

Reverse engineering

ОБРАТНАЯ РАЗРАБОТКА И ВЗЛОМ ПО

Основные задачи

- Установить логику программы с закрытым исходным кодом
- Воссоздать программу, аналогичную проприетарной
- Избавить проприетарную программу от ненужного функционала (проверка производителя/лицензии)

Что такое программа?

- Исполняемый файл популярных ОС и архитектур
- Байткод виртуальной машины (Java/.NET)
- Интерпретируемый код (PHP/Python/Perl)

В рамках сегодняшней лекции это только пункт 1

Исполняемый файл

Это, собственно, набор инструкций процессора, смешанный с данными, необходимыми для работы программы. На разных ОС приняты разные форматы исполняемых файлов: для Windows это PE (Portable Executable), для Linux ELF (Executable and Linkable Format). Расширения файлов .exe и <ничего> для Windows и Linux соответственно.

Важно помнить, что разделяемые библиотеки (.dll/.so) имеют схожий формат, хотя их обратная разработка затруднена не столь простой отладкой.

Как... запустить исполняемый файл?

Windows

Ммм, двойным щелчком.

Серьезно, больше ничего не нужно, можно еще запустить файл по имени из командной строки (cmd.exe)

Linux

Любой файл можно исполнить только когда у него есть права на исполнение. Запускать с файловой системы FAT без перемонтирования с -o exes невозможно.

Типичный сценарий запуска программы:

```
chmod +x file
```

```
./file #вы только посмотрите на этот путь
```

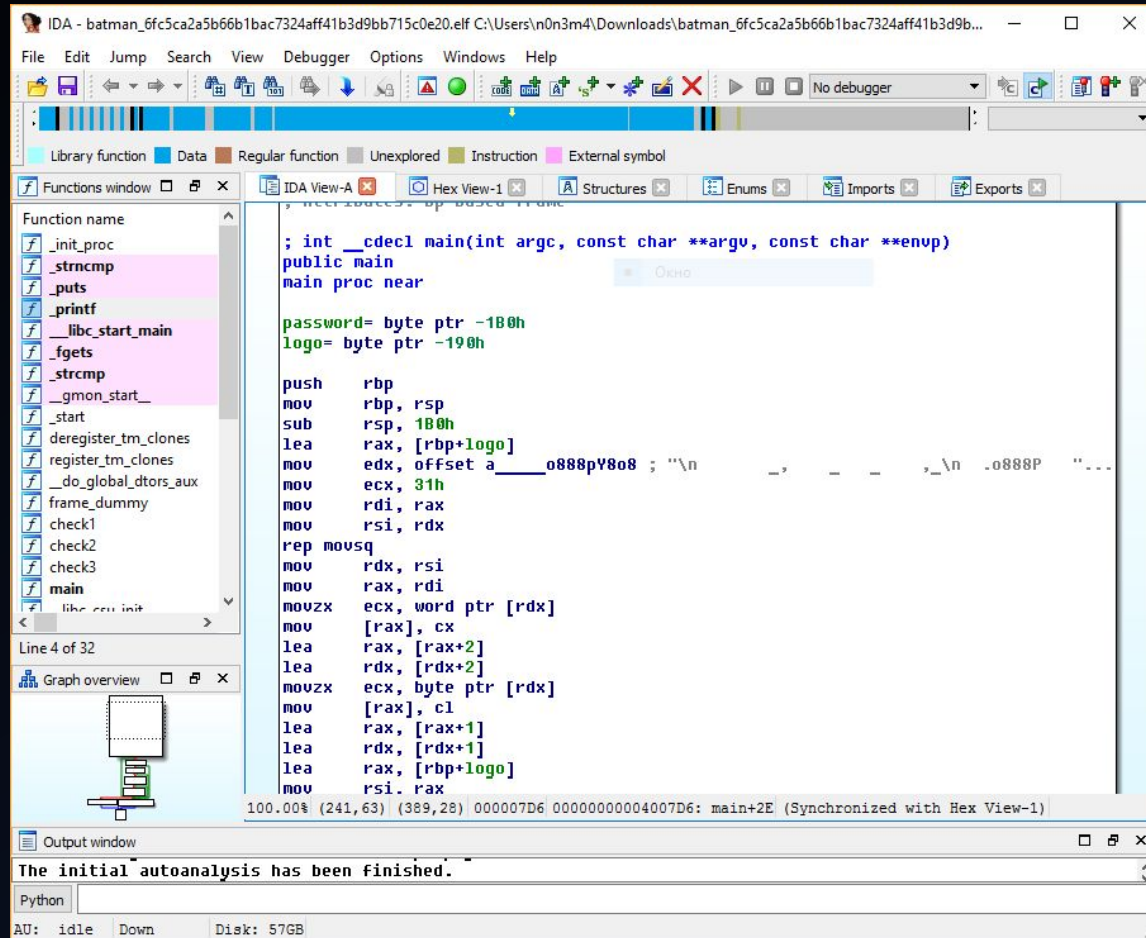
Структура исполняемого файла

Исполняемый файл состоит из сегментов, секций и всего такого. Вкратце они позволяют понять, где код, где данные, где константы и все такое.

В Linux это всё можно посмотреть командой `readelf`, в Windows – а черт его знает, это не очень нужно там.

IDA (о ней далее) покрасит все секции разными цветами сам.

Как понять что происходит?



```
IDA - batman_6fc5ca2a5b66b1bac7324aff41b3d9bb715c0e20.elf C:\Users\n0n3m4\Downloads\batman_6fc5ca2a5b66b1bac7324aff41b3d9b...
File Edit Jump Search View Debugger Options Windows Help
Library function Data Regular function Unexplored Instruction External symbol
Functions window IDA View-A Hex View-1 Structures Enums Imports Exports
Function name
_init_proc
_strcmp
_puts
_printf
_libc_start_main
_fgets
_strcmp
_gmon_start_
_start
deregister_tm_clones
register_tm_clones
__do_global_dtors_aux
frame_dummy
check1
check2
check3
main
lib_csu_init
Line 4 of 32
Graph overview
Output window
The initial autoanalysis has been finished.
Python
AU: idle Down Disk: 57GB

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

password= byte ptr -180h
logo= byte ptr -190h

push rbp
mov rbp, rsp
sub rsp, 180h
lea rax, [rbp+logo]
mov edx, offset a_____0888pV808 ; "\n _ _ _ _ _ _ _ _ _ _ \n .0888P ..."
mov ecx, 31h
mov rdi, rax
mov rsi, rdx
rep movsq
mov rdx, rsi
mov rax, rdi
movzx ecx, word ptr [rdx]
mov [rax], cx
lea rax, [rax+2]
lea rdx, [rdx+2]
movzx ecx, byte ptr [rdx]
mov [rax], cl
lea rax, [rax+1]
lea rdx, [rdx+1]
lea rax, [rbp+logo]
mov rsi, rax

100.00% (241,63) (389,28) 000007D6 000000000004007D6: main+2E (Synchronized with Hex View-1)
```

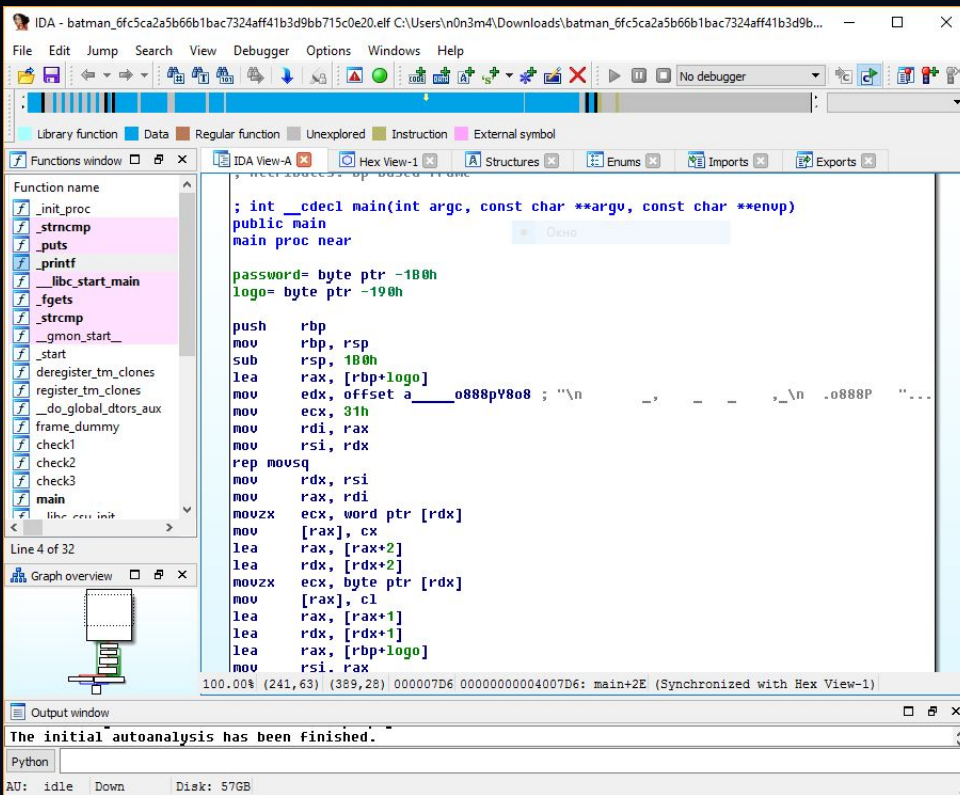


Interactive DisAssembler (IDA)

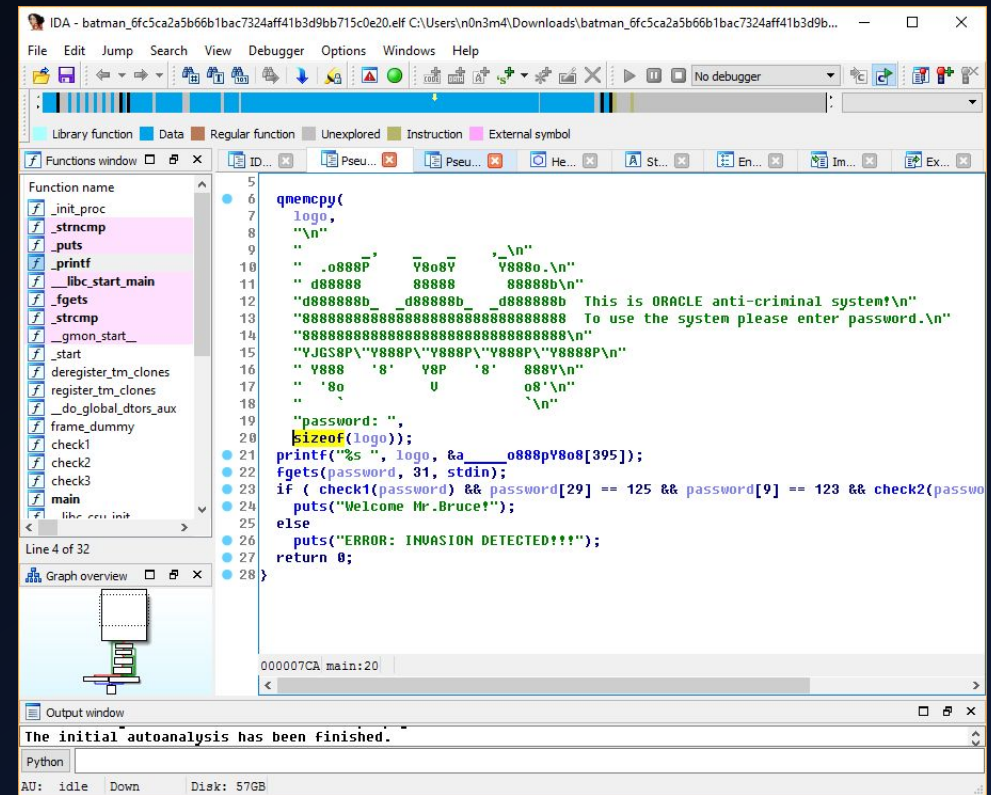
- Стоит всего лишь от \$1200. И это без декомпиляторов.
- Умеет, тем не менее, дизассемблировать почти всё
- Стандарт индустрии
- Обладает декомпилятором HexRays
- Его автор параноик и думает, все его покупатели – пираты
- А они и правда пираты... ну или неудачники.
В интернете доступна версия 6.8, которую украли у HackingTeam (если кто помнит новости)

Hexrays

- Стоит еще больше
- Умеет ДЕКОМПИЛИРОВАТЬ



```
IDA - batman_6fc5ca2a5b66b1bac7324aff41b3d9bb715c0e20.elf C:\Users\n0n3m4\Downloads\batman_6fc5ca2a5b66b1bac7324aff41b3d9bb...
File Edit Jump Search View Debugger Options Windows Help
Library function Data Regular function Unexplored Instruction External symbol
Functions window IDA View-A Hex View-1 Structures Enums Imports Exports
Function name
_init_proc
_strncmp
_puts
_printf
_libc_start_main
_fgets
_strncmp
_gmon_start_
_start
deregister_tm_clones
register_tm_clones
_do_global_dtors_aux
frame_dummy
check1
check2
check3
main
lib_cxx_init
Line 4 of 32
Graph overview
100.00% (241,63) (389,28) 000007D6 0000000004007D6: main+2E (Synchronized with Hex View-1)
Output window
The initial autoanalysis has been finished.
Python
AU: idle Down Disk: 57GB
```



```
IDA - batman_6fc5ca2a5b66b1bac7324aff41b3d9bb715c0e20.elf C:\Users\n0n3m4\Downloads\batman_6fc5ca2a5b66b1bac7324aff41b3d9bb...
File Edit Jump Search View Debugger Options Windows Help
Library function Data Regular function Unexplored Instruction External symbol
Functions window IDA View-A Hex View-1 Structures Enums Imports Exports
Function name
_init_proc
_strncmp
_puts
_printf
_libc_start_main
_fgets
_strncmp
_gmon_start_
_start
deregister_tm_clones
register_tm_clones
_do_global_dtors_aux
frame_dummy
check1
check2
check3
main
lib_cxx_init
Line 4 of 32
Graph overview
000007CA main:20
Output window
The initial autoanalysis has been finished.
Python
AU: idle Down Disk: 57GB
```

Что же теперь делать?

- Заходите в подозрительную функцию (обычно это main)
- Жмете F5
- Готово, теперь вы можете читать “код”
- Двойным щелчком можно переходить между функциями, X выводит список ссылок на объект под курсором в программе.

Как, тем не менее, понять что происходит?

- Просто прочитать. Это же легко, правда?
- Загуглить встреченные константы
- Загуглить названия функций. Макрос `assert()` выдает имена оригинальных файлов, их можно гуглить
- Отладить
- Ничего не помогло? Есть система доказательства теорем Z3
Это ее научное название, на самом деле это решалка всего подряд

Отладчик?

- Позволяет выполнять программу пошагово, посмотреть регистры, инструкции и всё такое. Только ассемблер.
- Под Windows самыми известными являются OllyDbg и x64dbg (слышали о Denuvo? Его официальный “спонсор”)
- GNU Debugger (gdb). Вообще не только для Linux, но в винде не очень хорош. А вообще крут, еще и плагины есть (PEDA)
- IDA. Ходят слухи, она умеет отлаживать даже линукс через удаленный gdb, но это неточно. Под виндой отладчик даже и неплох

Гугл?

- Позволяет искать (кто бы мог подумать)
- Ищет весьма неплохо, даже по исходным кодам
- И по константам
- Еще позволяет скачать пиратскую IDA :3

Z3/Z3py?

- Если вы обратились к Z3 вы или очень круты
- Или совсем отчаялись, причем скорее всего второе
- Позволяет обращаться хеш-функции, туповатые LFSR, решать уравнения
- Беда только в том, что почти никто не знает, как это делать правильно. Разработчики Z3 тоже, поэтому у них есть множество разных решателей систем условий (между которыми можно выбирать в самом Z3)

Пример

```
import string
from z3 import *
s = Solver()
x=BitVec('x',32)
y=BitVec('y',32)
s.add(x*y==2016)
s.add(x^y==0xDEADBEEF)
print s.check()
m=s.model()
print (m[x].as_long()*m[y].as_long())&0xFFFFFFFF,hex(m[x].as_long()^m[y].as_long())
```

Вывод:

sat

2016 0xdeadbeef

Сервер с задачами

<http://dmz.n0n3m4.ru/tasks>

Вопросы? :)