

Крис Касперски,
Ева Рокко



ИСКУССТВО ДИЗАССЕМБЛИРОВАНИЯ

- Обзор хакерского инструментария для Windows и Linux
- Техника работы с отладчиками и дизассемблерами
- Практическое дизассемблирование и идентификация ключевых структур языков высокого уровня
- Защитные механизмы и методы их обхода
- Антиотладочные приемы и борьба с ними
- Обфускация кода и ее преодоление

**Наиболее
полное
руководство**

+Ocd

В ПОДЛИННИКЕ®



УДК 681.3.06
ББК 32.973.26-018.1
К28

Касперски, К.

К28 Искусство дизассемблирования / К. Касперски, Е. Рокко. — СПб.: БХВ-Петербург, 2008. — 896 с.: ил. + CD-ROM — (В подлиннике)

ISBN 978-5-9775-0082-1

Книга посвящена вопросам и методам дизассемблирования, знание которых позволит эффективно защитить свои программы и создать более оптимизированные программные коды. Объяснены способы идентификации конструкций языков высокого уровня таких, как C/C++ и Pascal, показаны различные подходы к реконструкции алгоритмов. Приводится обзор популярных хакерских инструментов для Windows, UNIX и Linux — отладчиков, дизассемблеров, шестнадцатеричных редакторов, API- и RPC-шпионов, эмуляторов. Рассматривается исследование дампов памяти, защитных механизмов, вредоносного программного кода — вирусов и эксплоитов. Уделено внимание противодействию антиотладочным приемам. К книге прилагается компакт-диск с полноцветными иллюстрациями и кодами рассматриваемых примеров.

Для программистов и продвинутых пользователей

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Наталья Таркова</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Ольга Кокорева</i>
Компьютерная верстка	<i>Натали Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.10.07.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 72,24.

Тираж 2000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Введение.....	1
ЧАСТЬ I. ОБЗОР ХАКЕРСКИХ ПРОГРАММ.....	3
Глава 1. Инструментарий хакера.....	5
Отладчики.....	5
Дизассемблеры.....	10
Декомпиляторы.....	12
Шестнадцатеричные редакторы.....	13
Распаковщики.....	16
Дамперы.....	16
Редакторы ресурсов.....	17
Шпионы.....	18
Мониторы.....	19
Модификаторы.....	21
Копировщики защищенных дисков.....	21
Глава 2. Эмулирующие отладчики и эмуляторы.....	22
Вводная информация об эмуляторах.....	22
Исторический обзор.....	22
Области применения эмуляторов.....	24
Аппаратная виртуализация.....	29
Обзор популярных эмуляторов.....	30
DOSBox.....	30
Bochs и QEMU.....	31
VMware.....	33
Microsoft Virtual PC.....	35
Xen.....	37
Ближайшие конкуренты.....	38
Выбор подходящего эмулятора.....	39
Защищенность.....	39
Расширяемость.....	39
Доступность исходных текстов.....	39
Качество эмуляции.....	40
Встроенный отладчик.....	40
Сводная таблица характеристик эмуляторов.....	41

Глава 3. Хакерский инструментарий для UNIX и Linux	42
Отладчики	42
Дизассемблеры	46
Шпионы	47
Шестнадцатеричные редакторы	48
Дамперы	49
Скрытый потенциал ручных сборок	49
Философская подготовка	53
Пошаговая инструкция	53
Приступаем к сборке	56
Инсталляция	62
Заключение	62
Глава 4. Ассемблеры	63
Философия ассемблера	63
Объяснение ассемблера на примерах C	65
Ассемблерные вставки как тестовый стенд	66
Необходимый инструментарий	67
Сравнение ассемблерных трансляторов	67
Основополагающие критерии	68
MASM	71
TASM	73
FASM	73
NASM	75
YASM	76
Программирование на ассемблере для UNIX и Linux	77
Заключение	82
Ссылки на упомянутые продукты	83
ЧАСТЬ II. БАЗОВЫЕ ТЕХНИКИ ХАКЕРСТВА	85
Глава 5. Введение в защитные механизмы	87
Классификация защит по роду секретного ключа	89
Надежность защиты	91
Недостатки готовых "коробочных" решений	92
Распространенные ошибки реализации защитных механизмов	93
Защита от несанкционированного копирования и распространения серийных номеров	93
Защита испытательным сроком и ее слабые места	94
Реконструкция алгоритма	98
Общие рекомендации	101
Защита от модификации на диске и в памяти	102
Противодействие дизассемблеру	102
Антиотладочные приемы	103
Антимониторы	103
Противодействие дамперам	103
Мелкие промахи, ведущие к серьезным последствиям	104

Глава 6. Разминка.....	107
Создаем защиту и пытаемся ее взломать	107
Знакомство с дизассемблером	109
Пакетные дизассемблеры и интерактивные дизассемблеры.....	110
Использование пакетных дизассемблеров.....	111
От EXE до CRK	113
Практический пример взлома.....	125
Подавление NAG-screen.....	126
Принудительная регистрация	129
Чистый взлом или укрощение окна About.....	132
Заключение.....	135
Глава 7. Знакомство с отладкой.....	136
Введение в отладку	137
Дизассемблер и отладчик в одной упряжке.....	137
Точки останова на функции API.....	139
Точки останова на сообщения	141
Точки останова на данные.....	142
Раскрутка стека	143
Отладка DLL.....	145
Заключение.....	146
Глава 8. Особенности отладки в UNIX и Linux	147
Ptrace — фундамент для GDB.....	149
Библиотека Ptrace и ее команды	150
Поддержка многопоточности в GDB	151
Краткое руководство по GDB.....	152
Трассировка системных вызовов.....	156
Отладка двоичных файлов в GDB	157
Подготовка к отладке	157
Приступаем к трассировке	162
Погружение в технику и философию GDB	164
Заключение.....	173
Глава 9. Особенности термоядерной отладки с Linice	174
Системные требования	175
Компиляция и конфигурирование Linice	176
Загрузка системы и запуск отладчика	177
Основы работы с Linice.....	180
Заключение.....	184
Глава 10. Расширенное обсуждение вопросов отладки	185
Использование SoftICE в качестве логгера.....	185
Легкая разминка.....	186
Более сложные фильтры	189
Анимированная трассировка в SoftICE	192
Отладчик WinDbg как API- и RPC-шпион.....	193
Первое знакомство с WinDbg	194
Техника API-шпионажа.....	197
Техника RPC-шпионажа.....	203

Хакерские трюки с произвольными точками останова	204
Секреты пошаговой трассировки.....	204
Взлом через покрытие	213
Руководящая идея.....	213
Выбор инструментария	213
Алгоритмы определения покрытия.....	215
Выбор подхода.....	216
Пример взлома.....	218
Заключение.....	222
 ЧАСТЬ III. ИДЕНТИФИКАЦИЯ КЛЮЧЕВЫХ СТРУКТУР ЯЗЫКОВ ВЫСОКОГО УРОВНЯ.....	 223
Глава 11. Идентификация функций.....	225
Методы распознавания функций.....	225
Перекрестные ссылки.....	226
Автоматическая идентификация функций посредством IDA Pro.....	231
Пролог	232
Эпилог	232
"Голые" (naked) функции	234
Идентификация встраиваемых (inline) функций	234
Модели памяти и 16-разрядные компиляторы.....	237
Глава 12. Идентификация стартовых функций.....	238
Идентификация функции WinMain	238
Идентификация функции DllMain	239
Идентификация функции main консольных Windows-приложений	240
Глава 13. Идентификация виртуальных функций.....	242
Идентификация чистой виртуальной функции	247
Совместное использование виртуальной таблицы несколькими экземплярами объекта	249
Копии виртуальных таблиц	251
Связный список.....	251
Вызов через шлюз.....	252
Сложный пример, когда неvirtуальные функции попадают в виртуальные таблицы.....	252
Статическое связывание.....	257
Идентификация производных функций.....	261
Идентификация виртуальных таблиц.....	263
Глава 14. Идентификация конструктора и деструктора	266
Объекты в автоматической памяти — ситуация, когда конструктор/деструктор идентифицировать невозможно	270
Идентификация конструктора/деструктора в глобальных объектах	271
Виртуальный деструктор	273
Виртуальный конструктор	273
Конструктор раз, конструктор два... ..	274
Пустой конструктор.....	274

Глава 15. Идентификация объектов, структур и массивов	275
Идентификация структур	275
Идентификация объектов	282
Объекты и экземпляры	286
Мой адрес — не дом и не улица	286
Глава 16. Идентификация <i>this</i>	288
Глава 17. Идентификация <i>new</i> и <i>delete</i>	289
Идентификация <i>new</i>	289
Идентификация <i>delete</i>	291
Подходы к реализации кучи	291
Глава 18. Идентификация библиотечных функций	292
Глава 19. Идентификация аргументов функций	297
Соглашения о передаче параметров	297
Цели и задачи	298
Определение количества и типа передачи аргументов	299
Адресация аргументов в стеке	304
Стандартное соглашение — <i>stdcall</i>	308
Соглашение <i>cdecl</i>	310
Соглашение <i>Pascal</i>	312
Соглашения о быстрых вызовах — <i>fastcall</i>	323
Соглашения о вызовах <i>thiscall</i> и соглашения о вызове по умолчанию	352
Аргументы по умолчанию	354
Техника исследования механизма передачи аргументов неизвестным компилятором	355
Глава 20. Идентификация значения, возвращаемого функцией	356
Возврат значения оператором <i>return</i>	357
Возврат вещественных значений	371
Возвращение значений встроенными ассемблерными функциями	375
Возврат значений через аргументы, переданные по ссылке	377
Возврат значений через динамическую память (кучу)	383
Возврат значений через глобальные переменные	386
Возврат значений через флаги процессора	391
Глава 21. Идентификация локальных стековых переменных	393
Адресация локальных переменных	394
Детали технической реализации	395
Идентификация механизма выделения памяти	395
Инициализация локальных переменных	396
Размещение массивов и структур	396
Выравнивание в стеке	397
Как IDA Pro идентифицирует локальные переменные	397
Исключение указателя на фрейм	404
Глава 22. Идентификация регистровых и временных переменных	408
Регистровые переменные	409
Временные переменные	413
Создание временных переменных при пересылках данных и вычислении выражений	414

Создание временных переменных для сохранения значения, возвращенного функцией, и результатов вычисления выражений.....	416
Область видимости временных переменных.....	417
Глава 23. Идентификация глобальных переменных	418
Техника восстановления перекрестных ссылок	418
Отслеживание обращений к глобальным переменным контекстным поиском их смещения в сегменте кода [данных]	418
Отличия констант от указателей	419
Косвенная адресация глобальных переменных.....	420
Статические переменные	423
Глава 24. Идентификация констант и смещений	424
Определение типа непосредственного операнда	426
Сложные случаи адресации или математические операции с указателями	429
Порядок индексов и указателей	433
Использование LEA для сложения констант.....	433
"Визуальная" идентификация констант и указателей	434
Глава 25. Идентификация литералов и строк.....	435
Типы строк	437
С-строки	437
DOS-строки.....	438
Pascal-строки.....	438
Комбинированные типы.....	439
Определение типа строк.....	439
Turbo-инициализация строковых переменных	445
Глава 26. Идентификация конструкций IF — THEN — ELSE	449
Типы условий	451
Наглядное представление сложных условий в виде дерева	453
Исследование конкретных реализаций	456
Сравнение целочисленных значений	456
Сравнение вещественных чисел.....	457
Условные команды булевой установки.....	460
Прочие условные команды	461
Булевские сравнения	462
Идентификация условного оператора "(условие)?do_it:continue".....	462
Особенности команд условного перехода в 16-разрядном режиме	466
Практические примеры	468
Оптимизация ветвлений	478
Глава 27. Идентификация конструкций SWITCH — CASE — BREAK.....	482
Идентификация операторов множественного выбора.....	482
Отличия switch от оператора case языка Pascal.....	490
Обрезка (балансировка) длинных деревьев	492
Сложные случаи балансировки или оптимизирующая балансировка.....	495
Ветвления в case-обработчиках	495
Глава 28. Идентификация циклов.....	496
Циклы с предусловием	497
Циклы с постусловием	497

Циклы со счетчиком	498
Циклы с условием в середине	500
Циклы с множественными условиями выхода	500
Циклы с несколькими счетчиками	501
Идентификация <i>continue</i>	501
Сложные условия	502
Вложенные циклы	502
Дизассемблерные листинги примеров	503
Глава 29. Идентификация математических операторов	527
Идентификация оператора +	527
Идентификация оператора –	530
Идентификация оператора /	532
Идентификация оператора %	536
Идентификация оператора *	538
Комплексные операторы	543
ЧАСТЬ IV. ПРОДВИНУТЫЕ МЕТОДЫ ДИЗАССЕМБЛИРОВАНИЯ	545
Глава 30. Дизассемблирование 32-разрядных PE-файлов	547
Особенности структуры PE-файлов в конкретных реализациях	547
Общие концепции и требования, предъявляемые к PE-файлам	548
Структура PE-файла	549
Техника внедрения и удаления кода из PE-файлов	552
Понятие X-кода и другие условные обозначения	552
Цели и задачи X-кода	553
Требования, предъявляемые к X-коду	555
Внедрение X-кода	555
Предотвращение повторного внедрения	556
Классификация механизмов внедрения	557
Категория А: внедрение в свободное пространство файла	558
Категория А: внедрение путем сжатия части файла	570
Категория А: создание нового NTFS-потока внутри файла	571
Категория В: раздвижка заголовка	573
Категория В: сброс части секции в оверлей	575
Категория В: создание собственного оверлея	578
Категория С: расширение последней секции файла	578
Категория С: создание собственной секции	581
Категория С: расширение срединных секций файла	582
Категория Z: внедрение через автозагружаемые dll	584
Глава 31. Дизассемблирование ELF-файлов под Linux и BSD	585
Необходимый инструментарий	585
Структура ELF-файлов	587
Внедрение чужеродного кода в ELF-файл	590
Заражение посредством поглощения файла	590
Заражение посредством расширения последней секции файла	592
Сжатие части оригинального файла	595
Заражение посредством расширения кодовой секции файла	600

Сдвиг кодовой секции вниз	603
Создание собственной секции	604
Внедрение между файлом и заголовком	605
Практический пример внедрения чужеродного кода в ELF-файл	606
Особенности дизассемблирования под Linux на примере tiny-crackme	612
Исследование головоломки tiny-crackme	613
Заключение	624
Глава 32. Архитектура x86-64 под скальпелем ассемблера	625
Введение	625
Необходимый инструментарий	626
Обзор архитектуры x86-64	629
Переход в 64-разрядный режим	631
Программа "Hello, world" для x86-64	633
Глава 33. Исследования ядра Linux	639
Вне ядра	639
Штурм ядра	640
Внутри ядра	642
Где гнездятся ошибки	646
Секреты кернел-хакинга	647
Меняем логотип Linux	647
Глава 34. Современные методы патчинга	652
Секреты онлайн-патчинга	652
Простейший on-line patcher	653
Гонки на опережение	655
Перехват API-функций как сигнальная система	656
Аппаратные точки останова	658
Малоизвестные способы взлома клиентских программ	660
Обзор способов взлома	660
Модификация без изменения байт	661
Хак ядра Windows NT/2000/XP	669
Структура ядра	669
Типы ядер	670
Методы модификации ядра	673
Модификация загрузочного логотипа	680
Есть ли жизнь после BSOD?	682
Преодоление BSOD с помощью SoftICE	683
Автоматическое восстановление	687
Насколько безопасна утилита Anti-BSOD?	691
Заключение	691
Глава 35. Дизассемблирование файлов других форматов	692
Дизассемблирование файлов PDF	692
Что Adobe Acrobat обещает неконформистам	693
Модификация Adobe Acrobat	697
Взлом с помощью PrintScreen	697
Становитесь полиглотами	697
Структура файла PDF	698

Генерация ключа шифрования	702
Атака на U-пароль	704
Практический взлом паролей PDF	705
Интересные ресурсы.....	708
ЧАСТЬ V. ПРАКТИЧЕСКОЕ КОДОКОПАТЕЛЬСТВО	709
Глава 36. Антиотладочные приемы и игра в прятки под Windows и Linux.....	711
Старые антиотладочные приемы под Windows на новый лад	712
Самотрассирующаяся программа.....	713
Антиотладочные примеры, основанные на доступе к физической памяти	718
Как работает Win2K/XP SDT Restore.....	722
Stealth-технологии в мире Windows	722
Синяя пилюля и красная пилюля — Windows вязнет в Матрице.....	723
Синяя пилюля.....	723
Красная пилюля	728
Stealth-технологии в мире Linux.....	730
Модуль раз, модуль два.....	731
Исключение процесса из списка задач	734
Перехват системных вызовов	737
Перехват запросов к файловой системе.....	739
Когда модули недоступны	740
Прочие методы борьбы	742
Интересные ссылки по теме стелсирования.....	743
Захватываем ring 0 в Linux.....	743
Честные способы взлома.....	744
Дырка в голубом зубе или Linux Kernel Bluetooth Local Root Exploit.....	744
Эльфы падают в дамп.....	745
Проблемы многопоточности	746
Получаем root на многопроцессорных машинах	747
Глава 37. Переполнение буфера в системах с неисполняемым стеком	750
Конфигурирование DEP	753
Проблемы совместимости.....	755
Атака на DEP.....	756
В лагере UNIX.....	761
BufferShield или PaX на Windows	763
Интересные ресурсы.....	764
Глава 38. Борьба с паковщиками	765
Предварительный анализ	765
Распаковка и ее альтернативы	768
Алгоритм распаковки	768
В поисках OEP	769
Дамп живой программы.....	769
Поиск стартового кода по сигнатурам в памяти	771
Пара популярных, но неудачных способов: GetModuleHandleA и gs:0	772
Побочные эффекты упаковщиков или почему не работает VirtualProtect.....	776
Универсальный метод поиска OEP, основанный на балансе стека.....	779

Техника снятия дампа с защищенных приложений	784
Простые случаи снятия дампа	785
В поисках самого себя	789
Дамп извне	790
Механизмы динамической расшифровки	791
Дамп изнутри	792
Грязные трюки	794
Полезные ссылки	796
Упаковщики исполняемых файлов в LINUX/BSD и борьба с ними	796
Упаковщики и производительность	797
ELF-Crypt	798
UPX	805
Burneye	807
Shiva	809
Сравнение упаковщиков	811
Глава 39. Обфускация и ее преодоление	813
Как работает обфускатор	814
Как это ломается	819
Распутывание кода	820
Черный ящик	822
Застенки виртуальной машины	824
Будущее обфускации	824
Глава 40. Обнаружение, отладка и дизассемблирование зловредных программ	826
Время тоже оставляет отпечатки	826
Дерево процессов	828
Допрос потоков	830
Восстановление SST	836
Аудит и дизассемблирование эксплоитов	840
Как препарировать эксплоиты	841
Анализ эксплоита на практическом примере	842
Как запустить shell-код под отладчиком	854
Заключение	854
Интересные ссылки	855
Глава 41. Атака на эмуляторы	856
Атака через виртуальную сеть	857
Атака через folder.htt	858
Атака через backdoor-интерфейс	859
Новейшие эксплоиты для виртуальных машин	862
VMware: удаленное исполнение произвольного кода I	862
VMware: удаленное исполнение произвольного кода II	864
VMware: перезапись произвольного файла	865
Подрыв виртуальных машин изнутри	865
Описание компакт-диска	873
Предметный указатель	875



Глава 1

Инструментарий хакера

С чего начинается хакерство? Некоторые скажут — с изучения C/C++, языка ассемблера, обучения искусству отладки и дизассемблирования... И будут правы. Другие же добавляют к этому "джентльменскому списку" изучение архитектуры разнообразных операционных систем, сетевых протоколов, поиск уязвимостей. И тоже будут правы. Но, с другой стороны — а за счет чего хакеры добиваются результатов? Правильно, за счет знаний и упорного труда. Но при этом они не только работают руками и головой, но и пользуются хакерским инструментарием. И правильный подбор программ очень важен, поскольку именно они формируют сознание, позволяя начинающему сделать свои первые шаги в дремучем лесу машинных кодов. Вот только этих программ настолько много, что новичок, впервые попавший на хакерский сайт, оказывается в полной растерянности — что качать, а что не стоит внимания. Основная цель данной главы как раз и состоит в том, чтобы дать предельно сжатый обзор хакерского софта, покрывающего практически любые потребности.

ПРИМЕЧАНИЕ

Помимо упорства и стремления к самостоятельному поиску ответов на возникающие вопросы, девизом любого хакера должна стать фраза: "Знай и люби свои инструменты". Иными словами, вы не должны довольствоваться лишь приведенным здесь кратким обзором, который призван лишь обратить ваше внимание на ту или иную программу. Как правило, в комплекте с любой из этих программ поставляется и руководство пользователя, и руководства эти настоятельно рекомендуются внимательно изучать.

Отладчики

Лучший отладчик всех времен и народов — это, конечно же, SoftICE, на котором выросло не одно поколение хакеров. Это — интерактивная программа с развитым командным интерфейсом, представляющим собой компромисс между легкостью освоения и удобством использования (рис. 1.1). Иными словами, без чтения руководства здесь не обойтись, тем более что никаких интуитивно-понятных меню SoftICE не предоставляет¹.

Изначально созданный фирмой NuMega, SoftICE был продан компании Compuware, долгое время распространявшей его в составе громоздкого пакета DriverStudio Framework. К глубокому разочарованию, 3 апреля 2006 по малопонятным причинам компания объявила о прекращении работы над продуктом, похоронив тем самым уникальнейший проект. Последняя версия DriverStudio 3.2 поддерживает всю линейку Windows вплоть до Server 2003, а также архитектуру

¹ Подробное "Руководство пользователя SoftICE" в русском переводе находится здесь: <http://www.podgoretsky.com/ftp/Docs/softice/siug.pdf>. Кроме того, достаточно краткое, но весьма полезное руководство по использованию SoftICE, позволяющее быстро начать работу с данным отладчиком, можно найти по адресу <http://www.reconstrucster.org/papers/The%20big%20SoftICE%20howto.pdf>.

AMD x86-64. То есть лет на пять запаса прочности у SoftICE еще должно хватить, а там хакеры что-нибудь придумают².

The screenshot shows the SoftICE debugger interface. At the top, CPU registers are displayed: EAX=09000000, EBX=008E60FC, ECX=00000000, EDX=008E77EC, ESI=00000000, EDI=008E78F0, EBP=0012065C, ESP=0011DD8C, EIP=77E92B8D, CS=001B, DS=0023, SS=0023, ES=0023, FS=0038, GS=0000. Below the registers is a memory dump starting at address 0010:008E77EC, showing hex values and ASCII characters like 'keyfile.dat'. The main window displays disassembly code for the function KERNEL32!CreateFileA, with instructions like PUSH EBP, MOV EBP,ESP, and CALL 77E94DA1. A status bar at the bottom indicates 'Break due to BPX KERNEL32!CreateFileA (ET=12.64 seconds)' and prompts the user to 'Enter a command (H for help)'.

Рис. 1.1. SoftICE — профессионально ориентированный отладчик, на котором выросло не одно поколение хакеров

Найти SoftICE можно практически на любом хакерском сайте (например, здесь: <http://www.woodmann.com/crackz/Tools.htm>). Чтобы не качать целиком весь пакет DriverStudio, можно воспользоваться пакетом DeMoNiX (<http://reversing.kulichki.net>), содержащим только SoftICE, выдернутый из DriverStudio v. 2.7 build 562. Этот пакет занимает всего 2,27 Мбайт. Однако инсталлятор содержит ошибки, а старая версия не поддерживает новых веяний Microsoft (хотя замечательно идет под Windows 2000).

Вместе с SoftICE желательно сразу же установить IceExt (<http://sourceforge.net/projects/iceext>) — неофициальное расширение, позволяющее скрывать присутствие отладчика от большинства защит, сохранять дампы памяти, задействовать кириллические кодировки 866/1251, приостанавливать потоки и выполнять множество других важных задач (рис. 1.2).

Если IceExt откажется запускаться, скорректируйте следующие ключи в данной ветви системного реестра: HKLM\SYSTEM\CurrentControlSet\Services\NTice: KDHeapSize (DWORD): 0x8000; KDStackSize (DWORD): 0x8000.

Другое неофициальное расширение для SoftICE, IceDump (<http://programmerstools.org/system/files?file=icedump6.026.zip>), также умеет делать много полезных вещей и удачно дополняет IceExt (рис. 1.3).

² На данный момент на роль адекватного преемника SoftICE претендует коммерческий отладчик Syser Debugger, о котором речь пойдет чуть далее. Если же вас интересует отладчик Open Source, то возможно, ваше внимание привлечет проект Rasta Ring 0 Debugger (<http://rr0d.droids-corp.org/>).

```
EAX=00300E40  EBX=7FFDF000  ECX=004060B8  EDX=00000003  ESI=00000000
EDI=00000000  EBP=0012FFC0  ESP=0012FF84  EIP=00401001  o d I s z a P c
CS=001B  DS=0023  SS=0023  ES=0023  FS=0038  GS=0000

test_dump  byte  PROT--(0)
0010:00400000  4D 5A 90 00 03 00 00 00-04 00 00 00 FF FF 00 00  MZÉ.....
0010:00400010  B8 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00  .....0.....
0010:00400020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
0010:00400030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
PROT32-
001B:00401001  PUSH  00406030
001B:00401006  CALL  00401010
001B:0040100B  POP   ECX
001B:0040100C  RET
001B:0040100D  NOP
001B:0040100E  NOP
001B:0040100F  NOP
001B:00401010  PUSH  EBX
001B:00401011  PUSH  ESI
001B:00401012  MOV   ESI,00406068
(PaSStUE)-KTEB(811EB020)-TID(0190)--test_dump!.text+0001
.:!DUMP
Dump memory to disk
!dump FileName Addr Len
Ex:
!dump c:\dump.dat 400000 1000
!dump \??\c:\dump.dat 400000 1000
!dump \??\c:\dump.dat edx+ebx ecx
.:!DUMP C:\dumped 400000 7DE8
DUMP: \??\C:\dumped 400000 7de8
.
Enter a command (H for help) test_dum
```

Рис. 1.2. Снятие дампа с помощью IceExt

```
EAX=00300E40  EBX=7FFDF000  ECX=004060B8  EDX=00000003  ESI=00000000
EDI=00000000  EBP=0012FFC0  ESP=0012FF84  EIP=00401001  o d I s z a P c
CS=001B  DS=0023  SS=0023  ES=0023  FS=0038  GS=0000

test_dump!.text+0001  byte  PROT--(0)
001B:00401001  68 30 60 40 00 E8 05 00-00 00 59 C3 90 90 90 53  h0'@.0...Y.EÉES-
001B:00401011  56 BE 68 60 40 00 57 56-E8 4B 01 00 00 8B F8 8D  U.h@.WUeK...Y.i!
001B:00401021  44 24 18 50 FF 74 24 18-56 E8 04 02 00 00 56 57  D$.P.t$.Ue...Uw!
001B:00401031  8B D8 E8 BE 01 00 00 83-C4 18 8B C3 5F 5E 5B C3  i.0...ä..i..^!
PROT32-
001B:00401001  PUSH  00406030
001B:00401006  CALL  00401010
001B:0040100B  POP   ECX
001B:0040100C  RET
001B:0040100D  NOP
001B:0040100E  NOP
001B:0040100F  NOP
001B:00401010  PUSH  EBX
001B:00401011  PUSH  ESI
001B:00401012  MOV   ESI,00406068
(PaSStUE)-KTEB(812121E0)-TID(03B0)--test_dump!.text+0001
.:MOD test_dump
hMod Base PEHeader Module Name File Name
00400000 00400000 test_dump \TEMP\test_dump.exe
.:MAP32 test_dump
Owner Obj Name Obj# Address Size Type
test_dump .text 0001 001B:00401000 00003B46 CODE R0
test_dump .rdata 0002 0023:00405000 0000080E IDATA R0
test_dump .data 0003 0023:00406000 00001DE8 IDATA RW
.
Enter a command (H for help) test_dum
```

Рис. 1.3. Снятие дампа с помощью IceDump

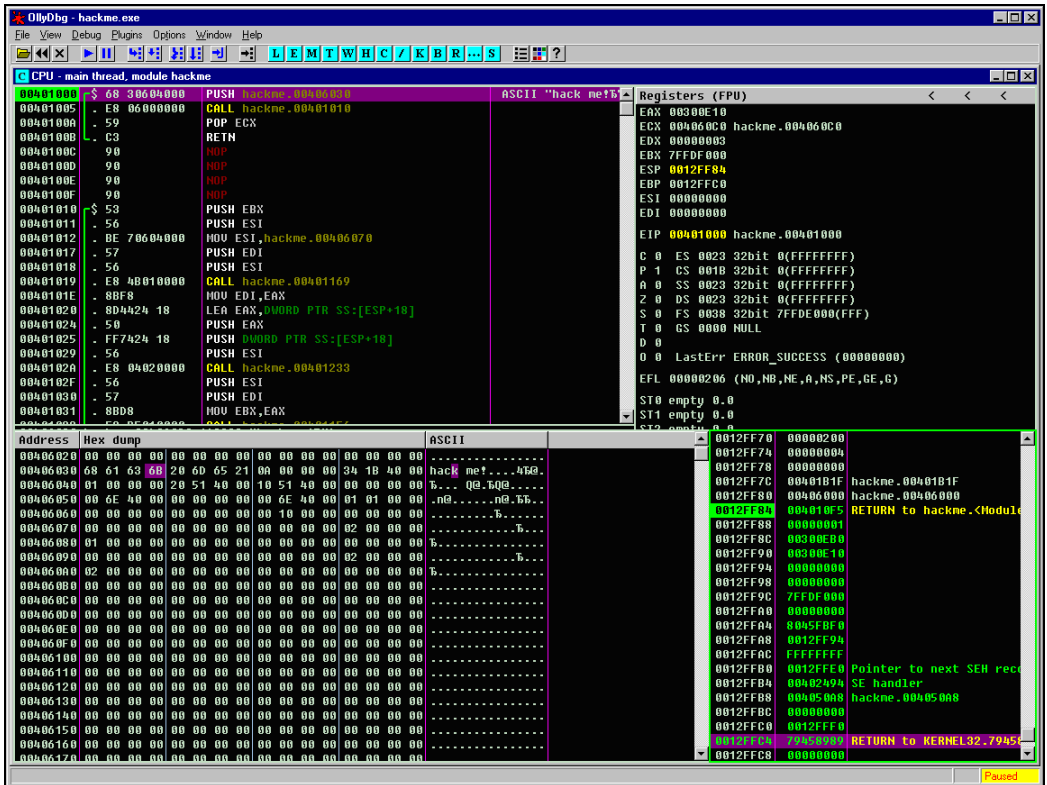


Рис. 1.4. Компактный и шустрый отладчик OllyDbg

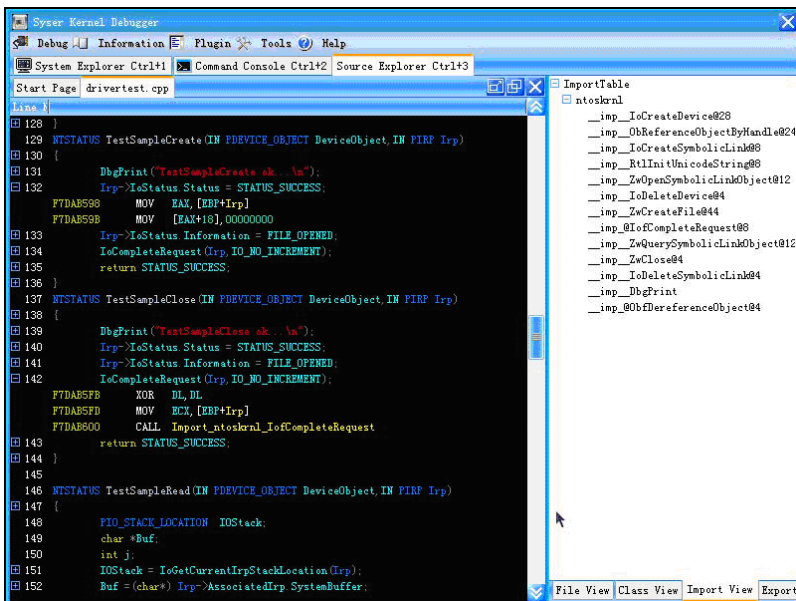


Рис. 1.5. Отладчик Syser Debugger для отладки термоядерного драйвера

Кстати, сам SoftICE замечательно работает под виртуальной машиной VMware, для этого достаточно добавить в конфигурационный файл с расширением .vmx следующие две строки: `paevm = TRUE` и `processor1.use = FALSE`.

ПРИМЕЧАНИЕ

При работе с SoftICE на многоядерных процессорах и процессорах с поддержкой HyperThreading также были отмечены некоторые проблемы (хотя возникают они нерегулярно). Устранить эти проблемы в случае их возникновения можно, добавив ключ `/ONECPU` в файл `boot.ini`.

Помимо SoftICE существуют и другие отладчики, из которых в первую очередь хотелось бы отметить бесплатный Olly Debugger (<http://www.ollydbg.de>). Это — удобный инструмент прикладного уровня (рис. 1.4), ориентированный на хакерские потребности, поддерживающий механизм плагинов (plug-ins) и собравший вокруг себя целое сообщество, написавшее множество замечательных расширений и дополнений, прячущих OllyDbg от защит, автоматически определяющих оригинальную точку входа в упакованной программе, облегчающих снятие протекторов и т. д.

Неплохие коллекции плагинов можно найти на сайтах <http://www.wasm.ru> и <http://www.openrce.org>.

Самый свежий (и пока еще во многом экспериментальный) ядерный отладчик — это, бесспорно, Syser (<http://www.sysersoft.com>). Данный отладчик выпущен китайскими разработчиками и в настоящее время переживает стадию активного развития и становления (рис. 1.5). Как уже говорилось чуть ранее, этот отладчик претендует на роль преемника SoftICE. Его последняя версия (v. 1.9, датированная 17 мая 2007 года) работает с 32-разрядными версиями Windows 2000/XP/2003/Vista, а также обеспечивает поддержку SMP, HyperThreading и многоядерных процессоров.

Довольно многие программисты пользуются отладчиком уровня ядра Microsoft WinDbg, входящим в состав бесплатного набора Debugging Tools (<http://www.microsoft.com/whdc/devtools/debugging/debugstart.msp>). Он вполне пригоден для взлома (рис. 1.6). Тем не менее, большинство хакеров, привыкших к черному экрану SoftICE, считают его неудобным.

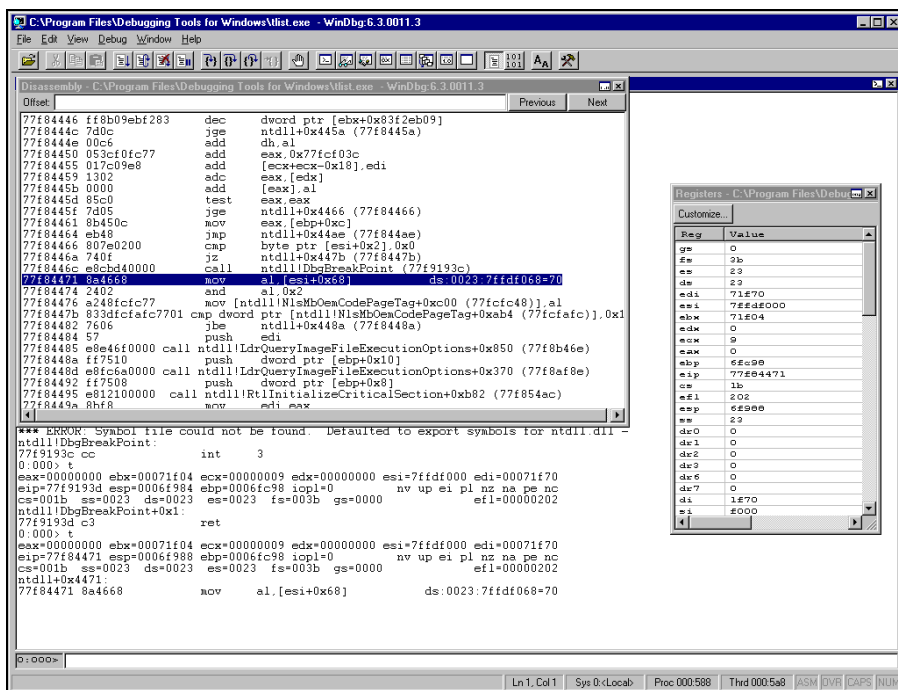


Рис. 1.6. Отладчик WinDbg

Дизассемблеры

Существует всего лишь один дизассемблер, пригодный для по-настоящему профессиональной работы — IDA Pro (<http://www.idapro.com>). Этот дизассемблер имеет консольную (рис. 1.7) и графическую (рис. 1.8) версии. IDA Pro воспринимает огромное количество форматов файлов и множество типов процессоров, легко справляясь с байт-кодом виртуальных машин Java и .NET, поддерживает макросы, плагины и скрипты, содержит интегрированный отладчик, работает под MS-DOS, Windows, Linux (рис. 1.9) и обладает уникальной способностью распознавать имена стандартных библиотечных функций по их сигнатурам.

Существует несколько версий IDA Pro — бесплатная (freeware), стандартная (standard) и расширенная (advanced). Бесплатную версию IDA Pro можно скачать по адресу http://www.dirfile.com/ida_pro_freeware_version.htm. Стоит, правда, отметить, что бесплатная версия, по сравнению со стандартной и расширенной, обладает ограниченными возможностями. Из всех процессорных архитектур поддерживается только x86, а функция поддержки подключаемых модулей попросту отсутствует. Что касается стандартной и расширенной версий, то они, как и любое хорошее программное обеспечение, стоят дорого (хотя, если хорошо поискать в файлообменных сетях, их можно стянуть оттуда бесплатно).

Основное достоинство IDA Pro состоит в том, что это — интерактивный дизассемблер, то есть интеллектуальный инструмент, позволяющий работать с двоичным файлом, мыслить и творить, а не тупой автомат, заглатывающий исследуемую программу и выплевывающий "готовый" дизассемблированный листинг, в котором все дизассемблировано неправильно.

```

IDA - nls.sys
File Edit Jump Search View Options Window IDA View-1 2
INIT:00086599 89 45 F8      mov     [ebp+P1], eax
INIT:0008659C          loc_8659C:
INIT:0008659C 8D 45 FF      lea     eax, [ebp+var_11]
INIT:0008659F C7 45 F4 FC+   mov     [ebp+ValueName.Buffer], offset aNtfsMftZones ; "NtfsMftZoneReservation"
INIT:000865A6 50          push   eax
INIT:000865A7 8D 45 F8      lea     eax, [ebp+P1]
INIT:000865AA 50          push   eax
INIT:000865AB 8D 45 EC      lea     eax, [ebp+var_14]
INIT:000865AE 50          push   eax
INIT:000865AF 8D 45 F0      lea     eax, [ebp+ValueName]
INIT:000865B2 50          push   eax
INIT:000865B3 8D 45 E4      lea     eax, [ebp+var_1C]
INIT:000865B6 50          push   eax
INIT:000865B7 66 C7 45 F0+   mov     [ebp+ValueName.Length], 2Ch
INIT:000865BD 66 C7 45 F2+   mov     [ebp+ValueName.MaximumLength],
INIT:000865C3 E8 54 09 00+   call   sub_86F1C
INIT:000865C8 3B C6      cmp     eax, esi
INIT:000865CA 8B 45 F8      mov     eax, [ebp+P1]
INIT:000865CD 7C 15      j1      short loc_865E4
INIT:000865DF 8B 48 08      mov     ecx, [eax+8]
INIT:000865D2 8B 0C 01      mov     ecx, [ecx+eax]
INIT:000865D5 3B CE      cmp     ecx, esi
INIT:000865D7 74 0B      jz      short loc_865E4
INIT:000865D9 83 F9 04      cmp     ecx, 4
INIT:000865DC 77 06      ja      short loc_865E4
INIT:000865DE 89 80 84 53+   mov     dword_25384, ecx
INIT:000865E4          loc_865E4:
INIT:000865E4 3B C6      cmp     eax, esi
INIT:000865E6 75 10      jnz     short loc_865F8
INIT:000865E8 80 65 FF 00+   and     byte ptr [ebp+var_14], 0
INIT:000865EC 8D 85 3C FF+   lea     eax, [ebp+var_C]
INIT:000865F2 89 7D EC      mov     [ebp+var_14], eax
INIT:000865F5 89 45 F8      mov     [ebp+P1], eax
INIT:000865F8          loc_865F8:
INIT:000865F8 8D 45 FF      lea     eax, [ebp+var_11]
INIT:000865FB C7 45 F4 2C+   mov     [ebp+ValueName.Buffer], offset aNtfsQuotaNotif ; "NtfsQuotaNotifyRate"
INIT:00086602 50          push   eax
INIT:00086603 8D 45 F8      lea     eax, [ebp+P1]
INIT:00086606 50          push   eax
INIT:00086607 8D 45 EC      lea     eax, [ebp+var_14]
INIT:0008660A 50          push   eax
INIT:0008660B 8D 45 F0      lea     eax, [ebp+ValueName]
-0008659C: start:loc_8659C
Executing function 'main'...
Flushing buffers, please wait... ok
F1 Help  C Code  D Data  N Names  Alt-X Quit  F10 Menu  DISK: 943M

```

Рис. 1.7. Консольная версия IDA Pro

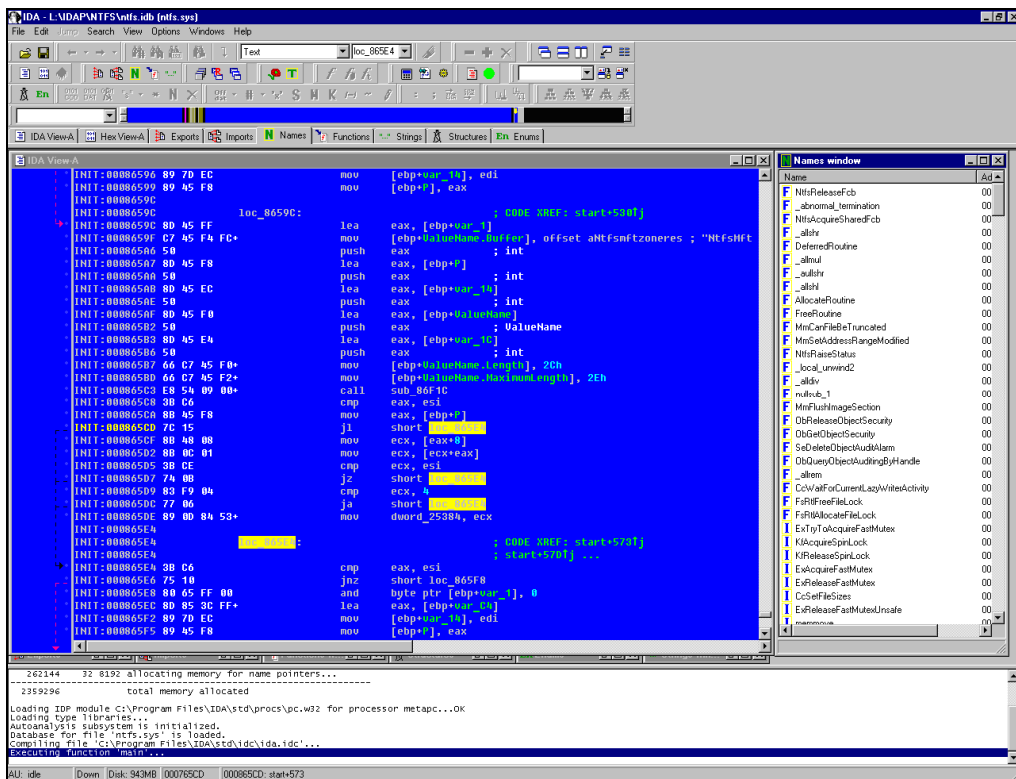


Рис. 1.8. Графическая версия IDA Pro

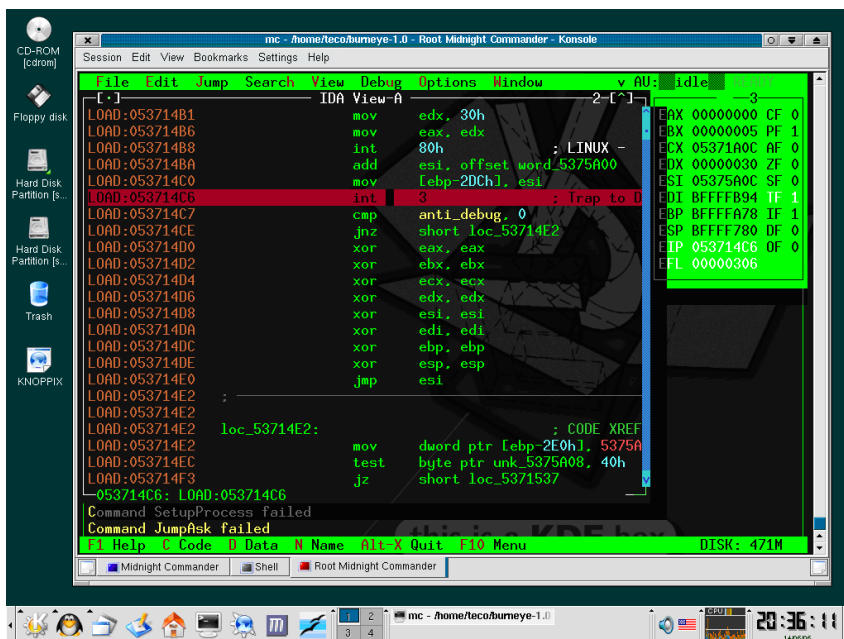


Рис. 1.9. IDA Pro под Linux

В последних версиях IDA Pro сделаны определенные подвижки в сторону автоматической распаковки файлов и снятия обфускаторов³. Внушительную коллекцию плагинов и скриптов можно найти как на официальном сайте, так и на сайте <http://www.openrce.org>.

Конкурирующие продукты не выдерживают никакого сравнения с IDA Pro. Тем не менее, народ активно качает бесплатный (ныне заброшенный) дизассемблер с функциями отладчика — W32Dasm (<http://www.wasm.ru/baixado.php?mode=tool&id=178>) и, судя по всему, остается доволен (рис. 1.10).

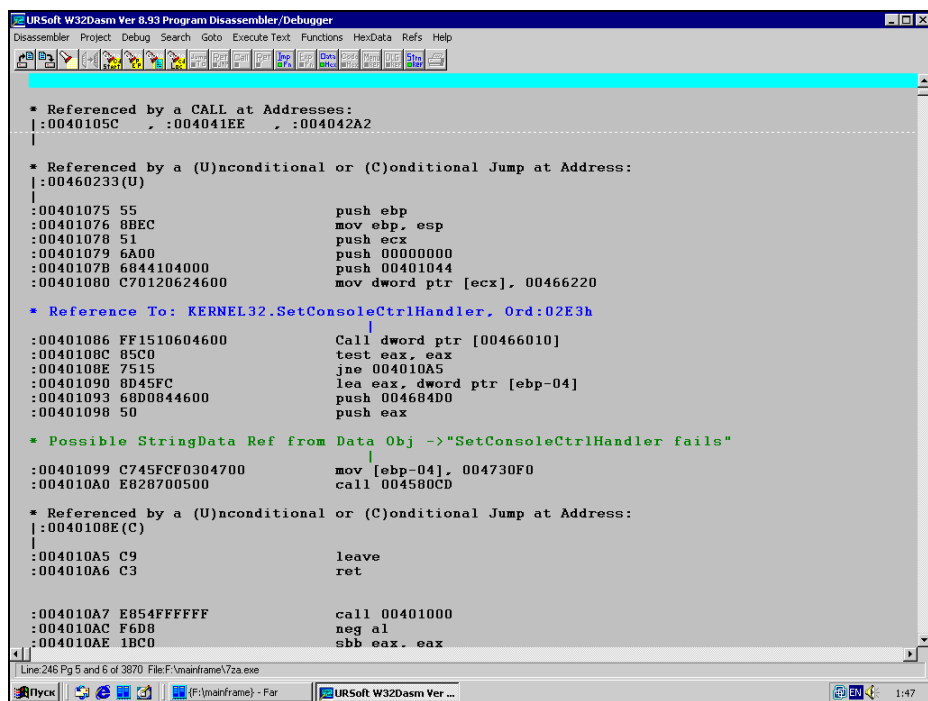


Рис. 1.10. Дизассемблер WDASM

Остальные дизассемблеры обладают еще более ограниченными возможностями, и поэтому здесь не рассматриваются. Единственный из такого рода продуктов, заслуживающий упоминания — это Hacker Disassembler Engine (<http://patkov-site.narod.ru/lib.html>), представляющий собой дизассемблер длин, распространяющийся в исходных текстах и предназначенный для встраивания в различные хакерские программы, занимающиеся перехватом функций, автоматической распаковкой, генерацией полиморфного кода и т. д.

Декомпиляторы

Декомпиляцией называется процесс получения исходного текста программы (или чего-то очень на него похожего) из двоичного файла. В полном объеме декомпиляция принципиально невозможна, поскольку компиляция — однонаправленный процесс, причем с потерей данных. Однако декомпиляторы все-таки существуют и со своей задачей достойно справляются.

³ Более подробную информацию о распаковке файлов, снятии протекторов и обфускаторов можно найти в *части V* данной книги.

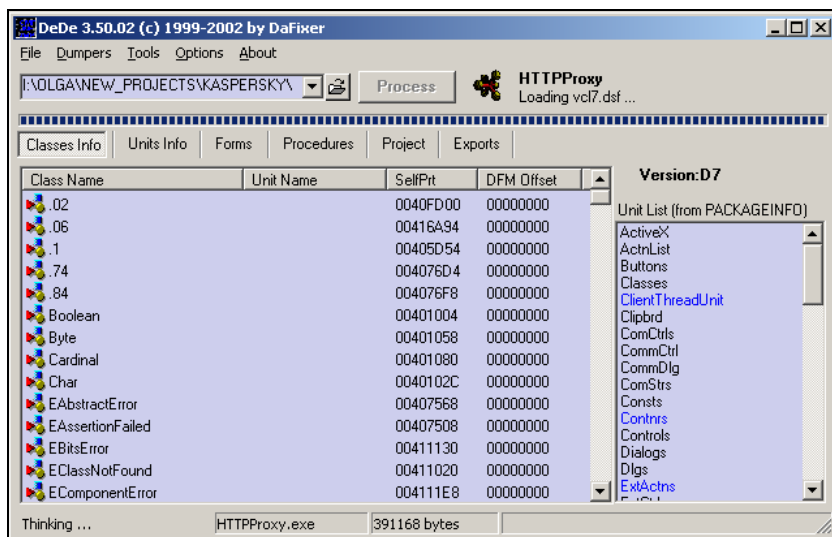


Рис. 1.11. Декомпилятор DeDe ломает HTTPProxy

Для программ, написанных на Delphi и Borland Builder с использованием RTTI, возможно восстановить исходную структуру классов вплоть до имен функций-членов, а также реконструировать формы и "вычислить" адреса обработчиков каждого из элементов. Допустим, у нас имеется диалоговое окно **Registration** с кнопкой **OK**, и мы хотим знать, какая процедура считывает серийный номер и что она с ним делает. Нет ничего проще! Берем бесплатный декомпилятор DeDe (<http://programmerstools.org/node/120>), декомпилируем программу и вперед (рис. 1.11)!

Для Visual Basic существуют свои декомпиляторы, лучшим из которых считается VB Decompiler от GPcH (<http://www.vb-decompiler.org/index.php?p=Products>). Другие бейсик-декомпиляторы, в том числе VB RezQ (<http://www.vbrezq.com/>), VBDE (<http://programmerstools.org/node/129>) и Spices.Decompiler (<http://programmerstools.org/node/635>) также полезно положить в свой хакерский чемоданчик.

Особый интерес представляют декомпиляторы программ-инсталляторов, поскольку многие проверки (на истечение срока работы демо-версии, на серийный номер или ключевой файл) производятся как раз на стадии инсталляции. Самый популярный инсталлятор — это InstallShield. Для него имеется множество удобных декомпиляторов. Вот только некоторые из них:

- ☐ InstallShield X Unpacker (<http://programmerstools.org/node/154>)
- ☐ Windows InstallShield Decompiler (<http://programmerstools.org/node/118>)
- ☐ InstallShield Decompiler (<http://programmerstools.org/node/114>)
- ☐ isDcc (<http://programmerstools.org/node/115>)

Что же касается Java и платформы .NET, то с ними замечательно справляется IDA Pro. Если же у вас нет под рукой IDA Pro, можно воспользоваться специализированными декомпиляторами, которые можно найти на сайтах <http://www.cracklab.ru> и <http://www.wasm.ru> вместе с декомпиляторами Fox Pro, Clipper и прочей экзотики.

Шестнадцатеричные редакторы

Давным-давно шестнадцатеричные редакторы (hex-редакторы) представляли собой простые программы, умеющие лишь отображать двоичные файлы в шестнадцатеричном виде и править бай-

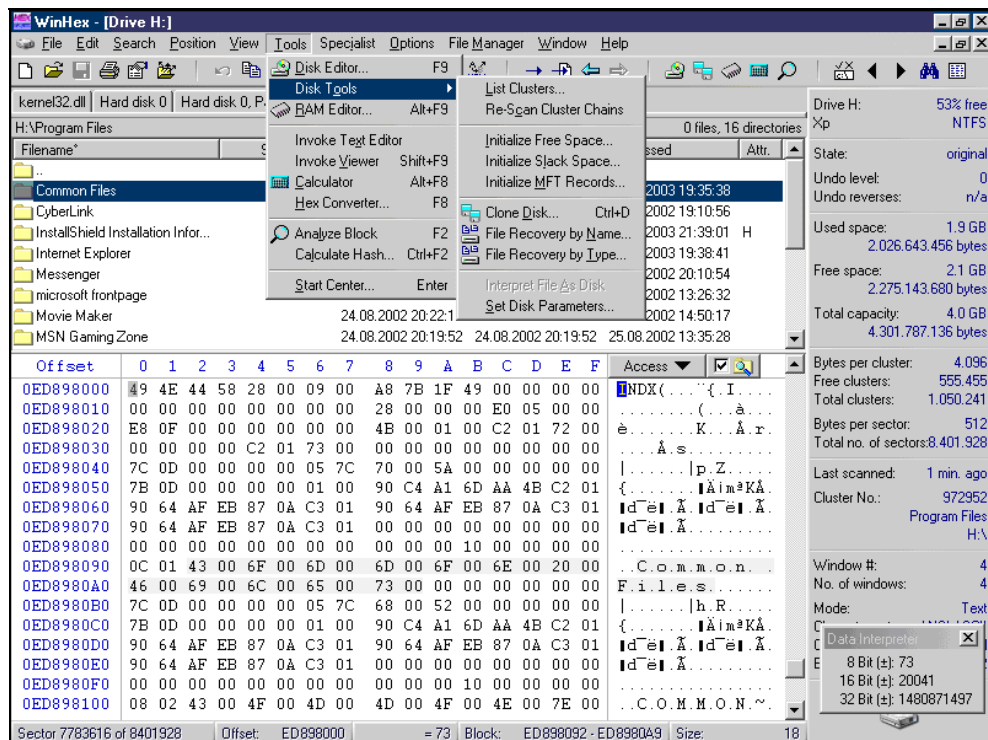


Рис. 1.14. Коммерческий хек-редактор WinHex

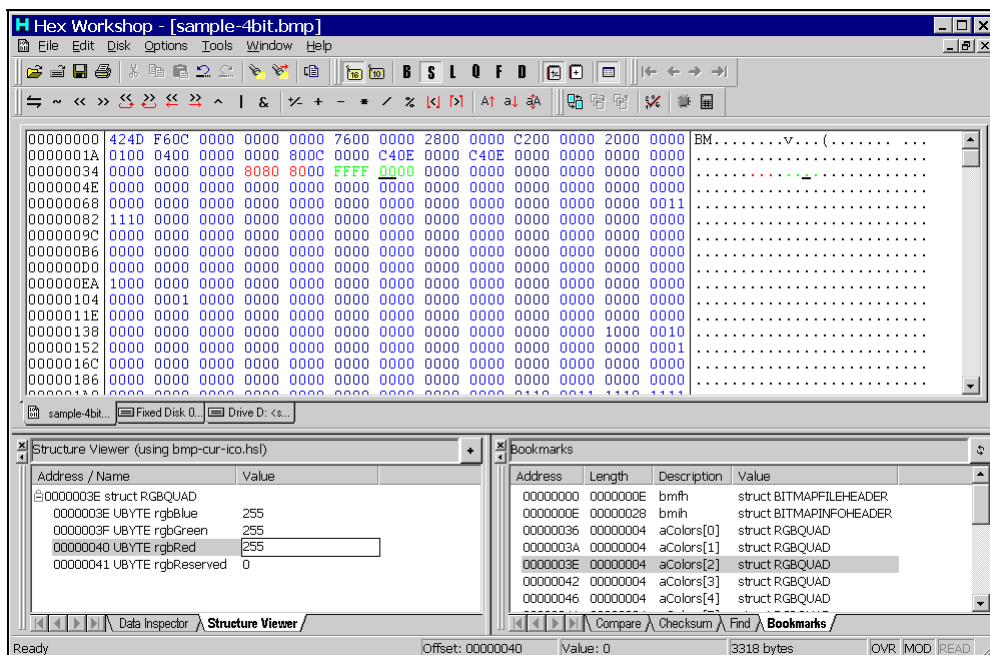


Рис. 1.15. Коммерческий хек-редактор Hex Workshop

Другой хороший редактор, по своим возможностям не только не уступающий HIEW, но даже превосходящий его, — это HTE (<http://hte.sourceforge.net>), распространяющийся в исходных кодах на бесплатной основе. В отличие от HIEW, HTE позволяет выбирать способ ассемблирования инструкции (если инструкция может быть ассемблирована несколькими путями), а также поддерживает мощную систему перекрестных ссылок, вплотную приближающую его к бесплатной версии IDA Pro (рис. 1.13).

Западные хакеры очень любят коммерческие шестнадцатеричные редакторы вроде WinHex (<http://www.winhex.com/winhex/index-m.html>) и Hex Workshop (<http://www.bpssoft.com>). Что привлекательного они в них нашли — непонятно. Ни WinHex (рис. 1.14), ни Hex Workshop (рис. 1.15) не содержат никаких ассемблеров или дизассемблеров, причем маловероятно, что эти функции появятся в дальнейшем. Единственное положительное качество этих редакторов, которое можно отметить, — это наличие калькулятора контрольных и хэш-сумм (например, CRC16, CRC32, MD5, SHA-1), что в некоторых случаях оказывается очень удобным.

Распаковщики

Все больше и больше программ распространяются в упакованном виде (или защищаются протекторами, что еще хуже). Как результат, непосредственное дизассемблирование таких программ становится невозможным. Наконец, поскольку многие упаковщики/протекторы содержат антиотладочные приемы, то страдает и отладка.

Попытки создать универсальный распаковщик многократно предпринимались еще со времен MS-DOS. Всякий раз эти попытки оканчивались полным провалом, поскольку разработчики защит придумывали новую гадость. Тем не менее, в состав большинства хакерских инструментов (IDA Pro, OllyDbg) входят универсальные распаковщики, справляющиеся с несложными защитами. Что касается сложных защит, то, столкнувшись с одной из них, хакер вынужден распаковывать защищенный файл вручную. В *части V* данной книги этот вопрос будет рассмотрен более подробно. Пока же отметим, что когда же один и тот же упаковщик встречается хакеру десятый раз кряду, он садится за написание автоматического или полуавтоматического распаковщика, чтобы облегчить себе работу. Коллекции таких утилит собраны на сайтах <http://www.exetools.com/unpackers.htm>, <http://programmerstools.org/taxonomy/term/16>, <http://www.woodmann.com/crackz/Packers.htm>. Основная проблема состоит в том, что каждый такой распаковщик рассчитан на строго определенную версию упаковщика/протектора и с другими работать просто не может! Чем чаще обновляется упаковщик/протектор, тем сложнее найти подходящий распаковщик, поэтому лучше полагаться только на самого себя.

Кстати, прежде чем искать распаковщик, неплохо бы для начала выяснить: чем же вообще защищена ломаемая программа? В этом поможет бесплатная утилита PEiD (<http://peid.has.it>), содержащая огромную базу сигнатур. Хотя эта программа довольно часто ошибается или дает расплывчатый результат, но, тем не менее, это все-таки лучше, чем совсем ничего.

Дамперы

Снятие дампа с работающей программы — универсальный способ распаковки, позволяющий справиться практически с любым упаковщиком и большинством протекторов. Правда, над полученным дампом еще предстоит как следует поработать, поэтому такие дампы рекомендуется использовать лишь для дизассемблирования. Сломанная таким образом программа может работать неустойчиво, периодически падая в самый ответственный момент.

Какие же дамперы имеются в вашем распоряжении? Первой из утилит этого класса (и самой неуклюжей) была программа ProcDump (<http://www.fortunecity.com/millennium/firemansam/962/html/procdump.html>). Затем появился дампер Lord PE (<http://www.softpedia.com/get/Programming/File-Editors/LordPE.shtml>), учитывающий горький опыт своего предшественника и способный

сохранять дампы даже в тех случаях, когда заголовок файла PE умышленно искажен защитой, а доступ к некоторым страницам памяти отсутствует (атрибут `PAGE_NOACCESS`). Венцом эволюции стал дампер PE Tools (рис. 1.16), базовый комплект поставки которого можно найти практически на любом хакерском сервере, например, на WASM (<http://www.wasm.ru/baixado.php?mode=tool&id=124>) или на CrackLab (<http://www.cracklab.ru/download.php?action=get&n=MTU1>), а свежие обновления лежат на "родном" сайте проекта <http://petools.org.ru/petools.shtml>.

ПРИМЕЧАНИЕ

"Родной" сайт проекта часто меняет свой адрес.

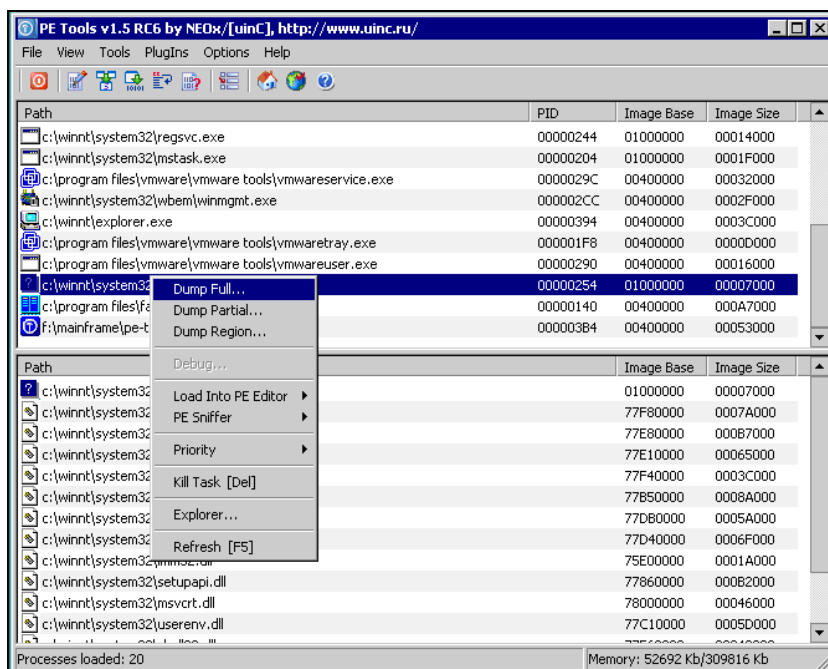


Рис. 1.16. PE Tools — один из лучших дамперов PE-файлов

После снятия дампа необходимо, как минимум, восстановить таблицу импорта, а иногда еще и таблицу перемещаемых элементов вместе с секцией ресурсов. Таблицу импорта лучше всего восстанавливать знаменитой утилитой Import REConstructor, которую вместе с утилитой ReloX (восстанавливающей таблицу перемещаемых элементов) и минимально работающим универсальным распаковщиком можно найти по адресу: <http://wave.prohosting.com/mack/main.htm>. А вот здесь лежит коллекция программ для восстановления таблицы ресурсов: <http://www.wasm.ru/baixado.php?mode=tool&id=156>. Если же ни одна из этих утилит не справится со своей задачей, то, быть может, поможет бесплатная программа Resource Binder: <http://www.setisoft.com/ru/redirect.php?dlid=89>.

Редакторы ресурсов

Редактировать ресурсы приходится во многих случаях. Например, чтобы сменить текст диалогового окна, разблокировать элемент управления, заменить логотип и т. д. Формально, редактор ресурсов входит в каждый Windows-компилятор, в том числе и в Microsoft Visual Studio. Вот только