержательные имена известным точкам входа, прокомментировать разобранные им фрагменты программы. Обычно однопроходный дизассемблер (как и построчный ассемблер) является составной частью отладчика.

Чаще всего дизассемблер используют для анализа программы (или её части), исходный текст которой неизвестен — с целью модификации, копирования или взлома. Реже — для поиска ошибок (багов) в программах и компиляторах, а также для анализа и оптимизации создаваемого компилятором машинного кода.

При работе с исполняемым кодом или байт-кодом, созданным на некоторых языках высокого уровня (например, java) имеется возможность восстановить не только текст на языке ассемблера, но даже и структуру классов программы, а если при компиляции исполняемого файла не была отключена отладочная информация — то и исходный текст программы. Для исключения таких возможностей используется обфусфикация.

## Примеры программ-дизассемблеров

• IDA

• Sourcer

• Beye

• HT editor

• Hacker Disassembler Engine

• CADt

• Objdump

• Radare2

• Ghidra[en]

Изучение логики работы программы может выполняться в одном из двух режимов: статическом и динамическом. Сущность статического режима заключается в изучении исходного текста программы. Для получения листингов исходного текста выполняемый программный модуль дизассемблируют, то есть получают из программы на машинном языке программу на языке Ассемблер.

Динамический режим изучения алгоритма программы предполагает выполнение трассировки программы. Под трассировкой программы понимается выполнение программы на ЭВМ с использованием специальных средств, позволяющих выполнять программу в пошаговом режиме, получать доступ к регистрам, областям памяти, производить остановку программы по определенным адресам и т. д. В динамическом режиме изучение алгоритма работы программы осуществляется либо в процессе трассировки, либо по данным трассировки, которые записаны в запоминающем устройстве.

Средства противодействия дизассемблированию не могут защитить программу от трассировки и наоборот: программы, защищенные только от трассировки, могут быть дизассемблированы. Поэтому для защиты программ от изучения необходимо иметь средства противодействия как дизассемблированию, так и трассировке.

Существует несколько методов противодействия дизассемблированию:

- шифрование;

- архивация;

- использование самогенерирующих кодов;

- «обман» дизассемблера.

Архивацию можно рассматривать как простейшее шифрование. Причем архивация может быть объединена с шифрованием. Комбинация таких методов

позволяет получать надежно закрытые компактные программы. Зашифрованную программу невозможно дизассемблировать без расшифрования. Зашифрование (расшифрование) программ может осуществляться аппаратными средствами или отдельными программами. Такое шифрование используется перед передачей программы по каналам связи или при хранении ее на ВЗУ. Дизассемблирование программ в этом случае возможно только при получении доступа к расшифрованной программе, находящейся в ОП перед ее выполнением (если считается, что преодолеть криптографическую защиту невозможно).

Другой подход к защите от дизассемблирования связан с совмещением процесса расшифрования с процессом выполнения программ. Если расшифрование всей программы осуществляется блоком, получающим управление первым, то такую программу расшифровать довольно просто. Гораздо сложнее расшифровать и дизассемблировать программу, которая поэтапно расшифровывает информацию, и этапы разнесены по ходу выполнения программы. Задача становится еще более сложной, если процесс расшифрования разнесен по тексту программы.

Сущность метода, основанного на использовании самогенерируемых кодов, заключается в том, что исполняемые коды программы получаются самой программой в процессе ее выполнения. Самогенерируемые коды получаются в результате определенных действий над специально выбранным массивом данных. В качестве исходных данных могут использоваться исполняемые коды самой программы или специально подготовленный массив данных. Данный метод показал свою высокую эффективность, но он сложен в реализации.

Под «обманом» дизассемблера понимают такой стиль программирования, который вызывает нарушение правильной работы стандартного дизассемблера за счет нестандартных приемов использования отдельных команд, нарушения общепринятых соглашений. «Обман» дизассемблера осуществляется несколькими способами:

- нестандартная структура программы;

- скрытые переходы, вызовы процедур, возвраты из них и из прерываний;

- переходы и вызовы подпрограмм по динамически изменяемым адресам;

- модификация исполняемых кодов.

Для дезориентации дизассемблера часто используются скрытые переходы, вызовы и возвраты за счет применения нестандартных возможностей команд.

Маскировка скрытых действий часто осуществляется с применением стеков.

Трассировка программ обычно осуществляется с помощью программных продуктов, называемых отладчиками. Основное назначение их - выявление ошибок в программах. При анализе алгоритмов программ используются такие возможности отладчиков как пошаговое (покомандное) выполнение программ, возможность останова в контрольной точке.

Покомандное выполнение осуществляется процессором при установке пошагового режима работы. Контрольной точкой называют любое место в программе, на котором обычное выполнение программы приостанавливается, и осуществляется переход в особый режим, например, в режим покомандного выполнения. Для реализации механизма контрольной точки обычно используется прерывание по соответствующей команде ЭВМ (для IВМ-совместных ПЭВМ такой командой является команда INТ). В современных процессорах можно использовать специальные регистры для установки нескольких контрольных точек при выполнении определенных операций: обращение к участку памяти, изменение участка памяти, выборка по определенному адресу, обращение к определенному порту ввода-вывода и т. д.

При наличии современных средств отладки программ полностью исключить возможность изучения алгоритма программы невозможно, но существенно затруднить трассировку возможно. Основной задачей противодействия трассировке является увеличение числа и сложности ручных операций, которые необходимо выполнить программисту-аналитику.

Для противодействия трассировке программы в ее состав вводятся следующие механизмы:

- изменение среды функционирования;

- модификация кодов программы;

- «случайные» переходы.

Под изменением среды функционирования понимается запрет или

переопределение прерываний (если это возможно), изменение режимов работы, состояния управляющих регистров, триггеров и т. п. Такие изменения вынуждают аналитика отслеживать изменения и вручную восстанавливать среду функционирования.

Изменяющиеся коды программ, например, в процедурах приводят к тому, что каждое выполнение процедуры выполняется по различным ветвям алгоритма.

«Случайные» переходы выполняются за счет вычисления адресов переходов. Исходными данными для этого служат характеристики среды функционирования, контрольные суммы процедур (модифицируемых) и т. п. Включение таких механизмов в текст программ существенно усложняет изучение алгоритмов программ путем их трассировки.

***Методы защиты программ от исследования***

*дизассемблирование программа язык листинг*

Для защиты программ от исследования необходимо применять методы защиты от исследования файла с ее исполняемым кодом, хранящемся на внешнем носителе, а также методы защиты исполняемого кода, загружаемого в оперативную память для выполнения этой программы.

*В первом случае защита может быть основана на шифровании секретной части программы, а во втором*- на блокировании доступа к исполняемому коду программы в оперативной памяти со стороны отладчиков. Кроме того, перед завершением работы защищаемой программы должен обнуляться весь ее код в оперативной памяти. Это предотвратит возможность несанкционированного копирования из оперативной памяти дешифрованного исполняемого кода после выполнения защищаемой программы.

Таким образом, защищаемая от исследования программа должна включать следующие компоненты:

- инициализатор;

- зашифрованную секретную часть;

- деструктор (деициниализатор).

Инициализатор должен обеспечивать выполнение следующих функций:

- сохранение параметров операционной среды функционирования (векторов прерываний, содержимого регистров процессора и т.д.);

- запрет всех внутренних и внешних прерываний, обработка которых не может быть запротоколирована в защищаемой программе;

- загрузка в оперативную память и дешифрование кода секретной части программы;

- передача управления секретной части программы.

Секретная часть программы предназначена для выполнения основных

целевых функций программы и защищается шифрованием для предупреждения внесения в нее программной закладки.

Деструктор после выполнения секретной части программы должен выполнить следующие действия:

- обнуление секретного кода программы в оперативной памяти;

- восстановление параметров операционной системы (векторов прерываний, содержимого регистров процессора и т.д.), которые были установлены до запрета неконтролируемых прерываний;

- выполнение операций, которые невозможно было выполнить при запрете неконтролируемых прерываний;

- освобождение всех незадействованных ресурсов компьютера и завершение работы программы.

Для большей надежности инициализатор может быть частично зашифрован и по мере выполнения может дешифровать сам себя. Дешифроваться по мере выполнения может и секретная часть программы. Такое дешифрование называется динамическим дешифрованием исполняемого кода. В этом случае очередные участки программ перед непосредственным исполнением расшифровываются, а после исполнения сразу уничтожаются.

Для повышения эффективности защиты программ от исследования необходимо внесение в программу дополнительных функций безопасности, направленных на защиту от трассировки. К таким функциям можно отнести:

- периодический подсчет контрольной суммы области оперативной памяти, занимаемой защищаемым исходным кодом; сравнение текущей контрольной суммы с предварительно сформированной эталонной и принятие необходимых мер в случае несовпадения;

- проверку количества занимаемой защищаемой программой оперативной памяти; сравнение с объемом, к которому программа адаптирована, и принятие необходимых мер в случае несоответствия;

- контроль времени выполнения отдельных частей программы;

- блокировку клавиатуры на время отработки особо секретных алгоритмов.

Для защиты программ от исследования с помощью дизассемблеров можно использовать и такой способ, как усложнение структуры самой программы с целью запутывания злоумышленника, который дизассемблирует эту программу. Например, можно использовать разные сегменты адреса для обращения к одной и той же области памяти. В этом случае злоумышленнику будет трудно догадаться, что на самом деле программа работает с одной и той же областью памяти.