РЕКОМЕНДОВАННАЯ ЦЕНА 360 ₽

06 (185) 2014

ХАКЕР

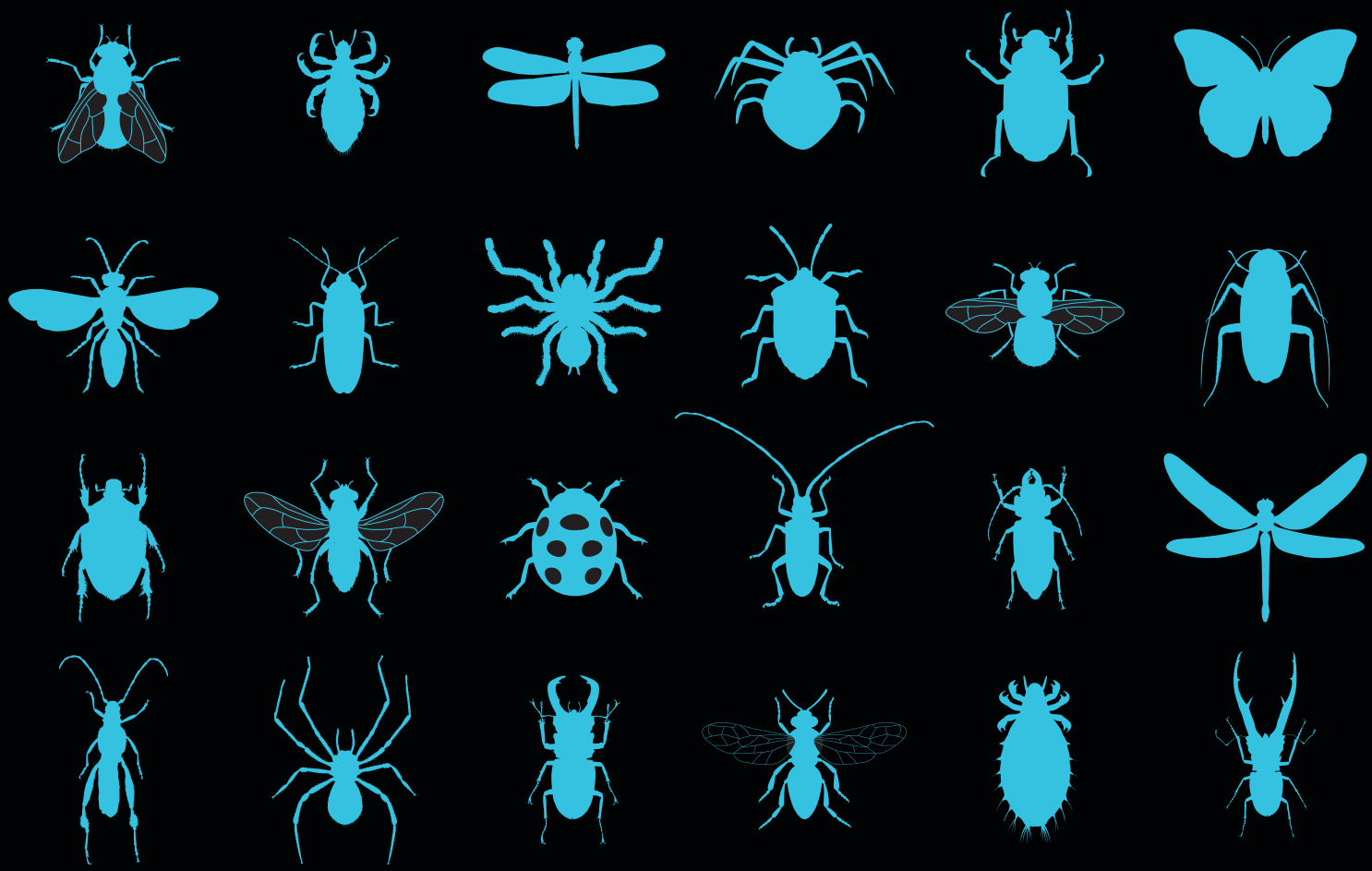
WWW.XAKEP.RU

hi-tun media (game)land
publishing for enthusiasts

4 607157 100063 14006

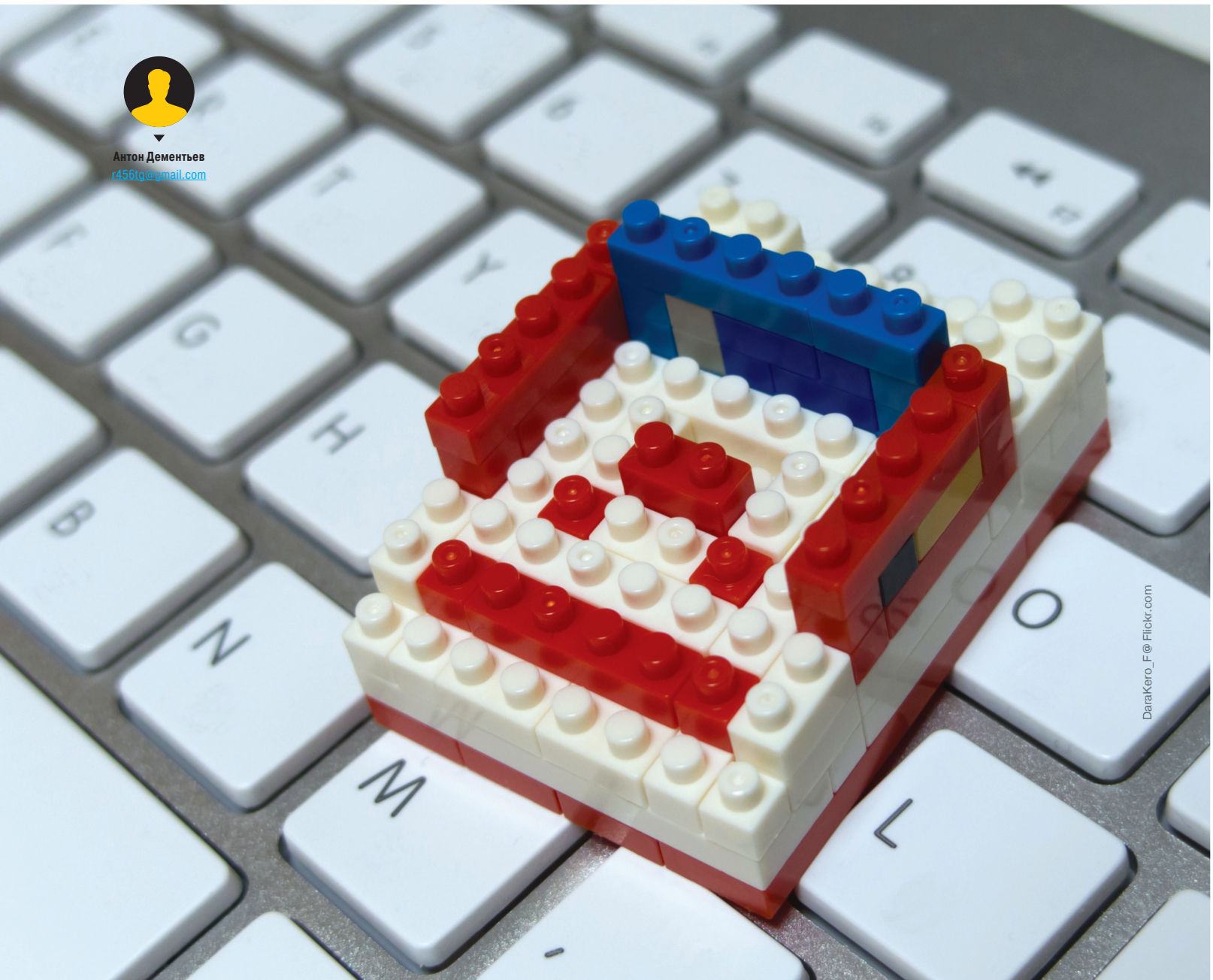
Докажи баг!

Как правильно представлять вендорам нестандартные уязвимости и делать интернет безопаснее



МЕНЯЕМ ПРАВИЛА ИГРЫ

ВВЕДЕНИЕ В РОМХАКИНГ



Антон Дементьев
r456tg@gmail.com

DaraKero_F@ Flickr.com

До появления лазерных дисков и облаков консольные игры выходили на картриджах. Те, кто застал девяностые, наверняка помнят самые популярные в нашей стране игровые приставки Dendy (российский клон японской Famicom) и Sega Mega Drive. Если очень хочется «вернуться к истокам» — всегда можно запустить эмулятор. Но можно пойти и чуть дальше — а что, если покопаться в самой игре? Добро пожаловать в эму-сцену.

Эму-сценой называют сообщество энтузиастов, формирующееся вокруг той или иной приставки. Эти люди ковыряются в своей любимой платформе вдоль и поперек: выкладывают полные дампы картриджей (ROM'ы), придумывают способы для их модификации и добиваются порой самых необычных результатов. Самый распространенный пример — локализация игр. Но можно пойти и дальше. Как тебе история про чувака, который хакал ROM'ы игр для NES, заменяя мужских персонажей на женских (j.mp/1nJg5p9)? А что еще прикажешь делать, если его дочке хочется играть за девочек, а в Donkey Kong такой возможности не предусмотрено?

КРАТКИЙ ЭКСКУРС

Самым главным толчком, послужившим появлению ромхакинга, стала история с игрой Final Fantasy V в 1992 году. Японская компания Squaresoft решила не издавать игру в США, посчитав ее слишком сложной для западных игроков. С появлением эмуляции западные энтузиасты не согласились с японцами и не только сыграли в нее, но и выпустили первый любительский перевод образа картриджа. Официального же перевода на английский не было вплоть до переиздания FFV в конце 1999-го на PlayStation. Именно с перевода Final Fantasy V и появилась русская ромхакинг-сцена, это был дебютный проект группы «Шедевр» в 2001 году.

В 1994 году история отчасти повторилась и с Final Fantasy VI. На этот раз игра вышла в США (под названием Final Fantasy III), через несколько месяцев после релиза в Японии (1994 год). Однако локализация оказалась крайне неудачной: перевод всей игры был выполнен всего одним человеком (Тедом Вулси) в крайне сжатые сроки. Например, в этом переводе были выкинуты скрытые отсылки к дальнейшему развитию сюжета, а смысл одного из предложений был заменен на противоположный из-за неправильно понятого японского крылатого выражения (фраза, близкая по значению к «бизнес испарился», была воспринята Тедом как «бизнес пошел вверх»). Сам Тед оправдывает низкое качество его перевода тем, что ему приходилось делать текст максимально коротким, так как он не влезал в картридж, в то время как японский текст заведомо более компактный. Тем не менее недовольные официальным переводом фанаты с задачей помещения в ром близкого к оригиналу английского текста справились вполне успешно.

ФОРМУЛИРУЕМ УСЛОВИЯ ЗАДАЧИ

Есть одно обстоятельство, которое делает ромхакинг одновременно сложнее и интереснее более популярного моддинга PC-игр. Разработчики современных игр для ПК часто поддерживают моддеров, создавая для них официальные инструменты и выпуская всю необходимую документацию. В случае с консольными играми очевидно, что разработчики никак не предусматривали последующую модификацию своих тайтлов. Поэтому при ромхакинге мы имеем бинарный файл и даже не знаем, по каким адресам и в каком формате хранятся нужные данные, а значит, изначально требуется действовать вслепую, на ощупь.

Ромхакинг может быть как с модификацией машинного кода — языка ассемблера, который у каждой платформы свой (в этом случае если изменения кардинальные, то модифицированный ром или образ диска называют хаком), так и без нее. Во втором случае трогать машинный код не нужно и работа происходит только с данными: графикой, шрифтами, текстом, пойнтерами (разделители текста) или даже музыкой. Но даже в этом случае расположение и формат этих данных изначально неизвестен.

Рассмотрим самую распространенную цель ромхакинга — любительский перевод. Изменение шрифтов необходимо, если алфавиты исходного и конечного языка отличаются, например если нужно заменить латиницу на кириллицу. То же самое касается и перевода с японского на английский или русский.

Кроме адреса, по которому начинается текст, нужно определить его кодировку, которая может быть абсолютно произвольной. Практически всегда разработчики старых консольных игр использовали поинтеры, благодаря которым также упрощается и деятельность ромхакера: изменяя их, он может не сохранять длины оригинальных участков текста. Также иногда при переводе редактируются надписи на спрайтах, хотя это делается далеко не всегда.

Абсолютно любые данные (а особенно текст) могут быть запакованы, и это может усложнить жизнь ромхакеру еще больше. В особенно запущенных случаях локализатору по-

WWW

Официальная страница No$gba: nocash.emubase.de/gba-dev.htm

INFO

Наиболее популярны у ромхакеров следующие платформы: Nes/Dendy/Famicom, SNES/Super Famicom, Sega Mega Drive / Genesis, Game Boy, Game Boy Advance, Nintendo DS, Nintendo 64, Game Cube, Wii, PSP, PlayStation 1, 2.

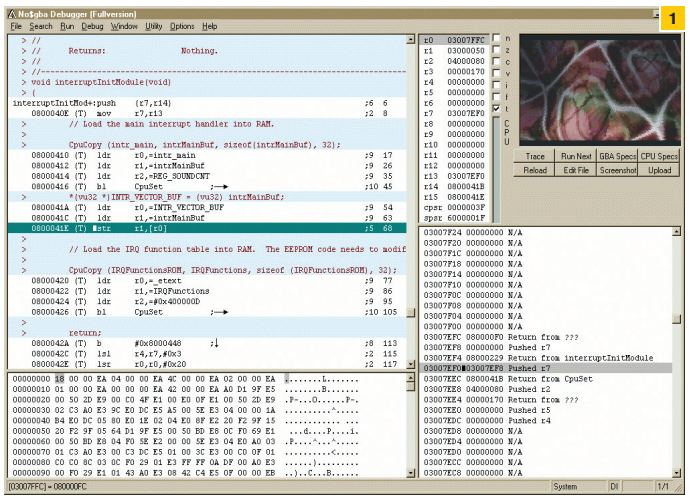


Рис. 1. No$gba — отладчик платформ Game Boy Advance и Nintendo DS

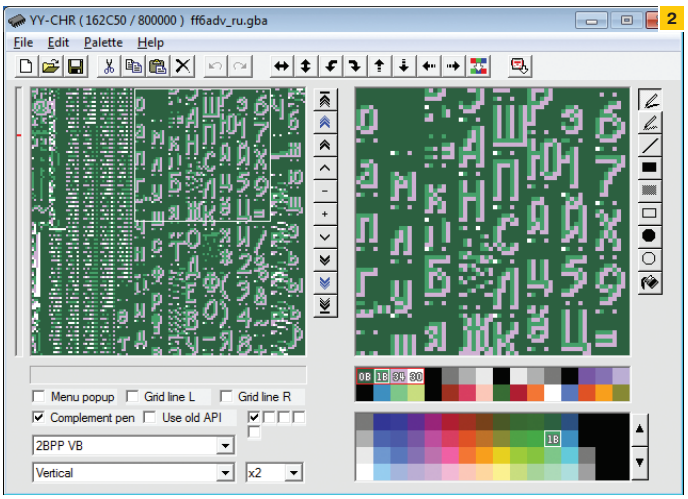
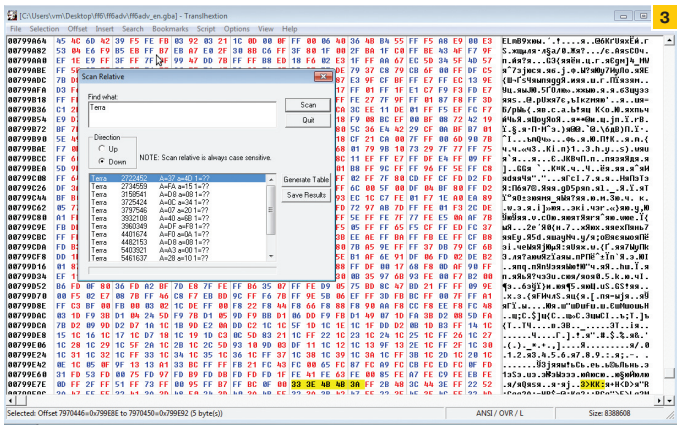
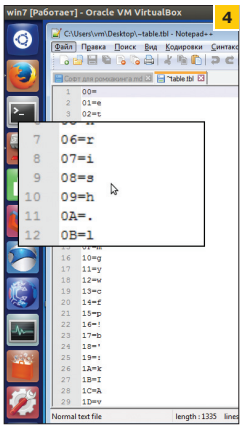


Рис. 2. Перерисованный шрифт в тайловом редакторе YY-CHR. В этом примере перед каждым символом идут данные, отвечающие за его ширину





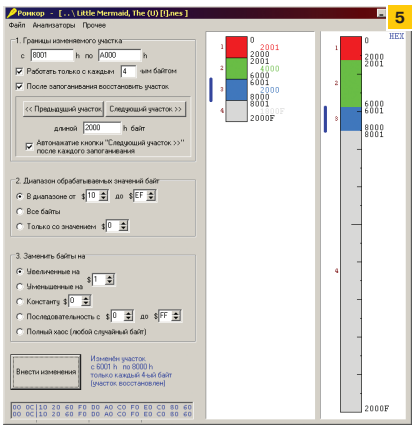


Рис. 3. **Translhextion — одна из ромхакерских утилит, которая позволяет искать текст в неизвестной кодировке, подбирая ее автоматически путем перебора вариантов. Напомню, что этот прием работает, только если алфавит кодирован по порядку, а текст не запакован**

Рис. 4. **Пример таблицы символов, в которой они идут не по порядку (реальный пример из Final Fantasy VI Advance для GBA). В данном случае автоматически выяснить местоположение и кодировку текста Translhextion не в состоянии, но их можно определить вручную при помощи корруптора**

Рис. 5. **Корруптор Ромкор: bhlady.narod.ru**

Рис. 6. **Калькулятор поинтеров Kruptar позволяет практически забыть о технических деталях во время редактирования текста. Также поддерживает извлечение/вставку запакованного методом MTE текста. Можно обеспечить поддержку другого метода сжатия благодаря возможности написания к нему плагинов**

Рис. 7. **Поинтеры можно увидеть в hex-редакторе невооруженным глазом**

надобится разобраться в языке ассемблера под конкретную платформу: если «на ощупь» никак не получается выяснить формат паковки данных, можно прибегнуть к дебаггингу для выяснения алгоритма распаковки. Но поскольку у каждой платформы ассемблер свой, я рассмотрю общие для всех платформ приемы, доступные даже тем, у кого нет желания или возможности редактировать машинный код соответствующей игровой приставки.

НАХОДИМ ЦЕЛЬ

Перед тем как выяснить, в каком формате хранятся нужные данные, надо сначала определить, по каким адресам они расположены. Если ты думаешь, что для того, чтобы найти расположение текста, нужно скроллить hex-редактор, то очень сильно ошибаешься. Дело в том, что кодировка текста не обязана быть стандартной. Например, латинской A может соответствовать абсолютно любое значение байта от 00 до FF. Поэтому, чтобы увидеть текст в hex-редакторе, сначала нужно скормить ему составленную таблицу кодировки символов. Только вот такую таблицу вряд ли получится сделать, пока точное расположение текста не будет найдено. А вот найти его путем скроллинга в различных режимах тайлового редактора, в принципе, можно, все равно это долго и не гарантирует, что найдешь: формат шрифта опять же может быть произвольным, иногда тайловый редактор может и не поддерживать этот формат. Не говоря уже о том, что любые данные могут быть запакованы.

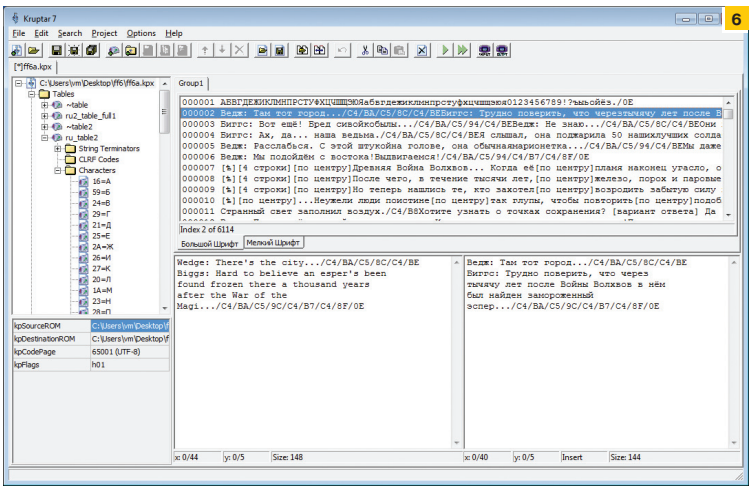
Тем не менее решение задачи по гарантированному нахождению всех нужных данных выглядит довольно просто. ПО для ромхакинга, которым нужно воспользоваться в первую очередь, называется корруптор. Его задача — временно испортить ром на определенном участке, запустить ром для проверки, а затем вернуть в исходное состояние. Опционально это может быть увеличение/уменьшение значений, случайные значения или заданные (последний вариант также полезен для определения кодировки текста, когда он будет найден).

Например, для начала можно разбить текст на восемь участков и пройтись по каждому. Если несмотря на то, что некоторые данные в роме испорчены, шрифт и текст отображаются нормально, значит, этих данных на проверенном участке нет и мы можем пометить его как полностью проверенный и больше к нему не возвращаться. Если после порчи данных игра зависает, то, скорее всего, был затронут программный машинный код, а значит, пока что неизвестно, есть ли на этом участке, помимо программного кода, текст или шрифт. Чтобы это выяснить, нужно данный большой участок разбить на более мелкие и пройтись уже по ним, разумеется рекурсивно уменьшая размер исследуемых участков. Наконец, если мы видим, что текст или шрифт на этом участке испорчен, то можно будет сразу же сосредоточиться именно на нем и, постепенно уменьшая исследуемые интервалы, точно установить начало и конец интересующих данных.

Существует еще один способ найти текст. Он более быстрый, но не гарантирует результат и сработает, только если текст не запакован и алфавит закодирован упорядоченно, то есть, например, если a = 2D, то b = 2E, c = 2F, d = 30 и так далее. Можно для начала попробовать взять какое-либо не очень короткое слово, встречающееся в тексте игры, только строчными (или только заглавными) буквами, а дальше пусть самописное или готовое ПО пробежится по рому в поисках этого слова (256 – 26 = 230) раз. Если ничего найдено не будет, то я рекомендую не париться и просто воспользоваться корруптором.

ВНОСИМ ИЗМЕНЕНИЯ

Для редактирования шрифта или графики можно воспользоваться готовым тайловым редактором, но нужно иметь в виду,



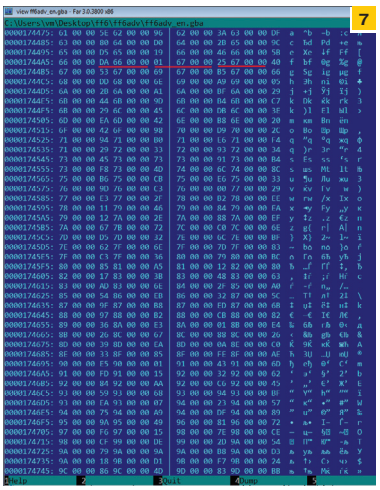


Рис. 8 Один из просмотрщиков, реализованный на Flash

что формат хранения шрифта/графики может оказаться экзотическим и/или сжатым. Тогда придется или самому писать редактор шрифта, или «перерисовывать» шрифт в hex-редакторе, произведя все расчеты вручную, или написать перекодировщик из BMP или PNG в необходимый формат.

При редактировании текста нужно учитывать, что по умолчанию любой замененный тобой кусок не должен быть длиннее оригинала. Как правило, это ограничение очень легко обойти за счет изменения поинтеров, которые, в отличие от текста, очень узнаваемы сразу же в hex-редакторе. Но если перевод не ограничивается главным меню и окном опций, то пересчитывать и редактировать их вручную, мягко говоря, не стоит: рассчитывать и изменять их нужно автоматически. Из уже готового ПО я могу порекомендовать для этих целей Kruptar. Используя подобный софт, можно комфортно делать перевод, совершенно не парясь о том, что если вносить корректировки в уже осуществленный перевод и длина отредактированного фрагмента текста изменится, то значения множества поинтеров сдвинутся. Kruptar или ПО с аналогичным функционалом полностью возьмет на себя постоянный пересчет и правку поинтеров.

Правда, даже если используется Kruptar, очень желательно отслеживать, чтобы переведенный текст всегда умещался на экране в отведенное ему место. Конечно, это можно делать, тестируя переведенный ром через эмулятор, но это долго. Иногда для этой цели пишется специальный просмотрщик сообщений.

Для Final Fantasy V Advance HoRRoR сделал схожий просмотрщик, с описанием и скриншотом которого можно ознакомиться здесь: j.mp/1jrR8hD, но он не выкладывает свой софт в паблик по непонятной для меня причине.

Можно проверять, не вылезает ли текст за отведенные ему границы, не вручную, а автоматически.

Например, если использовать Python, то достаточно всего лишь определить словарь, ключами которого являются символы, а их значения равны ширине этих символов в пикселях в соответствии с игровым шрифтом. Затем нужно, используя регулярное выражение, пробежаться по всему тексту, отдельно по символам каждой строки, посчитав таким образом ширину текущей строки и сравнивая с максимально возможным значением, записать в лог все случаи превышения лимита (если они есть). Если лог окажется не пустым, то нужно проверить все найденные им места с помощью графического просмотрщика.

ЗАКЛЮЧЕНИЕ

В статье кратко описан процесс базового ромхакинга с применением готового софта. Хотя, конечно, лучшие инструменты — это интерпретатор или компилятор любого подходящего языка по вкусу и голова. Если готовый софт отлично справляется с возникшими задачами, то оправданно идти по уже протоптанной другими людьми тропинке, вместо того чтобы ломиться через сугроб. Но и стесняться писать собственный софт не стоит. Например, отчаянно не хватает портов виндовых утилит — даже на romhacking.net в разделе UNIX всего лишь восемь софтин. Удачи!

WWW

Сайт первой и старейшей русскоязычной ромхакинг-группы «Шедевр», появившейся в 2001 году: shedevr.org.ru

Сайт группы Magic Team, на котором есть несколько обучающих статей и их софт, включая Kruptar: www.magicteam.net

Сайт группы Chief-Net, где также можно почитать обучающие статьи: chief-net.ru

Сайт группы Owls Group: owls-group.org.ru

Сайт группы ExclusivE: ex-ve.ru/pronas

Сайт, пожалуй, самого известного российского ромхакера HoRRoR. На сайте также присутствует Wiki-раздел, где можно не только почитать статьи на данную тему, но и поделиться своим опытом. Отдельно стоит отметить, что его командой выполнен идеальный перевод первой части культового хоррора Silent Hill, выходившего эксклюзивно для первого поколения PlayStation в 1999 году: consolgames.ru

Крупный англоязычный портал, посвященный ромхакингу: www.romhacking.net

Крупнейший русскоязычный портал, посвященный эмуляции с практически полным архивом эмуляторов и ромов: emu-land.net

Сайт хака Mortal Kombat. Работа проделана огромная: umk3.hacking-cult.org

МЕТОДЫ СЖАТИЯ ДАННЫХ

MTE

Самый простой для понимания метод — словарная система, как правило используемая для сжатия текста. Метод называется MTE, при его использовании одним или двумя байтами кодируется сразу несколько (а возможно, даже много) символов, комбинация которых часто встречается в тексте. Вообще говоря, MTE можно считать не методом сжатия, а всего лишь обычной кодировкой с поправкой на то, что одному/двум/нескольким байтам может соответствовать не только один символ, но и несколько/много.

Пресловутые комбинации символов прописаны в словаре, который также хранится где-то в роме. Формат словаря может быть разный: слова, разделенные спецсимволом; слова, записанные слитно, + указатели на них; слова, записанные слитно, + таблица длин. Под «словом» в данном случае понимается произвольный набор символов, среди которых могут встречаться и пробелы. То есть таким «словом» в отдельных случаях может являться и несколько слов, например какое-то часто встречающееся в тексте словосочетание. С другой стороны, это может быть и часть слова, например ing. Даже если слово совпадает с языковым, то оно может использоваться и как часть более длинных слов. Например, если артиклю the соответствует значение {D6}, то местоимение they, скорее всего, будет везде сокращаться до двух байт: {D6}y.

Если остались вопросы, то краткую статью о том, как ломать MTE, можно прочитать на сайте ромхакинг-группы Chief-Net: j.mp/1gks3jM.

Инструмент Kruptar, о котором мы неоднократно рассказывали в этой статье, поддерживает MTE из коробки просто потому, что поддерживает таблицы символов, в которых произвольному количеству байт может соответствовать произвольное количество символов. Благодаря поддержке таких таблиц есть также возможность обозначить специальные байт-коды, которые могут встречаться в тексте и которые не хочется запоминать, специальными кодами, понятными человеку и несложными для запоминания.

Также можно упомянуть DTE — это частный случай MTE, когда часто встречающаяся комбинация из двух символов (например, сочетание th) кодируется одним байтом.

RLE

Также довольно прост метод RLE (Run Length Encoding). Он не очень подходит для сжатия текста, но может пригодиться для сжатия графики. А прост он потому, что его фишка заключается всего лишь в замене длинной последовательности повторяющихся много раз одних и тех же элементов — байтов или последовательностей байт, например отвечающих за отображение пикселей одного и того же цвета. Многократное повторение элемента всего лишь заменяется на одну копию этого элемента и число, отвечающее за количество повторений этого элемента.

Несмотря на свою простоту, даже RLE способен помешать увидеть графику в тайловом редакторе. Если, конечно, это не специализированный редактор с поддержкой автораспаковки.

LZ77

Более эффективен широко известный метод сжатия семейства LZ, до сих пор имеющего кучу применений в различных областях современного IT. Назван в честь своих разработчиков Абрахама Лемпеля и Якоба Зива, а также года его публикации. Неплохо подходит для сжатия как графики, так и текста.

Идея заключается в использовании ссылок на ранее встречавшийся фрагмент информации, при этом данный метод реализован так, что, по сути, уже включает в себя и фишку RLE.

Обзор утилиты unLZ-GBA для борьбы с запакованной методом LZ77 графикой на GBA: j.mp/1m9KcGk.

Также можно найти более подробное описание RLE и LZ77: habrahabr.ru/post/141827/.