**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ (ИТТСУ)**

**Кафедра «Управление и защита информации»**

**Инструменты реверс-инжиниринга**

**Учебное пособие**

**Москва – 2023**

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ (ИТТСУ)**

**Кафедра «Управление и защита информации»**

**Инструменты реверс-инжиниринга**

**Учебное пособие**

**по направлению подготовки**

**10.05.01 «Компьютерная безопасность»**

**Москва – 2023**

УДК [004.056](https://www.teacode.com/online/udc/00/004.91.html).53

И 72

Инструменты реверс-инжиниринга: Учебное пособие/ Домашкин А.Д. Костенко С.А., Николаев А.М., Савицкий Д.Д., Сидоренко В.Г., Часников А.А. – РУТ(МИИТ).2023. – 53 с.

В учебном пособии излагаются примеры применения автоматизированного анализа типовых структур исполняемых файлов на основе реверс-инжиниринга для решения прикладных задач. Использование представленных в учебном пособии средств автоматизации реинжиниринга повышает производительность труда специалистов по защите информации, решающих задачи автоматического поиска уязвимостей программного обеспечения.

Учебное пособие предназначено для обучающихся по специальности 10.05.01 «Компьютерная безопасность», а также других направлений, связанных с информационными технологиями.

Рецензенты:

Заместитель начальника отдела разработки технологических информационных систем АО «ВНИИЖТ» М.А. Кулагин

Заведующий кафедрой [«Вычислительные системы, сети и информационная безопасность»](https://rut-miit.ru/depts/29) РУТ (МИИТ) Б.В. Желенков

© РУТ (МИИТ), 2023

# Оглавление

[Введение 4](#_Toc142929692)

[1. Инструмент radare2 5](#_Toc142929693)

[1.1. Основные концепции radare2 6](#_Toc142929695)

[1.2. Установка и настройка radare2 8](#_Toc142929696)

[1.2.1. Установка на Windows 10 8](#_Toc142929697)

[1.2.1.1. Установка собранной версии 8](#_Toc142929698)

[1.2.1.2. Сборка исходного кода. 10](#_Toc142929699)

[1.2.2. Установка в операционных системах на базе ядра Linux 15](#_Toc142929700)

[1.2.2.1. Установка с помощью пакетных менеджеров 15](#_Toc142929701)

[1.2.2.2. Сборка исходного кода 17](#_Toc142929702)

[1.3. Работа с radare2 20](#_Toc142929703)

[1.3.1. Открытие и анализ файла 20](#_Toc142929704)

[1.3.2. Базовое использование 21](#_Toc142929705)

[1.3.3. Навигация по файлу 24](#_Toc142929706)

[1.4. Работа с графическим интерфейсом radare2 29](#_Toc142929707)

[1.4.1. Введение в работу с графическим интерфейсом 29](#_Toc142929708)

[1.4.2. Команды и навигация в графическом интерфейсе 30](#_Toc142929709)

[1.5. Реверс-инжиниринг с использованием radare2 34](#_Toc142929710)

[2. Инструмент IDA Pro 39](#_Toc142929711)

[2.1. Работа с графическим интерфейсом IDA Pro 40](#_Toc142929712)

[2.2. Реверс-инжиниринг с использованием IDA Pro 45](#_Toc142929713)

[Заключение 50](#_Toc142929714)

[Список литературы 52](#_Toc142929715)

# Введение

Обратная разработка (англ. Reverse Engineering) – метод исследования устройств или программного обеспечения с целью понять принцип его работы или обнаружить недокументированные возможности. В информационной безопасности (ИБ) она играет значительную роль. Используя ее, специалисты в области ИБ могут исследовать вредоносные приложения, разбираться, как они работают, например, для последующего составления сигнатур в базах антивирусов и защиты других пользователей от возможной цифровой угрозы.

Выделяют 4 методики проведения обратной разработки:

* анализ обмена данными приложения с помощью различных анализаторов трафика;
* использование режима отладки для поиска нужных участков кода и просмотра данных, с которыми работает приложение;
* дизассемблирование машинного кода программы (изучение требует довольно много времени);
* декомпиляция кода программы для создания исходного кода программы на языке программирования высокого уровня.

Тема дизассемблирования машинного кода неразрывно связана с обратной разработкой, но у неспециалиста может вызвать затруднения. Связано это с необходимым уровнем подготовки и временем, которое затрачивается на "разборку". Тем не менее, каждый случай может оказаться довольно увлекательным "путешествием" в недра приложения.

Данное учебное пособие знакомит читателей с инструментарием, используемым при реверс-инжиниринге и позволит поэтапно погрузиться внутрь исполняемого файла, чтобы показать, что дизассемблирование – это совсем нестрашно.

# Инструмент Radare2

Radare2 – это свободно-распространяемое программное обеспечение (ПО) с открытым исходным кодом для обратной разработки, предоставляющее широкий спектр функций для анализа бинарных файлов, включая дизассемблирование, отладку, анализ кода и многое другое. Radare2 разработан с учетом модульности и расширяемости, что делает его мощным инструментом для исследователей в области безопасности и разработчиков.

Radare2 – полноценная экосистема, включающая в себя различные утилиты и библиотеки, каждая из которых специализируется на определенной задаче. Например, в состав radare2 входят следующие утилиты:

* rabin2, использующаяся для извлечения метаданных из бинарных файлов (импортируемых функций, экспортируемых символов, секций, подключаемых библиотек и прочего);
* radiff2, позволяющая сравнивать два бинарных файла;
* rahash2, вычисляющая хеш-суммы и другие криптографические значения;
* rafind2, реализующая поиск как строк, в том числе, с помощью регулярных выражений, так и данных в шестнадцатеричном формате или по двоичному шаблону;
* Rax2, конвертирующая данные в различных форматах
* др. модули.

В этом учебном пособии рассмотрим основные компоненты radare2, процесс его установки на операционные системы Windows 10 и на базе ядра Linux, рассмотрим основные команды и функции, необходимые для работы с командной строкой, а также использование графического интерфейса iaito. В разделе 1.5 представлен пример применения radare2 для обратной разработки, анализа и изучения файлов.

# Основные концепции radare2

Radare2 – полнофункциональный инструмент командной строки. Проект начинался как простой шестнадцатеричный редактор, ориентированный на криминалистический анализ (форензику), но со временем его функциональные возможности значительно расширились. Благодаря поддержке широкого спектра архитектур, radare2 позволяет анализировать, эмулировать, отлаживать, изменять и разбирать любой файл.

Перед непосредственной установкой и дальнейшим обзором возможностей radare2 познакомимся с некоторыми основными концепциями его работы.

**Блоки**

В radare2 все данные читаются и записываются в "блоки". Это основная единица данных, с которой работает данный инструмент.

**Отладка**

Radare2 содержит мощный функционал для отладки программ, включая поддержку различных архитектур и операционных систем.

**Визуализация**

Radare2 предоставляет различные инструменты для визуализации бинарных данных, включая графы потока управления (control flow graph), hex-дампы (шестнадцатеричные дампы памяти) и многое другое.

**Команды и скрипты**

Radare2 имеет собственный язык команд, позволяющий гибко управлять анализом и отладкой. Команды можно объединять в скрипты для автоматизации задач.

**Шелл**

Radare2 предоставляет интерактивный шелл для выполнения команд и скриптов.

**Режимы работы**

Анализ – используется для изучения поведения и структуры файла.

Сравнение – используется для сравнения двоичных файлов для нахождения различий и сходств.

Отладка – используется для отладки. Позволяет устанавливать точки остановки, выполнять код пошагово, наблюдать за регистрами и памятью.

# Установка и настройка radare2

Официальный сайт проекта, на котором можно найти всю необходимую информацию, включая документацию, ссылки на графический интерфейс и многое другое – <https://www.radare.org/n/>. Исходный код и собранные файлы radare2 можно найти на github – <https://github.com/radareorg/radare2>.

Порядок установки зависит от операционной системы, а подробные инструкции расположены в официальном репозитории на github [1]. Процессы инсталляции для операционных систем Windows 10 и на базе ядра Linux представлены в разделах 2.1 и 2.2 соответственно.

## Установка на Windows 10

Для операционной системы Windows 10 рассмотрим 2 варианта установки: собранную (портативную) версию и компиляцию из исходного кода.

### Установка собранной версии

Готовый к использованию инструмент radare2 для Windows 10 поставляется в портативном (portable) формате, т.е. все библиотеки, бинарные файлы и документация находятся в одном архиве. Это решает проблему с зависимостью от другого ПО и позволяет сразу запускать загружаемое ПО. Процесс состоит из следующих действий:

1. Перейдем в раздел с релизами в официальном репозитории проекта на github (<https://github.com/radareorg/radare2/releases>). Изображение ресурса представлено на рисунке 1.1.
2. Загрузим архив с названием radare2-x.x.x-w64.zip или radare2-x.x.x-w32.zip для 64 или 32-битной системы соответственно, как показано на рисунке 1.2.

|  |
| --- |
|  |
| Рисунок 1.1 – Официальный репозиторий проекта |

|  |
| --- |
|  |
| Рисунок 1.2 – Готовые файлы установки |

1. Перейдем в директорию с загруженным архивом, распакуем в удобное для пользователя место. Желательно, чтобы путь состоял только из латиницы без использования пробелов, иначе в режиме отладки и даже при запуске инструмента могут возникнуть проблемы. Пример представлен на рисунке 1.3.

Если установка собранной (портативной) версии radare2 прошла успешно, то п. 1.2.1.2 Сборка исходного кода выполнять не надо!

|  |
| --- |
|  |
| Рисунок 1.3 – Расположение программы |

### Сборка исходного кода.

Рассмотрим установку radare2 из исходных кодов.

Для начала откроем PowerShell от имени администратора. Пример представлен на рисунке 1.4.

Склонируем репозиторий в директорию на своём устройстве с помощью команды *git clone*. Процесс представлен на рисунке 1.5.

Для сборки перейдем в директорию, куда был склонирован репозиторий. Пример расположения представлен на рисунке 1.6.

Последовательно исполним 3 .bat-скрипта:

* preconfigure.bat;
* configure.bat;

|  |
| --- |
|  |
| Рисунок 1.4 – Запуск PowerShell |

|  |
| --- |
|  |
| Рисунок 1.5 – Клонирование репозитория |

* make.bat.

Процесс выполнения скриптов представлен на рисунках 1.7-1.9.

|  |
| --- |
|  |
| Рисунок 1.6 – Расположение репозитория |

|  |
| --- |
|  |
| Рисунок 1.7 – Результат выполнения preconfigure.bat |

|  |
| --- |
|  |
| Рисунок 1.8 – Результат выполнения configure.bat |

|  |
| --- |
|  |
| Рисунок 1.9 – Результат выполнения make.bat |

Собранная программа находится в директории prefix, что показано на рисунке 1.10.

## Установка в операционных системах на базе ядра Linux

Для операционных систем на базе ядра Linux рассмотрим 2 варианта установки: собранную (портативную) версию и компиляцию из исходного кода.

## Установка с помощью пакетных менеджеров

Установка в системах с пакетным менеджером snap представлена на рисунках 1.11-1.12.

|  |
| --- |
|  |
| Рисунок 1.10 – Собранная программа |

|  |
| --- |
|  |
| Рисунок .111 – Установка snap (Сверху – установка snap, снизу – основных компонентов snap) |

|  |
| --- |
|  |
| Рисунок 1.12 – Установка radare2 с помощью snap |
|  |
| Установка в системах с пакетным менеджером pacman представлена на рисунке 1.13. |
|  | |
| Рисунок 1.13 – Установка radare2 с помощью pacman | |

## Сборка исходного кода

Для сборки из исходного кода radare2 в операционных системах на базе ядра Linux перейдем в предварительно созданную директорию и склонируем репозиторий с помощью команды *git clone*. Процесс представлен на рисунке 1.14.

|  |
| --- |
|  |
| Рисунок 1.14 – Установка radare2 с помощью pacman |

Перейдем в директорию *radare2/sys* с помощью команды cd и выполнить скрипт *install.sh*. Процесс установки представлен на рисунке 1.15.

|  |
| --- |
|  |
| Рисунок 1.15 – Переход в директорию(сверху) и выполнение install.sh(снизу) |

Проверка установки представлена на рисунке 1.16.

|  |
| --- |
|  |
| Рисунок 1.16 – Запуск собранной программы |

# Работа с radare2

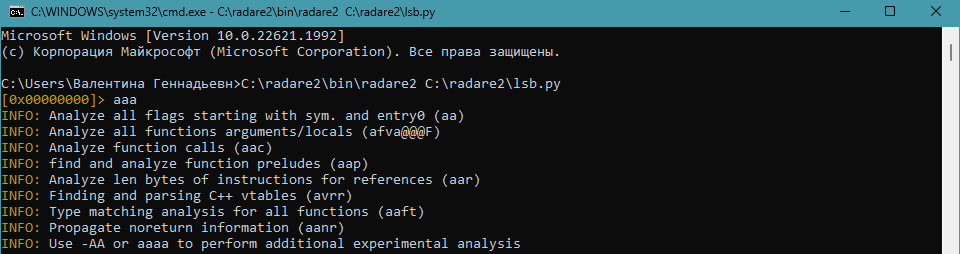
В этом разделе рассматриваются основные команды и их синтаксис.

## Открытие и анализ файла

Запуск radare2: после успешной установки radare2 осуществляем через терминал, вызываемый [сочетанием клавиш Windows + R](https://www.easeus.com/partition-master/how-to-open-command-prompt-in-windows-10.html#4), с помощью следующей команды: *radare2 <путь до файла>*, указав полный путь к файлу radare2. Процесс представлен на рисунке 1.17.

|  |
| --- |
|  |
| Рисунок 1.17 – Запуск программы radare2 |

Для начала проведем анализ файла, запустив команду *aaa* – инициирует полный анализ файла. Результат выполнения команды представлен на рисунке 1.18. Теперь можно начать использование инструмента. Чтобы увидеть результаты, надо использовать другие соответствующие команды.



|  |
| --- |
| Рисунок 1.18 – Результат выполнения команды *aaa* |

## Базовое использование

Получить информацию о файле, например, для какой архитектуры процессора он создан, дату сборки, целевую операционную систему и многое другое можно с помощью команды *i*, как показано на рисунке 1.19.

|  |
| --- |
|  |
| Рисунок 1.19 – Получение информации о файле |

Начинать исследование файла стоит с его изучения. Рассмотрим получение такой характеристики, как энтропия. Она отражает в информатике количество информации и является логарифмической функцией: . Значение энтропии файла связано со степенью его упакованности (сжатия). Большинство вредоносного ПО обладает высоким значением энтропии, то есть неупорядоченности, разнообразия, случайности. Программы, содержащие интерпретатор, обладают высоким значением этого показателя. В качестве примера можно привести собранные с помощью pyinstaller исполняемые файлы на языке python. Просмотр значений энтропии для отдельных секций бинарного файла можно выполнить с помощью команды *iS entropy* (7 столбец таблицы), что представлено на рисунке 1.20.

|  |
| --- |
|  |
| Рисунок 1.20 – Просмотр значений энтропии секций исполняемого файла |

При анализе исполняемых файлов высоким уровнем энтропии обычно считается значение, превышающее 6-7, близкое к максимально достижимому, при условии, что за элемент анализируемого ансамбля принимается 8-разрядное двоичное число. Это может указывать на наличие шифрования или сжатия в файле, что может быть признаком вредоносного ПО. Однако стоит отметить, что высокая энтропия сама по себе не является доказательством вредоносности, и требуется дополнительный анализ для подтверждения этого предположения.

Для определения значения энтропии файл, тип которого отличается от исполняемого, при работе под управлением операционных систем на базе ядра Linux можно использовать модуль rahash2 и команды -a entropy



|  |
| --- |
| Рисунок 1.21 – Просмотр значений энтропии |

## Навигация по файлу

В терминале возможно визуализировать вывод различных функций в radare2 двумя путями:

1. После выполнения полного анализа файла, используем команду *V*;

2. В случае, если нужно вывести большее количество информации по файлу, то после команды *V* нужно использовать *!*;

Примеры для первого и второго вариантов приведены рисунках 1.22 и 1.23 соответственно.

Для выхода из этого режима надо нажать клавишу Q. Для переключения между режимами используется комбинация клавиш Shift+V+V.

Выбрав второй путь, можно использовать *Backspace* для удобства использования radare2. Стрелками выбирается окно для вывода. Примеры возможностей приведены на рисунках 1.24 и 1.25.

Для навигации при выборе первого варианта визуализации нужно использовать команды *p*, *P*, *o*:

*P*, *p* – переключение между режимами;

*o* – активация режима ввода смещения, например, переход к смещению 0\*100.

Примеры вывода приведены на рисунках 1.26 и 1.27.

Для вызова команды help в radare2 используется команда *?*. Пример приведен на рисунке 1.28.

|  |
| --- |
|  |
| Рисунок 1.22 – Визуализация с использованием команды V |

|  |
| --- |
|  |
| Рисунок 1.23 – Визуализация с использованием команды ! |

|  |
| --- |
|  |
| Рисунок 1.24 – Вывод декомпиляции |

|  |
| --- |
|  |
| Рисунок 1.25 – Дизассемблирование. Выбор окон |

|  |
| --- |
|  |
| Рисунок 1.26 – Дизассемблирование |

|  |
| --- |
|  |
| Рисунок 1.27 – Код программы |

|  |
| --- |
|  |
| Рисунок 1.28 – Вывод команды help |

# Работа с графическим интерфейсом radare2

В данном разделе рассматривается работа с официальным графическим интерфейсом Iaito [2].

## Введение в работу с графическим интерфейсом

При работе в операционных системах на базе ядра Linux возможно использование официальной графической оболочки iaito. Установка и запуск интерфейса проводится с использованием команды sudo. При запуске iaito появится окно приложения. Его интерфейс приведен на рисунке 1.29. Выбираем файл для начала реверс-инжиниринга. Пример выбранного файла показан на рисунке 1.30.

|  |
| --- |
|  |
| Рисунок 1.29 – Интерфейс Iaito |

|  |
| --- |
|  |
| Рисунок 1.30 – Выбор файла .cpp |

После ввода команды *Start*, Iaito предоставит выбор способа анализа. Используем для начала работы параметры по умолчанию.

## Команды и навигация в графическом интерфейсе

В начале будет выводиться общая информация о файле:

1. **Архитектура и битность**: x86, ARM, MIPS, 32-бит, 64-бит и т.д.
2. **Операционная система**: Linux, Windows, macOS и т.д.
3. **Формат файла**: ELF, PE, Mach-O и т.д.
4. **Адрес начала**: Адрес, с которого начинается исполнение файла.
5. **Размер файла**: Общий размер файла в байтах.
6. **Секции**: Список всех секций в файле, включая их размеры и права доступа.
7. **Символы и импорты**: Список всех символов и импортированных функций в файле.
8. **Строки**: Список всех обнаруженных строк в файле.
9. **Энтропия**: Энтропия файла может быть использована для определения степени его сжатия или шифрования.

Пример приведен на рисунке 1.31.

|  |
| --- |
|  |
| Рисунок 1.31 – Общая информация о файле |

Iaito поддерживает несколько декомпиляторов, включая Ghidra и r2dec. Декомпиляция обычно используется в процессах обратного инжиниринга для анализа того, как работает определенное ПО, особенно когда исходный код не доступен. Это может быть полезно для исследования безопасности, отладки, восстановления исходного кода. Пример приведен на рисунке 1.32.

|  |
| --- |
|  |
| Рисунок 1.32 – Граф дизассемблированного файла |

Дизассемблирование – процесс преобразования машинного кода (бинарных инструкций, которые компьютер может напрямую исполнять) обратно в ассемблерный код. Это обычно делается для анализа того, как работает конкретная программа на уровне машинного кода, особенно когда исходный код недоступен. Пример дизассемблирования приведен на рисунке 1.33.

|  |
| --- |
|  |
| Рисунок 1.33 – Дизассемблирование |

Можно просмотреть шестнадцатеричный дамп. Это представление данных в компьютере, которое отображает содержимое файла или области памяти в шестнадцатеричном формате. Пример предоставлен на рисунке 1.34.

|  |
| --- |
|  |
| Рисунок 1.34 – Hex dump |

# Реверс-инжиниринг с использованием radare2

В качестве примера рассмотрим простейший файл crackme (исполняемые файлы для обучения в обратной разработке), в которых необходимо обойти проверку входных данных, например, пароля (bypass password) [3].

Перейдем в директорию, в которую предварительно было помещены файлы для обучения, что представлено на рисунке 1.35.

|  |
| --- |
|  |
| Рисунок 1.35 – Директория с файлами для обучения |

С помощью команды rabin2 -I получаем информацию о файле, что показано на рисунке 1.36.

С помощью команды radare2 <файл> запускаем фреймворк для работы с данным файлом. Процесс представлен на рисунке 1.37.

С помощью команды *aaa* запускаем анализ файла, что показано на рисунке 1.38.

|  |
| --- |
|  |
| Рисунок 1.36 – Вывод информации о файле |
|  | |
| Рисунок 1.37 – Запуск radare2 | |
|  | |
| Рисунок 1.38 – Анализ файла | |

С помощью команды V переходим в визуальны режим и ищем функцию main, которая является отличной точкой входа, что показано на рисунке 1.39.

|  |
| --- |
|  |
| Рисунок 1.39 – Исследование файла в графическом режиме |

С помощью команды VV переключаемся в другой графический режим, исследуем функцию main и находим, что проверка пароля осуществляется с помощью функции sys.imp.strcmp. Процесс представлен на рисунке 1.40.

|  |
| --- |
|  |
| Рисунок 1.40 – Найденный механизм проверки |

Выходим из графического режима с помощью команды *q*. После этого запустим декомпилятор с помощью команды *pdg*, что показано на рисунке 1.41.

|  |
| --- |
|  |
| Рисунок 1.41 – Найденный механизм проверки |

Из полученного кода видно, что пароль – 250382. Закрываем radare2. После запускаем сам исполняемый файл и вводим пароль. Процесс показан на рисунке 1.42.

|  |
| --- |
|  |
| Рисунок 1.42 – Найденный механизм проверки |

Из содержимого пользовательского экрана видно, что задача успешно решена.

# Инструмент IDA Pro

Для обучения Reverse Engineering(у) будем использовать IDA Pro от компании Hex-Rays.

IDA Pro как дизассемблер способен создавать карты выполнения команд приложениями, чтобы показать двоичные инструкции, которые фактически выполняются процессором, в символьном представлении (язык ассемблера) [4]. В IDA Pro внедрены передовые технологии, позволяющие генерировать исходный код на языке ассемблера из машинно-исполняемого кода и делать этот сложный код более удобочитаемым для человека.

Функция отладки дополнила IDA Pro динамическим анализом. Он поддерживает несколько целей отладки и может обрабатывать удаленные приложения. Его кроссплатформенная возможность отладки обеспечивает мгновенную отладку, простое подключение как к локальным, так и к удаленным процессам, а также поддержку 64-разрядных систем и новые возможности подключения [5].

Чтобы установить бесплатную версию, необходимо скачать её с [официального сайта разработчика](https://hex-rays.com/ida-free/#download) - [https://hex-rays.com/ida-free/#download](https://hex-rays.com/ida-free/%23download).

# Работа с графическим интерфейсом IDA Pro

При запуске программы наблюдаем окно, в котором и начнем работу с исполняемыми файлами (Рисунок 2.1).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.1 – Окно запуска программы IDA Pro

Рассмотрим элементы интерфейса:

* Кнопка “New” – позволяет моментально открыть исполняемый файл для дизассемблирования;
* Кнопка “Go” – позволяет запустить графический интерфейс программы IDA Pro без дизассемблирования кода;
* Кнопка “Previous” – загружает последний дизассемблированный файл.

Первое, на что обратим внимание на GUI программы IDA Pro – это главная панель инструментов.

Для того чтобы разобраться в панели инструментов, обозначим номером каждый элемент, как показано на Рисунке 2.2:

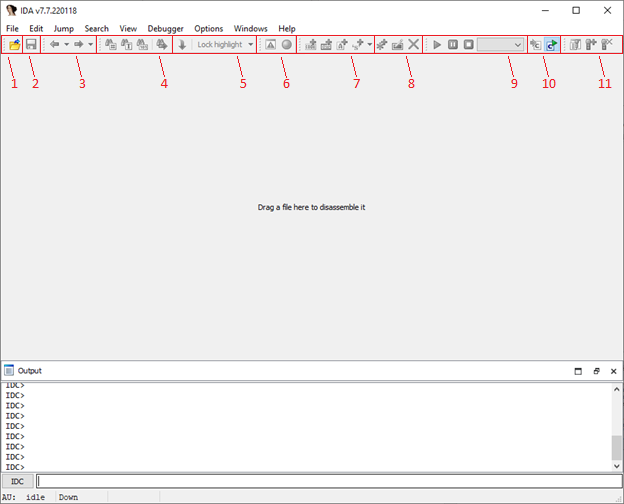


Рисунок 2.2 –GUI программы IDA Pro

1. Загрузка исполняемого файла либо базы данных.
2. Сохранение проекта (базы данных).
3. Возврат к предыдущей или переход к следующей сохраненной позиции.
4. Компоненты, отвечающие за поиск (значение, текст, последовательность байтов).
5. Фиксация выделенного объекта в коде.
6. Открытие окна с ошибками.
7. Конвертация выделенного объекта (в C подобный язык, в данные, строку, массив).
8. Переименование текущего объекта.
9. Запуск, остановка или завершение работы отладчика.
10. Изменение вида отображаемого кода.
11. Добавление и удаление точки остановки.

Начнем разбор процесса дизассемблирования исполняем ого файла. Для этого откроем его в программе IDA Pro, как показано на Рисунке 2.3.

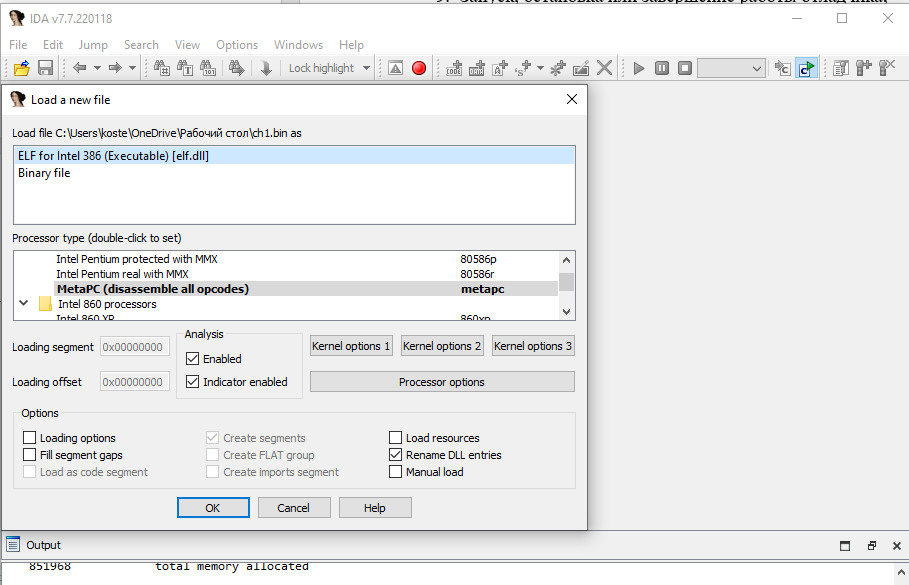


Рисунок 2.3 – Открытие исполняемого файла в IDA Pro

Как можно видеть, исполняемый файл имеет формат ELF. Данный формат используется во многих современных UNIX-подобных операционных системах. В основном, для дизассемблирования используются форматы ELF и PE различных архитектур, но это все зависит от нужд просмотра конкретного формата [6,7].

После открытия файла видим появившийся в окне дизассемблированный код, как показано на Рисунке 2.4.

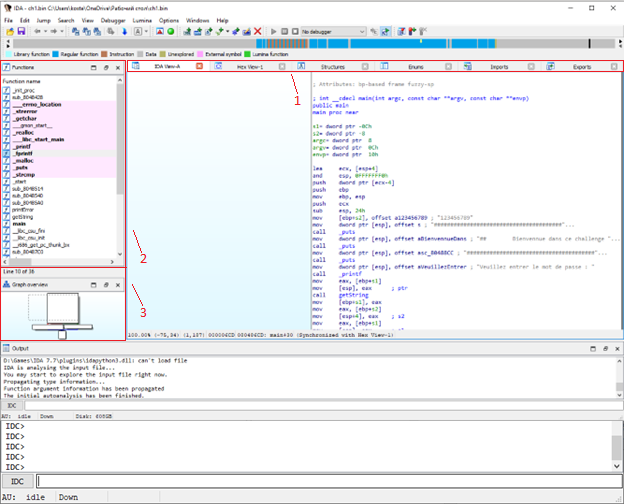


Рисунок 2.4 – Окно с кодом файла на языке Assembler

Подробно разберем каждый компонент. Начнем с верхней части окна. Вкладки, которые отвечают за отдельные функции программы IDA Pro, обведены на Рисунке 4 красным прямоугольником с номером 1:

* IDA View-A отвечает за отображение на экране кода на языке ассемблера;
* Hex View-1 отвечает за отображение на экране последовательностей байтов;
* Structures отвечает за отображение структуры кода;
* Enums позволяет создавать перечисляемый тип данных;
* Imports отвечает за импортируемые функции кода;
* Exports отвечает за экспортируемые функции кода.

Слева наблюдаем раздел со списком функций исполняемого файла, обведенным на Рисунке 2.4 красным прямоугольником с номером 2.

Для удобного перемещения по графам дизассемблированного кода имеется меню с отображением графов, обведенное на Рисунке 2.4 прямоугольником с номером 3.

# Реверс-инжиниринг с использованием IDA Pro

В данном разделе рассмотрены задачи, аналогичные выполняемым на соревнованиях, называемых CTF (захват флага, [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Capture the Flag), то есть будут рассмотрены задания с поиском флага в исполняемом файле [8-10].

##### Пример 1.

Дан исполняемый файл, необходимо найти флаг (он же пароль).

Открываем исполняемый файл в IDA Pro, как показано на Рисунке 2.5.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.5 – Открытие исполняемого файла в IDA Pro

Ищем в меню функций программы функцию “main”, как показано на Рисунке 2.6. В ней обычно описываются основные команды программы.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.6 – Поиск функции "main"

Далее необходимо выполнить поиск строки, как показано на Рисунке 2.7. Будем искать строку «print», так как она отвечает за вывод информации в консоль.

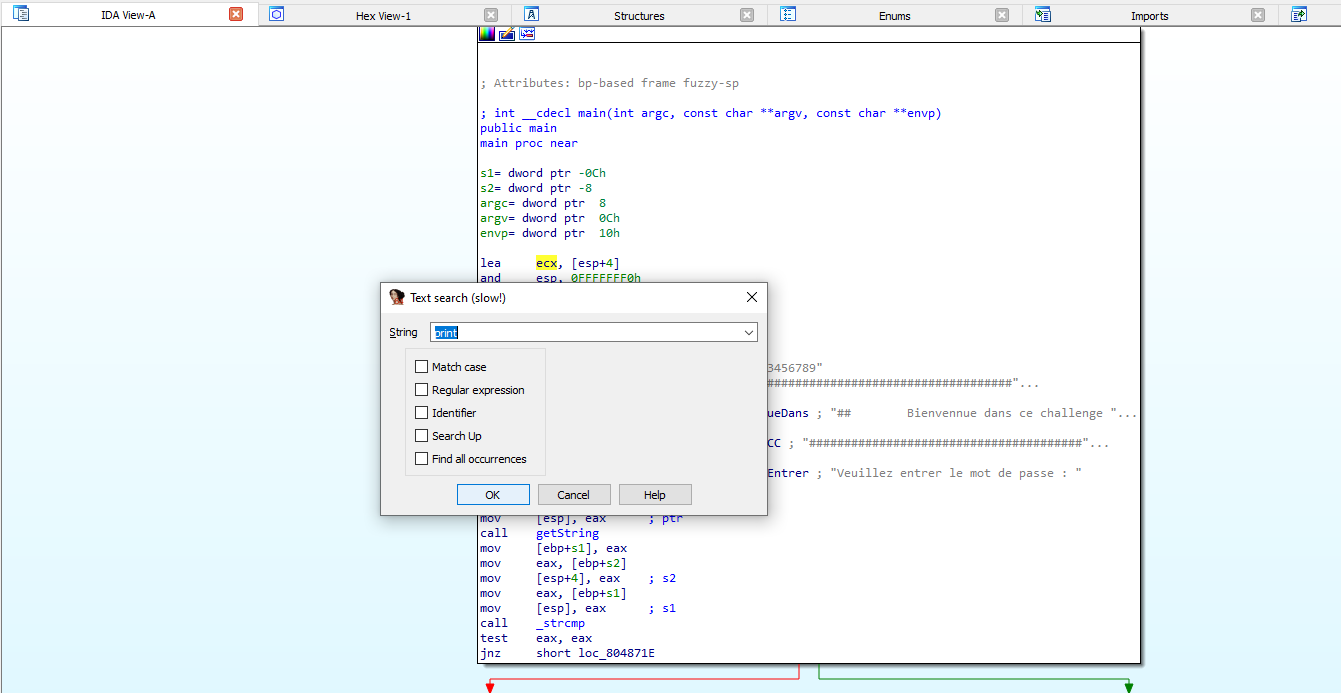


Рисунок 2.7 – Поиск строки "print"

Желтым цветом на Рисунке 2.8 выделена строка с тем параметром, который искали, а именно «print».

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.8 – Найденный фрагмент текста

Если проанализируем код выше искомой строки, то увидим строковые значения, обведенный красным прямоугольником на Рисунке 2.9.

Видим подозрительную строку с цифрами «123456789». Она и является искомым флагом поставленной задачи.

##### Пример 2.

Найти флаг исполняемого файла (он же пароль). В исполняемом файле есть базовая защита.

Повторяем действия, как в Примере 1, открываем файл, ищем функцию main и просматриваем код, представленный на Рисунке 2.10.

Можем видеть, в данном файле стоит базовая защита, а именно ввод имени пользователя и пароль. Но нас это не должно смущать, ведь мы залезли внутрь программы. Ищем строку «print». Результат поиска отображен на Рисунке 2.11.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.9 – Строковые значения в коде

Появилось несколько позиций с искомой строкой. Чтобы разобраться, какая строка отвечает за что, нужно проанализировать код [11].

Первая строка отвечает за вывод поля «Имя пользователя». Далее после ввода корректного имени пользователя продвигаемся налево и у нас запрашивают пароль от данного пользователя. Если введено неправильное «Имя пользователя», то двигаемся в средний блок и выдается сообщение «Неправильное имя пользователя».

Если мы ввели неправильный пароль, то нам выводится сообщение «Неправильный пароль» – это правый блок. Если спустимся в левый блок, то увидим искомый пароль от пользователя. Для того, чтобы узнать «Имя пользователя» необходимо посмотреть в начальный блок.

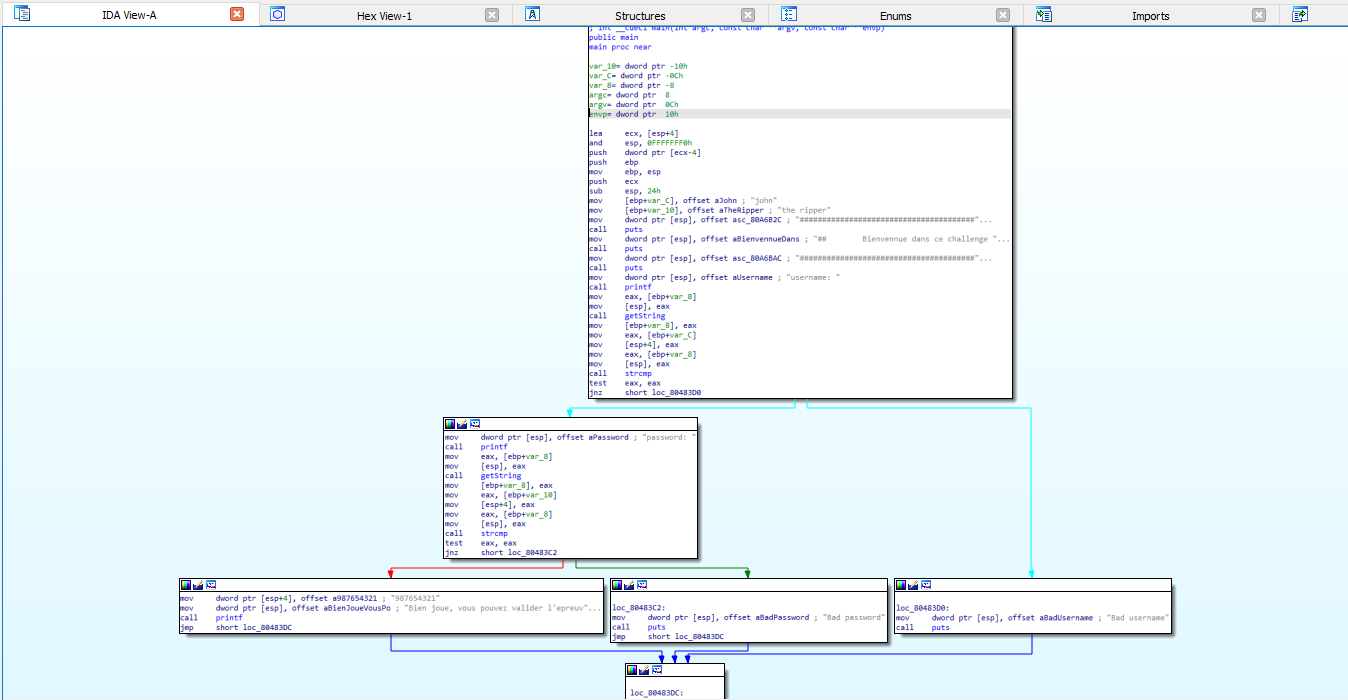


Рисунок 2.10 – Отображение кода исполняемого файла

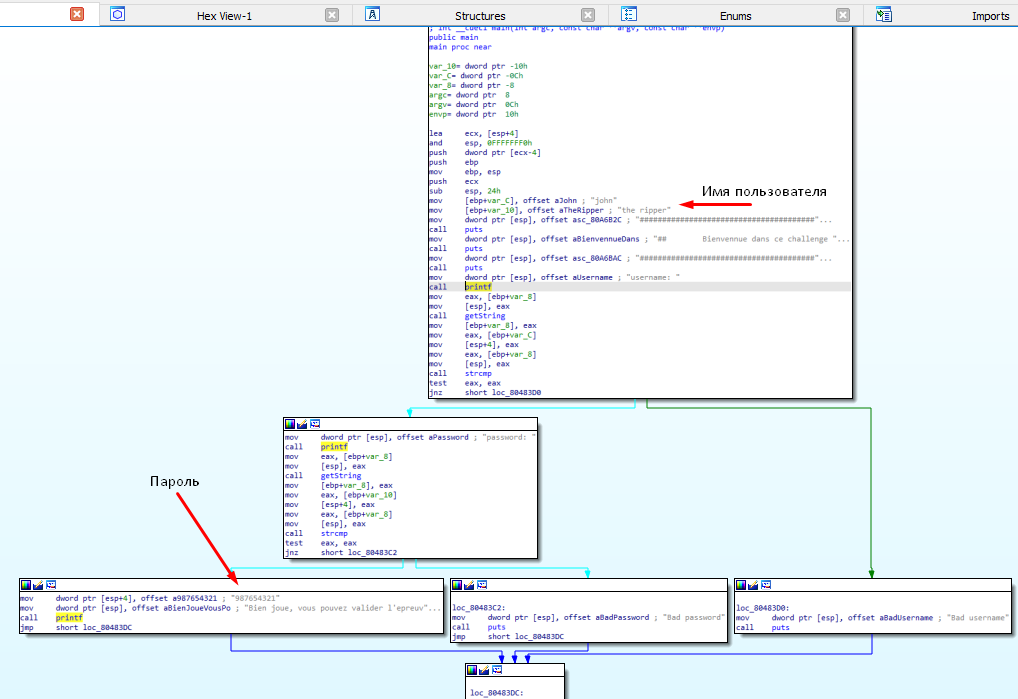


Рисунок 2.11 – Отображение искомой строки, пароля и имени пользователя

# Заключение

В заключении следует подчеркнуть важность умения владеть инструментами анализа обратного инжиниринга, к которым относятся представленные в учебном пособии radare2 и IDA Pro. Наличие возможности установки данного программного обеспечения различными способами на различные платформы Windows и Linux, делает этот инструмент универсальным и доступным для широкого круга пользователей.

Рассмотрены основные команды интерфейса и графического интерфейса, что позволяет начинающим пользователям быстро освоиться в данной среде. Приведенные примеры решения задачи демонстрирует эффективность и функциональность фреймворков в реальных условиях.

Дизассемблер используется для анализа программного продукта, исходный код которого неизвестен, а также для анализа и оптимизации создаваемого машинного кода. Представленные программные продукты широко используются для анализа программного кода, так как отличаются гибкостью, наличием встроенного командного языка, поддержкой множества форматов исполняемых файлов для большого числа процессоров и операционных систем, позволяют строить блок-схемы, изменять названия меток, просматривать локальные процедуры в стеке и многое другое.

В начале исследования дизассемблер выполняет автоматический анализ программы, а затем пользователь с помощью интерактивных средств IDA Pro начинает давать осмысленные имена, комментировать, создавать сложные структуры данных и другим образом добавлять информацию в листинг, генерируемый дизассемблером, пока не станет ясно, что именно и как делает исследуемая программа.

Стоит помнить, что представленные в учебном пособии программные продукты – это лишь инструмент, и его эффективность во многом зависит от навыков и знаний самого пользователя.

Регулярная практика, изучение новых материалов и постоянное решение задач позволят вам стать опытным специалистом в области обратного инжиниринга. Данное методическое пособие поможет студентам в области информационных технологий в выполнении лабораторных работ. Важно следить за обновлениями и улучшениями инструмента, чтобы быть в курсе последних достижений в этой области.

# Список литературы

1. Официальный репозиторий radare2. – URL: <https://github.com/radareorg/radare2?ysclid=lklhh2mau9327875799> (дата обращения: 27.08.2023).
2. Официальный сайт Iaito. – URL: <https://www.radare.org/n/iaito.html>] (дата обращения: 27.08.2023).
3. Crackmes. – URL: https://github.com/Maijin/radare2-workshop-2015 (дата обращения: 27.08.2023).
4. Bruce Dang, Alexandre Gazet – Practical Reverse Engineering:x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation. – URL: https://repo.zenk-security.com/Reversing%20.%20cracking/Practical%20Reverse%20Engineering.pdf (дата обращения: 27.08.2023).
5. Eagle C. The IDA Pro Book. – San Francisco: No Starch Press. 2011. – 646 с.
6. Денис Юричев – Реверсинг для начинающих. – URL: http://library.bagrintsev.me/unsorted/RE\_for\_beginners-ru.pdf (дата обращения: 27.08.2023).
7. Клементьев К.Е Компьютерные вирусы и антивирусы: взгляд программиста. – М.: ДМК Пресс, 2013 – 656 с.
8. Gynvael Coldwind – Practical RE tips. – URL: https://www.youtube.com/watch?v=Jk5Yad598vs (дата обращения: 27.08.2023).
9. Саттон М., Грин А., Амини П. Fuzzing: исследование уязвимостей методом грубой силы. –СПб.: Символ-Плюс, 2009. – 560 с.
10. Sikorski M. Practical Malware Analysis. – San Francisco: No Starch Press. 2012. – 766 с.
11. Hewardt M., Pravat D. Advanced Windows Debugging.  Addison-Wesley Professional, 2007. – 840 с.

Костенко Станислав Александрович

Домашкин Алексей Дмитриевич

Николаев Антон Михайлович,

Савицкий Даниил Дмитриевич,

Сидоренко Валентина Геннадьевна,

Часников Александр Артемович

Инструменты реверс-инжиниринга

Учебное пособие