МИНИСТЕРСТВО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Курсовая работа
по дисциплине «Системное Программирование»

Модификация исполняемого файла
на примере расширения функционала игры Age of Empires

Выполнил:
Врублевский Евгений,
студент IV курса гр. 552001

Проверил:
Лещев А.Е.

Минск, 2008

Содержание

Содержание....2
Введение....3
Отказ от ответственности....3
Обзор используемого ПО....4
Дизассемблер IDA....4
Отладчик OllyDbg....4
HEX-редактор Hexplorer....4
Microsoft Macro Assembler....4
Исследуемая программа....5
Общие сведения....5
Файлы программы....6
Модификация....7
Описание существующих проблем....7
Отключение проверки наличия CD....8
Исправление ошибки....9
Поддержка файла конфигурации....10
Поддержка оконного режима....13
Общий результат работы....14
Руководство пользователя....15
Профили....15
Стандартные профили....15
Структура файла конфигурации....15
Выводы....16
Литература....17

Введение

В мире ПО существует огромное количество программ, забытых своими разработчиками. Хорошо, когда уже существует хорошая альтернатива. А если ее нет? В программе может катастрофически не хватать каких-то мелочей, некоторые досадные ошибки могут годами доставлять массу неудобств пользователям, а на новых версиях ОС программа и вовсе может отказаться работать. Далеко не всегда имеются исходные коды, чтобы привести программу в порядок. Если программа простая — не составит труда за короткий срок создать альтернативу. Но если программа большая и сложная, что же делать в таком случае? Не всегда рационально тратить время и деньги на разработку полного аналога, ведь расширить в разумных рамках функциональность и исправить большинство ошибок можно уже в готовом исполняемом файле.

Целью данной работы является демонстрация методик модификации исполняемых файлов на примере реального приложения. В качестве целевой программы выбор пал на игру Age of Empires II по многим причинам:

1. программа вышла в 1999 году и имеет статус abandonware — то есть она более не выставляется на продажу производителем и не приносит прибыли
2. последнее обновление выпущено в 2000 году, доработка не производится
3. игра является хорошим примером сложного ПО, полный аналог которого практически невозможно создать заново
4. имеются досадные ошибки, которые не были исправлены разработчиками

Отказ от ответственности

Работа проведена исключительно в образовательных целях, и ни в коем случае не является призывом к нарушению действующего законодательства. Автор не несёт никакой ответственности за незаконное использование представленных материалов.

Обзор используемого ПО

Дизассемблер IDA

IDA — интерактивный дизассемблер который широко используется для реверс-инжиниринга. Он отличается исключительной гибкостью, наличием встроенного командного языка, поддерживает множество форматов исполняемых файлов для большого числа процессоров и операционных систем. Позволяет строить блок-схемы, изменять названия меток, просматривать локальные процедуры в стеке и много другое. Имеет встроенный отладчик x86.

IDA, до определенной степени, умеет автоматически выполнять анализ кода, используя перекрестные ссылки, знание параметров вызовов функций стандартных библиотек, и другую информацию. Однако вся сила его проявляется в интерактивном взаимодействии с пользователем. В начале исследования дизассемблер выполняет автоматический анализ программы, а затем пользователь с помощью интерактивных средств IDA 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.

Отладчик OllyDbg

OllyDbg — 32-битный отладчик уровня ассемблера для операционных систем Windows, предназначенный для анализа и модификации откомпилированных исполняемых файлов и библиотек, работающих в режиме пользователя. OllyDbg выгодно отличается от классических отладчиков (таких, как SoftICE) простым интерфейсом, простотой в установке и запуске.

Отладчик поддерживает процессоры серии 80x86 и совместимые, имеется поддержка расширений MMX, 3DNow! и SSE. OllyDbg распознает более двух тысяч типичных функций Windows API и языка C, что упрощает отладку. Имеются функции для распознания и расшифровки PE-заголовка.

HEX-редактор Hexplorer

Hexplorer — бесплатная программа для редактирования и просмотра двоичных данных в шестнадцатеричном представлении, которое, в большинстве случаев, более удобно и наглядно, чем двоичное. Имеет простой дизассемблер x86, что позволяет удобнее ориентироваться внутри исполняемого файла. Используется непосредственно для ручной модификации двоичного файла без изменения его размера, так называемого патча.

Microsoft Macro Assembler

Microsoft Макро-Ассемблер (сокращённо MASM) — ассемблер для семейства x86 микропроцессоров. Первоначально он был произведен компанией Microsoft для создания программ в их MS-DOS операционной системе. Это поддерживало широкое разнообразие макро-средств обслуживания и структурированность программных идиом, включая конструкции высокого уровня для повторов, вызовов процедур и чередований (поэтому MASM — пример ассемблера высокого уровня). Более поздние версии добавили способность создания программ для Windows операционных систем, которые были выпущены, чтобы последовать за MS-DOS.

Мы будем использовать MASM для генерации «донорских» исполняемых файлов, из которых мы будем извлекать нужные нам фрагменты машинных кодов для их последующей вставки в модифицируемый исполняемый файл.

Исследуемая программа

Общие сведения

Название:	Age of Empires II: The Conquerors
Разработчик:	Ensemble Studios
Издатель:	Microsoft
Год издания:	2000
Версия:	1.0c
Движок:	Genie
Платформа:	Windows

Исследуемая игра написана на движке Genie, разработкой которого с 1997 до 2000 года занималась компания Ensemble Studios. За это время на данном движке было выпущено 4 игры из серии Age of Empires. Далее Genie был лицензирован компании LucasArts, которая в 2001-2002 годах выпустила еще 2 игры на этом движке, но уже из серии Star Wars.

1997	Age of Empires
1998	Age of Empires: The Rise of Rome
1999	Age of Empires II: The Age of Kings
2000	Age of Empires II: The Conquerors
2001	Star Wars: Galactic Battlegrounds (LucasArts Entertainment)
2002	Star Wars: Clone Campaigns (LucasArts Entertainment)

Игры на движке Genie

Однако, к этому времени движок уже сильно устарел, поскольку работал в режиме 256 цветов, что очень скромно по сравнению с играми даже 2000 года. Именно поэтому доработка движка была остановлена, новых игр на его основе не выпускалось.

Название:	Genie
Разработчик:	Ensemble Studios
Тип:	Изометрический, для стратегий реального времени
Видео-режим:	256 цветов

Характеристики Genie

Поскольку все приведенные игры созданы на базе одного и того же движка, практически все изменения для одной игры из списка без особых проблем портируются на другие, что будет продемонстрировано в этой работе.

Файлы программы

Игровой движок Genie загружает всю игровую графику и множество собственных настроек из внешних файлов. Стоит немного разобраться в этом, чтобы стало ясно, что мы можем изменить в игре, не затрагивая при этом исполняемый файл. Для начала рассмотрим файлы, которые находятся в корневом каталоге игры.

age2_x1.exe	Исполняемый файл игры Age of Empires II: The Conquerors
empires2.exe	Исполняемый файл игры Age of Empires II: The Age of Kings
ebueulax.dll	Библиотека, отвечающая за вывод лицензионного соглашения
language.dll	Языковой файл Age of Empires II: The Age of Kings
language_x1.dll	Языковой файл Age of Empires II: The Conquerors
language_x1_p1.dll	Языковой файл Age of Empires II: The Conquerors v1.0c
player.nfp	Информация о игроках

Как видно, в корне находится 2 исполняемых файла — для оригинальной игры (empires2.exe), а так же для дополнения (age2_x1.exe). Дополнение проверяет существование файла empires2.exe, и не запускается без его.

Теперь рассмотрем структуру каталогов прогрммы.

ai	Скрипты искусственного интеллекта
campaign	Файлы игровых кампаний
data	Графика, звук, настройки движка
history	История цивилизаций, присутсвующих в игре
learn	Файлы игроков
random	Скрипты случайных карт
savegame	Сохраненные игры
scenario	Пользовательские сценарии
screenshots	Снимки игровых экранов
sound	Музыка, речь из кампаний
taunt	Голосовые сообщения для сетевой игры

Здесь больше всего нас интересует каталог data, в котором хранятся все служебные данные игры.

empires2_x1_p1.dat	Настройки движка Genie (цивилизации, здания и юниты)
graphics.drs	Игровая графика
interfac.drs	Графика и звуки интерфейса
sounds.drs	Звуки (которые издают игровые юниты)

Как видно из таблицы, все игровые юниты загружаются из файла, и для того, чтобы изменить баланс игры (действия новых технологий, характеристики юнитов, возможности строений) нет необходимости затрагивать исполняемый файл. Для этих целей энтузиастами создан специальный редактор GeniEd2. Для изменения графики и звуков также существуют специальные редакторы, самый мощный из которых — это Mod Pack Studio.

Мы же не будем этим заниматься. Нас интересуют внутренности исполняемого файла игры, исследованием которого мы и займемся.

Модификация

Описание существующих проблем

1. Проверка CD. В современных сверх-тонких и компактных ноутбуках уже не редкость, когда нет встроенного CD привода. Более того, поскольку игра очень стара, оригинальный CD может перестать читаться, а кто-то и вовсе мог потерять его. В итоге владельцы вполне легальной копии игры теряют возможность запускать игру. Поэтому необохимо убрать проверку наличия CD.

2. Отсутсвие поддержки широкоформатных разрешений. Сейчас особую популярность получают широкоформатные LCD мониторы, поскольку они отличаются более привлекательной ценой, по сравнению с обычными 4:3 мониторами. Однако, Age of Empires поддерживает 3 фиксированных разрешения: 800×600, 1024×768 и 1280×1024. С такими разрешениями картинка на широкоформатных мониторах выглядит растянутой, что доставляет дискомфорт. Добавить поддержку новых разрешений в игру можно, но так как в игре для каждого разрешения используются своя графика интерфейса, оставим это на потом, и ограничимся пока что добавлением оконного режима работы в игру.

3. Для включения каких-то системных игровых опций, их приходится передавать через командную строку при каждом запуске программы. Можно поместить все параметры в ярлык, но при перемещении игры в другой каталог, ярлык перестает работать. Выход из ситуации — добавление поддержки конфигурационных файлов.

4. Игра не переносима. Для запуска на другом компьютере недостаточно скопировать игровые файлы, необходимо пройти весь процесс установки игры. В связи с широким распространением внешних накопителей хотелось бы наделить игру этим свойством.

5. Ошибки в интерфейсе. Например, в диалоге подключения к IP игре в списке последних игр обрезаются ранее введенные адреса, что сводит удобство всей функции быстрого подключения на нет, приходится каждый раз вводить адрес подключения вручную.

Итак, сформируем цели:

1. Отключить проверку наличия CD
2. Добавить поддержку оконного режима
3. Добавить поддержку конфигурационных файлов
4. Сделать игру переносимой
5. Исправить известные ошибки

Отключение проверки наличия CD

Найти код, который отвечает за проверку наличия CD оказалось очень просто. Достаточно проследить вызовы функций GetDriveTypeA и GetVolumeInformationA, которые используются для получения информации о приводе, как мы обнаружим функцию проверки наличия CD по адресу 4485A0h (базовый адрес 400000h, то есть физически в файле машинный код находится по адресу 485A0h), которая при удачной проверке возвращает в eax единицу.

Дизассемблированный код процедуры проверки наличия диска

```
CheckCD         proc near
      sub     esp, 214h
      push    ebx
      push    esi
      ; ...
      ; множество проверок
      ; ...
      test    eax, eax
      jz cd_check_fail
cd_check_ok:
      pop     esi
      mov     al, 1
      pop     ebx
      add     esp, 214h
      retn    4
cd_check_fail:
      pop     esi
      pop     ebx
      add     esp, 214h
      retn    4
CheckCD         endp
```

При этом стоит заметить, что эта функция запрашивает значение CDPath из реестра, где хранится буква диска, с которого была установлена игра. То есть игра привязывается конкретно к той букве диска, с которого она была установлена, и при наличии оригинального диска в другом приводе игра все равно не запустится. Наиболее простое решение в данном случае — это заменить команду jz cd_check_fail, которая занимает 2 байта, на 2 операции nop, чтобы после проверки диска в eax всегда возвращалась единица. Однако, функция устроена таким образом, что при отсутсвии значения CDPath в реестре, функция сразу возвращает 0, что мешает переносимости программы. Поэтому всю функцию мы заменим на 3 простые команды, которые мы скомпилируем в MASM для получения машинного кода:

Ассемблерный код	Машинный код
CheckCD proc near	
xor eax, eax	33 C0
inc eax	40
retn 4	C2 04 00
CheckCD endp	

То есть нам нужно поместить эти 6 байт в начало функции определения наличия CD, а остальные команды заменить на nop (код 90h).

Исходный машинный код	Полученный машинный код
8B 44 24 04 81 EC 14 02 00 00 85 C0 53 56 8B D9	33 C0 40 C2 04 00 90 90 90 90 90 90 90 90 90 90
7E 55 55 33 ED 57 33 F6 8B 8B B8 01 00 00 8D 7E	90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
...	...

В обычном HEX редакторе заменяем начиная с смещения 485A0h указанные выше машинные коды, после чего игра больше никогда не требует для запуска CD, даже если в реестре нет необходимого ключа. При этом мы освободили 288 байт для дополнительного машинного кода, что пригодится нам в дальнейшем.

Исправление ошибки

При подключении к сетевой игре по IP игра выводит список IP и названий игр, к которым игрок подключался ранее. Однако, по каким-то причинам IP адрес часто сохранялся не полностью, обрезалось несколько его последних символов. Необходимо разобраться в чем дело и исправить ошибку.

Займемся поиском кода, отвечающего за сохранение IP адресов последних игр в реестре. Нам известно, что игра сохраняет список последних IP игр в ключах с именами вида RecentGameName%d и RecentGameAddr%d. При помощи IDA находим по 3 ссылки на эти строки. Определить где нужный код очень просто — в коде со смещением 0512116h происходит запись этих ключей, в остальных случаях — считывание. Не будем приводить найденный участок машинного кода — там около 50 инструкций, общий смысл которых можно представить в виде маленького фрагмента кода на C.

Код сохранения списка последних IP игр

```
for(int i = 0; i < count; i++)
{
	char* keyname;
	sprintf(keyname, "RecentGameName%d", i);
	RegSetVal(1, keyname, gamename[i], strlen(gamename[i]));
	sprintf(keyname, "RecentGameAddr%d", i);
	RegSetVal(1, keyname, gameaddr[i], strlen(gamename[i]));
}
```

Для установки значения в реестре игрой используется собственная функция-обертка RegSetVal(int, char* ValueName, char* Data, int DataSize) — название функции и параметров конечно же придуманы, в машинном коде их нет. Как видно, игра сохраняет в реестре ровно столько символов IP адреса, сколько их в названии сетевой игры. Судя по всему, программист скопировал 2 строчки сохранения названия игры для сохранения адреса, но при этом забыл исправить название массива в вызове strlen.

Для исправления ошибки у нас есть 2 варианта: подставить в strlen правильную строку или вместо strlen помещать константу 16 (максимальная длина строки адреса, например «255.255.255.255»).

Код strlen(gamename[i])

```
call    _sprintf ; +0x1022CF
mov     edi, ebp
or      ecx, 0FFFFFFFFh
xor     eax, eax
add     esp, 0Ch
repne scasb
not     ecx
push    ecx
```

Компилятор оптимизировал код и заменил вызов функции strlen() вычислением длины на месте. В предыдущем вызове strlen() использовалась та же строка, в вот опять оптимизация — указатель помещается в ebp один раз при первом вызове. То есть у нас нет драгоценных пары байт для того, чтобы поместить правильный указатель в ebp. Можно, конечно, сделать jmp за пределы функции, потом обратно — но это не самый красивый метод. Поэтому остановимся пока на самом простом варианте исправления ошибки — будем всегда сохранять 16 байт адреса. Для этого заменим приведенный выше машинный код на следующий:

Ассемблерный код	Машинный код
`call _sprintf ; +0x1022CF`	`E8 CF 22 10 00`
`add esp, 0Ch`	`83 C4 0C`
`mov ecx, 0Fh`	`B9 0F 00 00 00`

Поддержка файла конфигурации

Игра поддерживает множество полезных параметров командной строки, которые было бы удобно сохранить где-нибудь в файле, чтобы не передавать их каждый раз при запуске. Лучший способ реализовать подобное — это заставить игру при каждом старте загружать внешнюю библиотеку и выполнять функцию инициализации. При тако организации можно добавить множество полезных вещей в конфигурационные файлы, такие как: загрузка дополнительных библиотек (модулей) без необходимости модификации исполняемого файла, загрузка шрифтов при старте, перехват функций работы с реестром для организации хранения всех игровых данных в конфигурационном файле игры, а не в реестре. Не лишним будет поддержка множества профилей с различными наборами настроек с возможностью быстрого их переключения. Итак, приступим.

Наиболее удобным местом подключения внешней библиотеки оказалась функция загрузки библиотеки ebueulax.dll, отвечающей за отображение лицензионного соглашения. Она как раз вызывается практически сразу после начала работы WinMain, однако помимо загрузки библиотеки ebueulax.dll и выполнения функции EBUEula она делает много лишних действий, поэтому мы полностью перепишем ее, чтобы она загружала библиотеку config.dll и выполняла библиотечную функцию LoadConfig с указателем на расположение командной строки в памяти (для её изменения) в качестве параметра.

Ассемблерный код		Машинный код
LoadConfigLib	proc near	
sub	esp, 208h	81 EC 08 02 00 00
push	esi	56
push	edi	57
push	ebx	53
push	ecx	51
push	edx	52
mov	esi, ds:LoadLibraryA	8B 35 E0 51 63 00
push	67B8C4h; lpLibFileName	68 C4 B8 67 00
call	esi ; LoadLibraryA	FF D6
mov	ebx, eax	8B D8
test	ebx, ebx	85 DB
jz	short loc_587443	74 25
push	offset aLoadconfig	68 B8 B8 67 00
push	ebx ; hModule	53
call	ds:GetProcAddress	FF 15 C8 50 63 00
mov	edi, eax	8B F8
test	edi, edi	85 FF
jz	short loc_58743C	74 0C
mov	eax, [esp+220h];**args	8B 84 24 20 02 00 00
push	eax	50
call	edi	FF D7
jmp	short loc_587443	EB 07
loc_58743C:		
push	ebx ; hLibModule	53
call	ds:FreeLibrary	FF 15 20 51 63 00
loc_587443:		
pop	edx	5A
pop	ecx	59
pop	ebx	5B
pop	edi	5F
pop	esi	5E
add	esp, 208h	81 C4 08 02 00 00
retn		C3
LoadConfigLib	endp	

Приведенный код делает несколько вызовов библиотечных функций. Их адреса были подставлены вручную в машинный код.

Заменяем исходную функцию (начинается со смещения 187400h) новой при помощи обычного HEX-редактора. Поскольку новая функция значительно короче исходной, не забываем заменить оставшиеся байты кодом операции nop — 90h. Мы получили еще 192 байта свободных байта для дополнительного машинного кода. Это место можно будет использовать в дальнейшем.

Исходный машинный код

```
81 EC 08 02 00 00 53 56 8B 35 E0 51 63 00 57 68
CC B8 67 00 FF D6 8B D8 85 DB 75 0A 5F 5E 5B 81
C4 08 02 00 00 C3 68 C4 B8 67 00 53 FF 15 C8 50
63 00 8B F8 85 FF 75 13 53 FF 15 20 51 63 00 5F
5E 33 C0 5B 81 C4 08 02 00 00 C3 8B 84 24 18 02
00 00 50 FF D6 8B F0 85 F6 75 13 53 FF 15 20 51
...
```

Полученный машинный код

```
81 EC 08 02 00 00 56 57 53 51 52 8B 35 E0 51 63
00 68 C4 B8 67 00 FF D6 8B D8 85 DB 74 25 68 B8
B8 67 00 53 FF 15 C8 50 63 00 8B F8 85 FF 74 0C
8B 84 24 20 02 00 00 50 FF D7 EB 07 53 FF 15 20
51 63 00 5A 59 5B 5F 5E 81 C4 08 02 00 00 C3 90
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90
...
```

Всю остальную работу берет на себя модуль config.dll. Этот модуль написан на языке C++. В качестве формата хранения конфигурации был выбран XML, как наиболее гибкий вариант. Для его разбора используется библиотека TinyXML.

Исходный код config.dll

```
#include <windows.h>
#include <stdio.h>
#include "include/xmlfile.h"
//---------------------------------------------------------------------------
#define NULLTOSTR(str) (str)?(str):""
xmlelem* ForceOpenElement(xmlelem* root, char* nodeName)
{
  if( !root->FirstChild(nodeName) ) root->InsertEndChild(xmlelem(nodeName));
  return root->FirstChildElement(nodeName);
}
char* ForceGetAttribute(xmlelem* root, char* attrName, char* attrDefVal)
{
  if(!root->Attribute(attrName)) root->SetAttribute(attrName, attrDefVal);
  return (char*)root->Attribute(attrName);
}
bool FileExists(LPCTSTR fname)
{
  return ::GetFileAttributes(fname) != DWORD(-1);
}
//---------------------------------------------------------------------------
#pragma argsused
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fwdreason, LPVOID lpvReserved)
{
  return 1;
}
//---------------------------------------------------------------------------
extern "C"
{
char cmdln[2048] = "";
void __export __stdcall LoadConfig(char** oldCmdLine)
{
  // Параметры командной строки
  strcpy(cmdln, *oldCmdLine);
  xmlfile xmlroot ("config.xml");
  if(!xmlroot.LoadFile("config.xml"))
  {
    char defaultxml[] = "<config profile=\"default\"><default>"
      "<module enabled=\"1\" lib=\"wndmode.dll\" />"
      "<cmdline enabled=\"1\" param=\"NoStartup\" />"
```

```
        "</default></config>";
      xmlroot.Parse(defaultxml);
    }
    if(!xmlroot.FirstChild("config"))
      xmlroot.InsertEndChild(xmlelem("config"));
    xmlelem* xmlprof = xmlroot.FirstChildElement("config");
    char* profileName = ForceGetAttribute(xmlprof, "profile", "default");
    // Проверяем установлен ли другой профиль в командной строке
    char* cmdProf = 0;
    if(cmdProf = strstr(strupr(strdup(cmdln)), "PROFILE="))
    {
      cmdProf += strlen("PROFILE=");
      if(strchr(cmdProf, ' ')) strchr(cmdProf, ' ')[0] = 0;
      profileName = strlwr(cmdProf);
      if(strstr(strupr(strdup(cmdln)), "SETPROFILE="))
        xmlprof->SetAttribute("profile", profileName);
    }
    xmlprof = ForceOpenElement(xmlprof, profileName);
    // Загружаем команды из профиля
    xmlnode* node = 0;
    for( node = xmlprof->FirstChild(); node; node = node->NextSibling() )
    {
      xmlelem* elem = node->ToElement();
      if(!node->Value()) continue; // Если у тега нет названия
      int enabled = 0;
      elem->QueryIntAttribute("enabled", &enabled);
      if(enabled)
      {
        char* command = strdup( node->Value() );
        if(strcmpi(command, "module")==0)// Загружаем модуль
        {
          if( FileExists(NULLTOSTR(elem->Attribute("lib"))) )
            LoadLibrary( NULLTOSTR(elem->Attribute("lib")) );
        }
        // Добавляем параметр командной строки
        else if (strcmpi(command, "cmdline")==0)
        {
          strcat(cmdln, " ");
          char* param = strdup(NULLTOSTR(elem->Attribute("param")));
          strcat(cmdln, param);
          free(param);
        }
        // Загружаем шрифт
        else if (strcmpi(command, "font")==0)
        {
          if( FileExists(NULLTOSTR(elem->Attribute("file"))) )
            AddFontResource( NULLTOSTR(elem->Attribute("file")) );
        }
        free(command);
      }
    }
    *oldCmdLine = (char*)&cmdln;
    xmlroot.SaveFile();
  }
  } // extern "C"
```

В данной версии модуля реализовано:

1. Система профилей
2. Загрузка настроек командной строки из файла
3. Загрузка дополнительных модулей
4. Загрузка шрифтов при запуске программы

Поддержка оконного режима

Поддержка оконного режима организована в виде отдельного модуля wndmode.dll, загрузкой которого занимается config.dll. Этот модуль занимается тем, что перехватывает все вызовы DirectDraw, изменяет параметры таким образом, чтобы программа работала в окне, и только после этого передаёт управление оригинальным функциям DirectDraw. Рассмотрим существующие методы перехвата вызовов внешних функций.

Локальный перехват с использованием раздела импорта

Локальный перехват может быть реализован и в Win9X, и в WinNT посредством подмены адреса перехватываемой функции в таблице импорта. Для понимания механизма работы этого метода нужно иметь представление о том, как осуществляется динамическое связывание. В частности, необходимо разбираться в структуре раздела импорта модуля.

В разделе импорта каждого exe- или DLL-модуля содержится список всех используемых DLL. Кроме того, в нем перечислены все импортируемые функции. Вызывая импортируемую функцию, поток получает ее адрес фактически из раздела импорта. Поэтому, чтобы перехватить определенную функцию, надо лишь изменить её адрес в разделе импорта. Для того чтобы перехватить произвольную функцию в некотором процессе, необходимо поправить её адрес импорта во всех модулях процесса (так как процесс может вызывать эту функцию не только из exe-модуля, но и из DLL-модулей). Кроме того, процесс может воспользоваться для загрузки DLL функциями LoadLibraryA, LoadLibraryW, LoadLibraryExA, LoadLibraryExW или, если она уже загружена, определить её адрес при помощи функции GetProcAddress. Поэтому для перехвата любой API-функции необходимо перехватывать и все эти функции.

Локальный перехват путем изменения перехватываемой функции

Данный метод перехвата основан на следующем: первые несколько байт перехватываемой функции заменяются на команду безусловного перехода к функции перехвата. Этот трюк достаточно просто реализуется в WinNT (как я уже упоминал, в WinNT для каждого процесса создается своя копия образов системных библиотек), но практически нереализуем в Win9X (так как в Win9X если и можно внести изменения в образ системной библиотеки, то только в адресных пространствах всех процессов сразу).

Для подобного перехвата используются:

1. целевая функция (target function) – функция, перехват которой осуществляется;
2. функция-перехватчик (detour function) – функция, замещающая перехватываемую;
3. функция-трамплин (trampoline function) – функция, состоящая из заголовка целевой функции и команды перехода к остальному коду целевой функции.

В модуле wndmode.dll используется второй вариант перехвата, поскольку он более удобен и надежен, нежели первый. Сам по себе модуль wndmode.dll является сильно модифицированной версией библиотеки d3dhook.dll, где реализовано:

1. Полная независимость от программы D3D Windower
2. Настройки загружаются из секции [WINDOWMODE] файла wndmode.ini
3. Настройки по умолчанию заменены для совместимости с Genie
4. Добавлен параметр Border, который включает/выключает рамку вокруг окна
5. Если игровое разрешение равно системному, автоматически убирается рамка

Общий результат работы

Работа по модификации проделана огромная, здесь мы рассмотрели лишь некоторые изменения, поскольку описание даже небольших изменений требует огромных усилий. Результат проделанной работы можно представить в виде списка:

1. Вместо ebueulax.dll (отображение лицензии) загружается config.dll (поддержка конфигурационных файлов)
2. Оконный режим работы (модуль wndmode.dll — модифицированный d3dhook.dll)
3. Код функций работы с курсором адаптирован для работы в оконном режиме (решает много неприятных проблем, вызванных тем, что игра не рассчитана на работу в окне)
4. Игра корректно работает без ее предварительной установки (достаточно скопировать файлы игры), для игры не нужен CD
5. Исправлена ошибка оригинальной игры, когда при игре по IP в списке последних игр обрезались IP адреса
6. Изменен порядок и расположение кнопок в меню «Один игрок» и «Редактор», исправлена ошибка с выделением кнопок в меню «Один игрок» (переписан внушительный участок кода)
7. Исполняемый файл может запускаться из любого подкаталога, а не только из age2_x1 или корня (практически полностью переписано начало процедуры WinMain)
8. Нет проверки наличия файла empires2.exe (его можно удалить)
9. При включенном параметре MIDIMUSIC после сворачивания мелодия начинается заново (не сбрасываются инструменты)

Все изменения работают стабильно, никаких сбоев не обнаружено. В оконном режиме осталась пока что нерешенной проблема отображения системного курсора одновременно с игровым после того, как он покинет область игрового окна и вернется назад. Однако, это вызвано лишь только тем, что игровой движок не рассчитан на запуск в окне и не предусматривает вариант, когда курсор покинет игровое окно. В ближайшей версии модификации это будет исправлено.

Логическим завершением развития данной модификации может стать внедрение в игру поддержки игровых комнат с созданием соответсвующего игрового сервера, поскольку официальные игровые комнаты Age of Empires на Zone были закрыты ещё несколько лет назад. Всё это вполне возможно реализовать, не имея при этом на руках никаких исходных кодов.

Руководство пользователя

Профили

В файле конфигурации может находиться несколько различных наборов настроек, каждый из которых называется профилем.

Если вы хотите сменить профиль конфигурации, запустите игру с параметром Profile=имя_профиля — для временной установки нового профиля (на один запуск); с параметром SetProfile=имя_профиля — для постоянной установки нового профиля (на текущий и все последующие запуски игры).

Стандартные профили

1. fullscreen — полноэкранный режим. Стандартный режим оригинальной версии игры. Все игровые меню и сама игра работают в полноэкранном режиме.

2. default — управляемый оконный режим. Все игровые меню работают в оконном режиме. Если вы выберете в игре разрешение аналогичное системному, игровой процесс будет работать в полноэкранном режиме (без смены системного разрешения), иначе — в оконном.

3. windowed — принудительный оконный режим. Все игровые меню и сама игра работают только в оконном режиме.

Структура файла конфигурации

Пример файла config.xml

```
<config profile="default">
    <default>
        <module enabled="1" lib="wndmode.dll" />
        <cmdline enabled="1" param="NoStartup" />
            <cmdline enabled="0" param="NormalMouse" />
    </default>
    <fullscreen>
        <cmdline enabled="1" param="NoStartup" />
    </fullscreen>
</config>
```

Вся информация в файле config.xml, как видно из расширения, хранится в формате XML. Корневой элемент должен иметь имя config. Атрибут profile указывает название профиля, используемого по умолчанию. В примере по умолчанию будет использоваться профиль default. Внутрь корневого элемента включены элементы с названиями, соответствующими именам профилей, внутри которых соответственно включены все настройки каждого профиля. В примере 2 профиля — default и fullscreen.

Каждый из них содержит в себе собственный набор опций. Каждая элемент опции в качестве имени использует свой тип и имеет один обязательный атрибут enabled, который включает или выключает текущую опцию. То есть если enabled="0", опция будет проигнорирована, иначе — будет выполнена. Поддерживаемые типы опций: cmdline и module.

cmdline предназначен для включения стандартных опций игры, которые ранее можно было включать из командной строки. В атрибуте param указывается параметр, который необходимо передать в командную строку запуска игры.

module предназначен для загрузки дополнительных модулей в игру. В атрибуте lib указывается имя модуля, который необходимо загрузить при запуске игры.

Выводы

Целью данной работы была демонстрация того, что модификация и расширение функционала уже откомпилированных программ возможна, и это не только теоретическая возможность, во многих случаях это вполне выполнимая задача. Не глядя на всю видимую сложность подобной «операции» и большим риском из-за небольшой ошибки привести всю программу в полную неработоспособность, при грамотном и аккуратном подходе описанная методика позволяет делать необходимые изменения в программе без потери её стабильности.

Литература

1. Кип Ирвин «Язык ассемблера для процессоров Intel», 912 стр., 2005, Вильямс

2. Методы перехвата API-вызовов в Win32
   http://www.rsdn.ru/article/baseserv/apicallsintercepting.xml