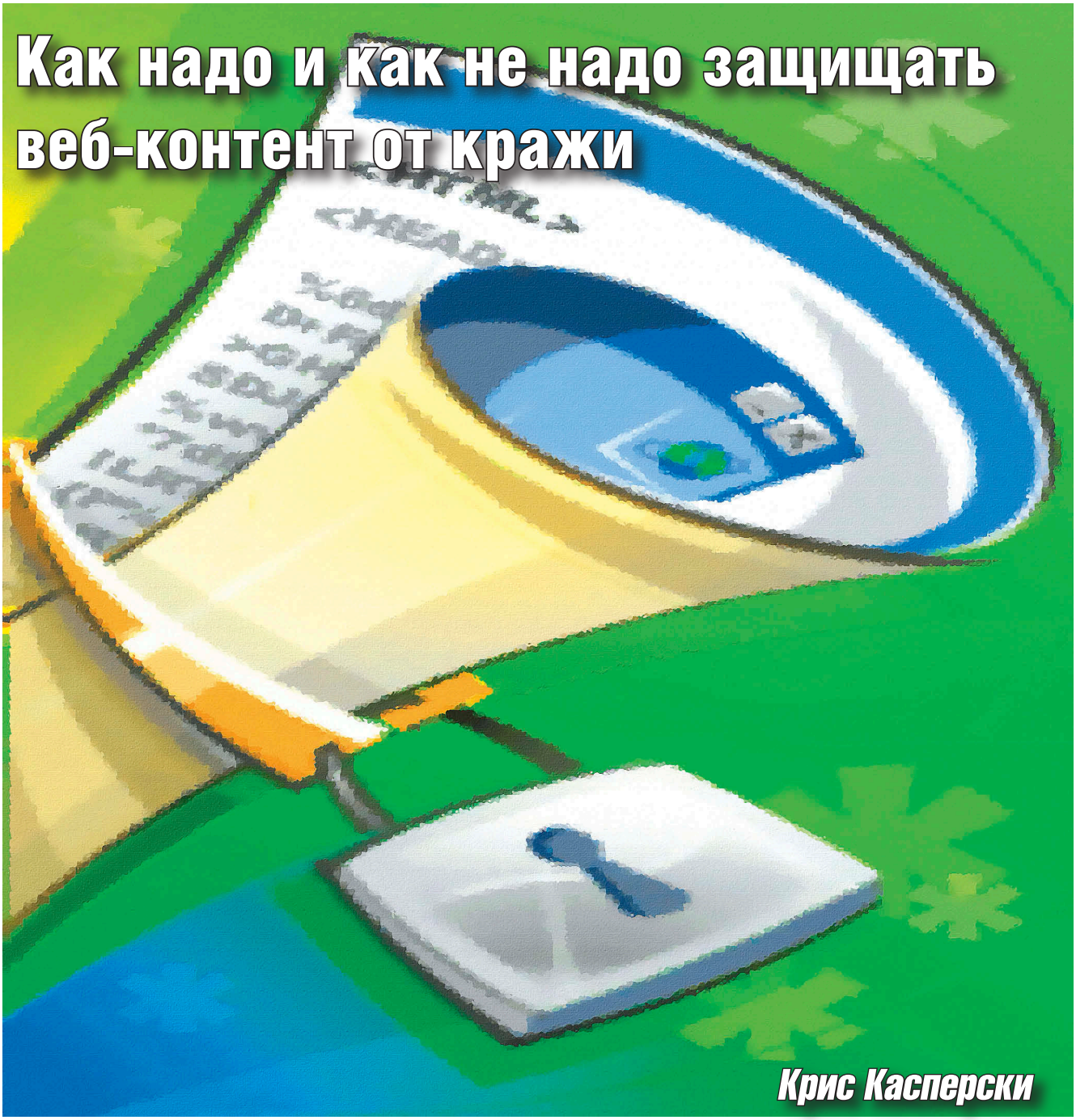
Как надо и как не надо защищать веб-контент от кражи



Крис Касперски

Для охраны веб-контента можно использовать одну из многих защитных систем, имеющихся на рынке (но большинство из них давно поломано), или попытаться смастерить что-то свое, переоткрывая колесо и наступая на грабли, сопутствующие с подводными рифами. Как надо и как не надо защищать веб-контент от гнусных посягательств?

Защищать или не защищать

Специфика защиты веб-контента (под которым мы будем понимать всю совокупность текстового и графического содержимого вместе с особенностями оформления) антагонистична по своей природе. С одной стороны, мы стремимся передать информацию посетителю, а с другой – очень сильно не хотим, чтобы он получил эту информацию в любом виде, пригодном для дальнейшего осмысления и обработки. Как говорится, что в Сеть попало, то пропало. В этом и есть сущность Сети. Любая полезная информация мгновенно распространяется по сотням тысяч узлов, и в мире нет силы, способной это предотвратить. Если вы хотите чем-то поделиться с миром – делитесь! Одна искорка не сделает мир теплее или светлее, но… сидеть в кромешной темноте и ничего не делать – еще глупее.

Но это все была лирика, которой сыт не будешь. А у кого-то уже жена, и дети растут. Все это хозяйство требу-

ет денег, и, чтобы их заработать, приходится охранять свой креатив от желающих обогатиться за чужой счет. Но всегда ли нужно защищаться? Например, сайтам, ненавязчиво рекламирующим бытовую и офисную технику (фотоаппараты, мобильные телефоны) и живущим преимущество за счет спонсоров и рекламы, защита идет только во вред. Если кто-то хочет скопировать рекламную статью, описание или документацию на новый смартфон – пусть копирует! Он ведь не зря копирует, а с расчетом где-то разместить! Пусть даже на своем сайте, в журнале, книжке. Чем больше людей узнают о товаре, тем круче взметнется кривая продаж!

Тем не менее существует большая группа людей, считающих, что любое несанкционированное копирование информации – это прямая упущенная выгода и без защиты сайт оставлять нельзя. Посетителям демонстрируются фотографии или электронные тексты, но за возможность сохранить их на диск/распечатать/вставить в свой реферат необходимо заплатить. А платить, естественно, никто не хочет (или хочет, но не может ввиду неразвитости систем оплаты). Вот и приходится защищать. При всей порочности этой методики и моем отвращении к ней однажды я оказался замешан в проектировании такой защиты. А для этого потребовалось проанализировать весь комплекс технических средств, имеющихся на рынке, намотать на ус чужие ошибки, отобрать лучшие решения, добавить свои собственные идеи и соединить все это воедино.

Что именно мы собираемся защищать

Современные сайты далеко ушли от своих прародителей, и, собственно, самого HTML-кода «в чистом виде» в них практически не осталось. Основную статью расходов ныне составляют разработка графического дизайна, проектирование PHP-движка (естественно, PHP приведен лишь в качестве примера, зная CGI, движок можно написать и на Си), ну и, наконец, «набивка» сайта живым содержимым – текстами статей, фотографиями и т. д.

Дизайн – единственное, что нельзя защитить, поскольку расположение графических элементов и организация доступа к ним – это идея в чистом виде, выпадающая из поля зрения авторского и патентного прав (**рис. 1**). Но вот графические элементы, слагающие дизайн, защитить можно (хоть и сложно) (**рис. 2**).

PHP-движок в защите не нуждается, поскольку правильно настроенный веб-сервер никогда не отдает программные модули, а всего лишь возвращает результат их работы, представляющий собой веб-страничку, сгенерированную на лету, благодаря чему потребность в защите самого HTML-кода практически полностью отпадает. Теоретически веб-мастер может запрограммировать хитрый JavaScript, внедряемый в страницу в «чистом виде», однако что мешает ему реализовать весь «стратегический» код на серверной стороне? Конечно, какая-то часть кода неизбежно должна присутствовать и на клиентской стороне для обеспечения надлежащего уровня интерактивности (ниспадающие меню, всплывающие подсказки…), но все это уже давно написано и претендовать на уникальность в этой «отрасли» по меньшей мере странно.

В защите нуждается не дизайн и не PHP/JS-код, а именно само содержимое сайта во всей его текстовой и графической совокупности, ведь посещаемость (в долговременной перспективе) определяется именно содержимым, неприкосновенности которого угрожает:

Рисунок 1. Дизайн сайта – единственное, что нельзя защитить

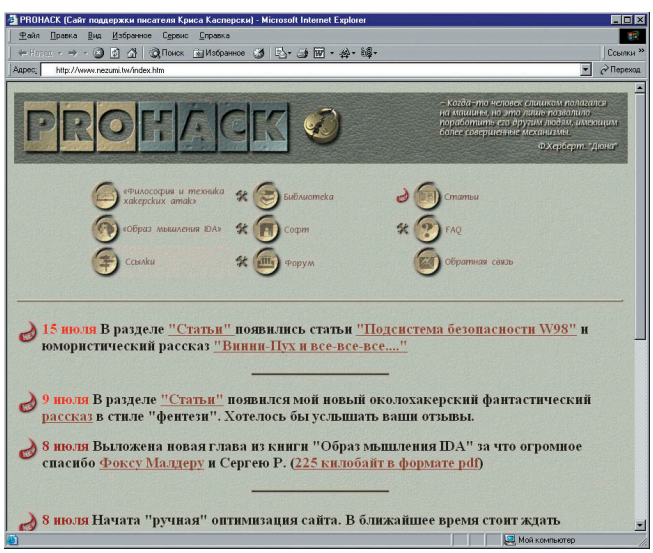


Рисунок 2. …а вот графические элементы дизайна защитить можно!

- сохранение HTML на диск;
- копирование фрагментов текста в буфер обмена;
- сохранение изображений правой кнопкой мыши;
- «телепортеры» – программы, сохраняющие весь сайт целиком (или его часть) для off-line просмотра или создания «пиратского» зеркала.

Однако прежде чем приступать к защите, неплохо бы рассмотреть ситуацию не только с технической, но и с экономической точки зрения. Никакая защита не достается бесплатно, даже если взять готовый некоммерческий протектор, то следует сразу же приготовить себя к тому, что зашифрованные страницы (если в защите использовалось шифрование) окажутся незамеченными поисковыми роботами, конфликты защиты с не IE-браузерами (а разработать неконфликтную защиту очень трудно). Короче, мы теряем пользователей, и если очень постараться, то можно добиться, что на сайт вообще никто не зайдет. Спрашива-

ется, а для чего тогда его вообще делать?! Кроме того, никакая защита не обходится без оверхида (т.е. накладных расходов), выражающихся в увеличении размеров страницы и, как следствие, увеличение нагрузки на сервер и дополнительный трафик. А вот это уже прямой убыток (особенно на сайтах с высокой посещаемостью).

Самое главное, что опытных пользователей никакая защита не остановит. Одно дело, если человек хочет перетащить картинку на свой рабочий стол или пополнить домашнюю библиотеку. Тогда действительно есть шанс, что в процессе ковыряния защиты его желание упадет ниже нуля и он просто махнет рукой и займется чем-то более прибыльным или интересным. А вот если копирование сайта преследует коммерческую выгоду, превышающую стоимость взлома, он будет скопирован. Кто-то привлекает к этой работе хакеров, кто-то голодных (и бесправных!) студентов, сравнительно быстро «перебивающих» сайт вручную.

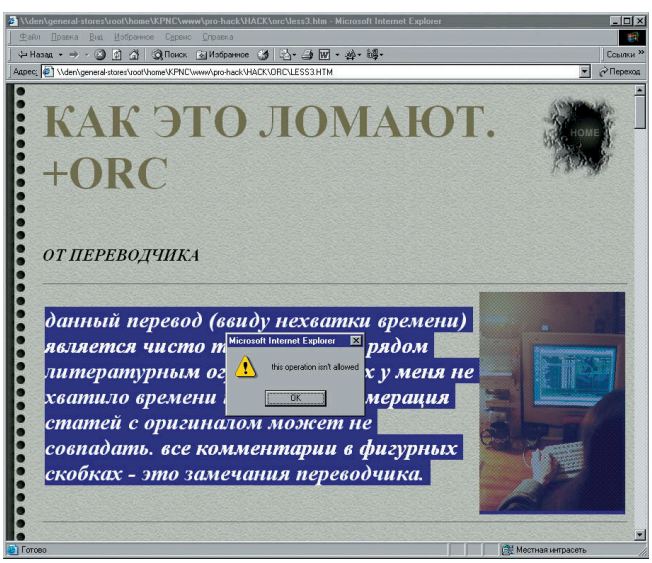


Рисунок 3. Защита перехватывает нажатие правой кнопки мыши и вместо контекстного меню выводит диалоговое окно с лаконичным сообщением «this operation isn't allowed»

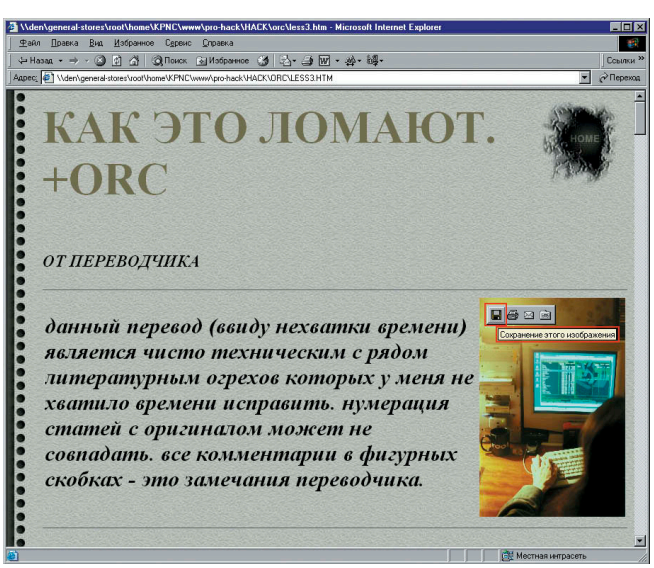


Рисунок 4. Обход защиты «правой кнопки» путем наведения мышиного курсора в угол сохраняемой картинки в последних версиях IE

То есть мысль следующая: по сравнению с защитой программного обеспечения защита веб-контента более бесполезна и менее надежна, потому что есть общий алгоритм ее взлома, связанный с разумной стоимостью, которая в разы ниже, чем для программного обеспечения, так как не требует квалифицированной рабочей силы.

Как не нужно защищать текст и картинки

Количество веб-мастеров, пользующихся «грязными» защитными приемами, просто поражает. Эти приемы доставляют множество неудобств легальным пользователям и ломаются за несколько секунд даже без напряжения мозговых извилин.

Самым распространенным методом защиты копирования картинок и грабежа текста был и остается JavaScript, перехватывающий click и возвращающий false если event.button == 2.

В простейшем случае исходный текст этой «защиты» выглядит так:

Листинг 1. Листинг Java-скрипта, блокирующий правую кнопку мыши

```
<HEAD>
<SCRIPT language=JavaScript>

      function click(x)
      {
         if (document.all)
         {
            if (event.button == 2)
            {
                alert("this operation isn't allowed");
                return false;
            }
         }

      }
      document.onmousedown=click;

</SCRIPT>
</HEAD>

<BODY>
```

А вот результат его работы: выделяем текст мышью, щелкаем правой кнопкой мыши и вместо привычного контекстного меню видим грозное диалоговое окно (см. **рис. 3**).

Если же после выделения текста не трогать мышь, а нажать <CTRL-Ins>, <CTRL-C> или обратиться к пункту «Копировать» в меню «Правка», то, несмотря ни на какие защиты, текст и даже изображение будут успешно скопированы в буфер обмена, откуда их (естественно, по раздельности) можно вставить в текстовый и графический редакторы соответственно.

Кстати, если в последних версиях IE подвести к картинке мышь и некоторое время ее не двигать, возникнет панель инструментов с изображением «дискетки», сохраняющей изображение несмотря ни на какие скрипты (см. **рис. 4**).

Хорошо, давайте усилим защиту и напишем скрипт, запрещающий не только контекстное меню, но еще и выделение текста, причем не только мышью, но и по комбина-

Выделить все»:

Листинг 2. Исходный код Java-скрипта, блокирующего не только правую кнопку мыши, но еще и выделение текста всеми доступными способами

```
<SCRIPT LANGUAGE="JavaScript">
      document.ondragstart = ops;
      document.onselectstart = ops;
      document.oncontextmenu = ops;

      function ops()
      {
               return false;
      }

</SCRIPT>
```

Проверка показывает, что мышь действительно «отдыхает», пункты «Выделить», «Копировать» и «Вставить» заблокированы, а «Выделить все» хоть и не заблокировано, но не работает, как и контекстное меню, вызываемое по <SHIFT-F10> или клавишей, расположенной слева от правой кнопки <CTRL> (см. **рис. 5**).

Однако торжествовать победу еще рано. Во-первых, панель инструментов, возникающая при наведении мышью на картинку, по-прежнему исправно работает, а во-вторых, пользователь может отключить JavaScript, возвращая своему любимому браузеру всю его функциональность (см. **рис .6**).

Раз наша защита построена на скриптах, необходимо проектировать страницу так, чтобы без скриптов она отображалась не полностью или вообще не отображалась совсем. Проще всего использовать конструкцию «<script>document.write("text");</script>», конкретное воплощение которой может выглядеть, например, так:

Листинг 3. Вывод содержимого сайта через скрипты и, как следствие, препятствующий их отключению

```
<script>document.write ("Данный перевод (ввиду нехватки
времени) является чисто техническим с рядом литературных
огрехов, которые у меня не хватило времени исправить. Нумерация
статей с оригиналом может не совпадать. Все комментарии
в фигурных скобках - это замечания переводчика");</script>
```

Ниже показан внешний вид защищенной странички с отключенными JavaScript (естественно, вывод предупреждения о необходимости включения скриптов лишним не будет и совсем не помешает, а то ведь некоторые могут и не догадаться, что тут что-то предполагается увидеть) (см. **рис. 7**).

Ладно, с текстом мы более или менее разобрались. А как быть с картинками? Некоторые разрезают одну картинку на множество мелких частей, наивно полагая, что пользователю будет лень сохранить пару десятков фрагментов, а затем подгонять их в текстовом редакторе. Но… тут выясняется, что:

- особенности формата jpg приводят к тому, что края разрезанной картинки уже не стыкуются и приходится либо уменьшать степень сжатия, либо переходить на png, но и то, и другое ведет к росту трафика и замедлению загрузки;
- браузеры очень плохо справляются со склейкой картинок и очень часто возникают «артефакты» в виде пустых линий или наложений картинок друг на друга (особенно если пользователь смотрит страничку с нестандартными настройками браузера типа разрешения, масштаба и т. д.);

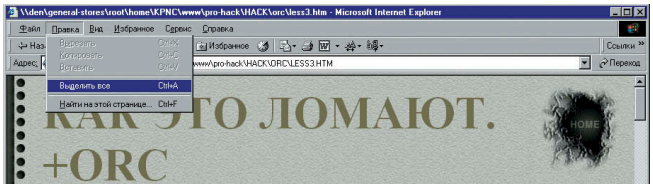


Рисунок 5. Заблокированные пункты меню «Вырезать», «Копировать» и «Вставить» в IE, «Выделить все» хоть и не заблокировано, но все равно не работает

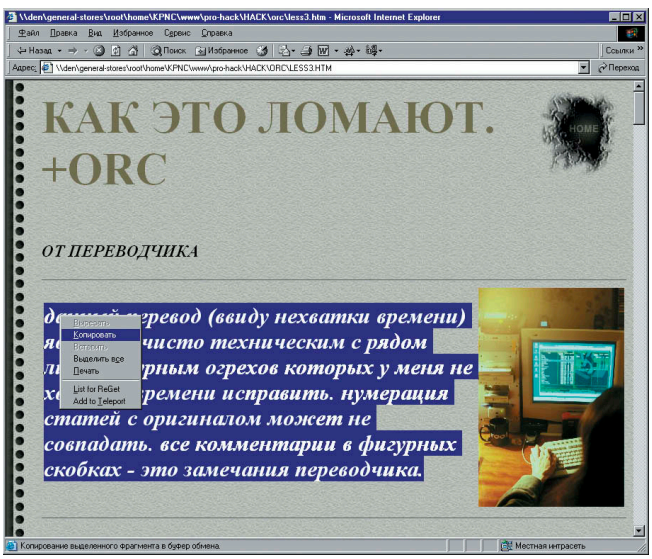


Рисунок 6. Обход защиты путем отключения скриптов — правая кнопка сразу же восстанавливает свою работоспособность

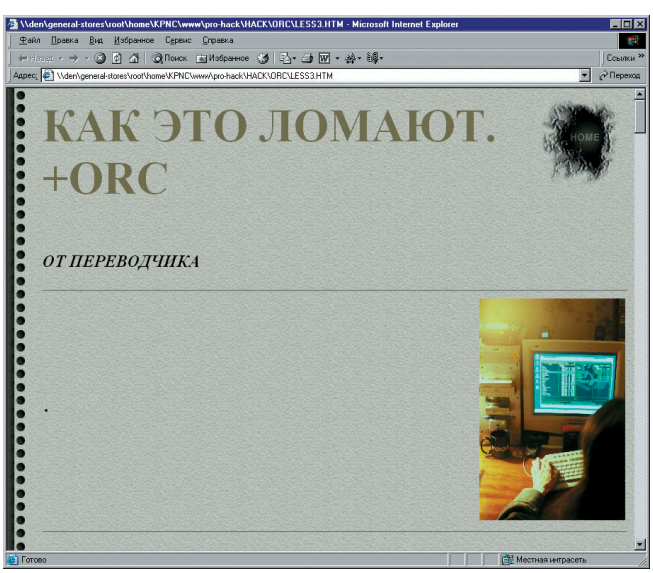


Рисунок 7. Внешний вид защищенной странички с отключенными скриптами (исчез ранее присутствующий текст, см. рис. 6)

- клавиша <print screen> делает грабеж картинки простым и приятным.

Так же не стоит использовать картинки в виде флеш-изображений. Их уже научились сохранять даже начинающие пользователи, а многие менеджеры закачек делают это автоматически.

Сложнее бороться с сохранением страницы на диск. Некоторые браузеры содержат ошибки, приводящие к невозможности сохранения при нарушении структуры HTML,

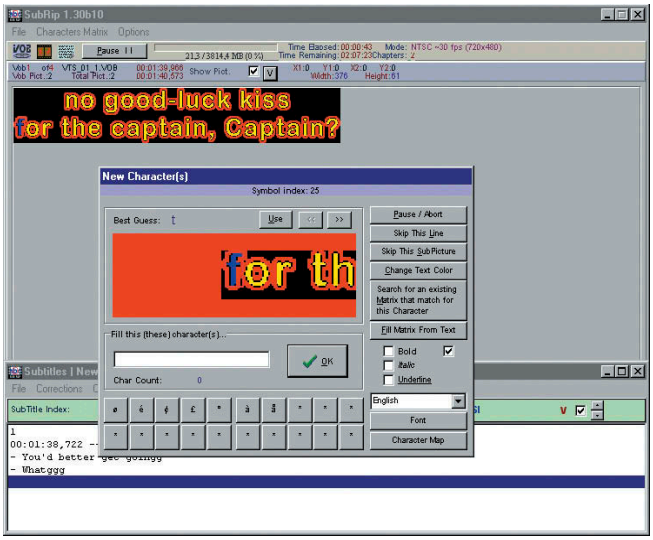


Рисунок 8. Внешний вид программы SubRip, обладающий способностью качественно распознавать текст с предельно низким разрешением (вообще-то он используется для «грабежа» субтитров с DVD, но при желании его можно приспособить и для работы и с другими видами графических текстов)

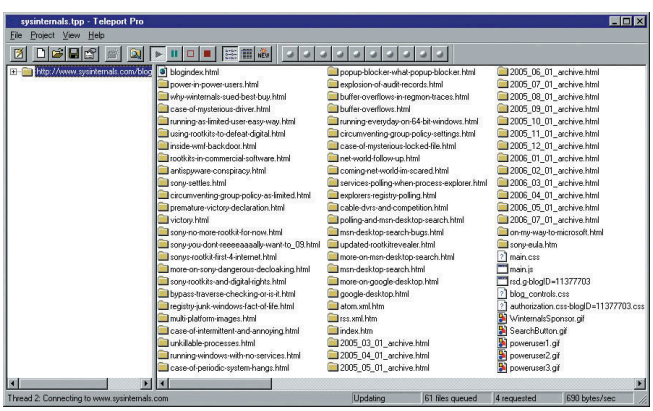


Рисунок 9. Сохранение целого сайта на жесткий диск с помощью программы Teleport Pro

при этом сама страница отображается вполне нормально, но… никаких гарантий, что остальные браузеры поведут себя точно так же, у нас нет. Зато вполне оправдывает себя прием «разрезания» текстового содержимого на кусочки по 1-2 Кб. Читать страницу (в интерактивном режиме) вполне возможно, и даже нельзя сказать, чтобы сильно затруднительно (хотя постоянно нажимать на ссылку «Далее», конечно, быстро надоедает), а вот собрать из всех сохраненных страничек исходную статью – это же сколько труда нужно затратить!

Кстати, если по каким-то причинам страница не сохраняется на диск, можно открыть ее в MS Word и сохранить на диск либо как html, либо как doc. Вместе с картинками, таблицами и другими элементами. Если же это не получается и Word с грохотом падает, имеет смысл попытаться выделить наиболее значимую часть страницы и вставить ее в Word через буфер обмена. В 99% случаев это срабатывает!

Сложная защита с простым взломом

Бороться с грабежом веб-контента традиционными средствами совершенно бесперспективно. А что если… отображать текстовое содержимое в графическом виде? Идея совсем не так уж и безумна, как кажется. Ведь преобразование текста в графику необязательно осуществлять на серверной стороне! Это можно поручить и JavaScript (или ActiveX-модулю, если это IE).

На сервере мы имеем PHP-модуль, шифрующий текст и передающий его клиенту. Браузер клиента «скармливает» его JavaScript/ActiveX-модулю и тот в окончательном виде выводит его на экран. Для чтения разрешения вполне достаточно, а вот для OCR уже маловато, да и кому придет в голову пропускать веб-страницы через OCR одну за другой?

Конечно, назвать это защитой можно только с большой натяжкой, поскольку вместе с зашифрованным текстом клиенту передается и сам ключ, следовательно, разобравшись в алгоритме шифрования (изучив код сценария JavaScript), хакер без труда напишет программу, расшифровывающую текст, но не превращающую его в графику.

Естественно, на медленных компьютерах такой скрипт будет очень сильно тормозить и по сложности своей реализации он вплотную приближается к самому браузеру или… к Adobe с ее PDF, в последние версии которого добавлена возможность эффективной работы по Сети со всем букетом атрибутов запретов – невозможность выделения текста/печати, запрет сохранения на диск и т. д. Зачем же изобретать велосипед, когда можно купить готовый? Увы, готовый страдает целым рядом врожденных пороков. Как и любая серийная защита, Adobe поломан еще много лет назад и никакие ухищрения не позволят ему противостоять армии хакеров уже хотя бы в силу того факта, что хакеров много, а он один. Напротив, ковырять JavaScript, созданный на «коленках» веб-дизайнером Васей, скорее всего никто не будет.

К тому же реализовывать полноценный rendering-engine в скрипте совершенно необязательно. Достаточно ограничиться одним лишь текстом. Для усложнения взлома можно преобразовывать в графику некоторое количество символов еще на сервере (например, 6 или 9%, чтобы несильно замедлять время загрузки страницы), тогда без OCR уже не обойтись.

Популярный FineReader с таким разрешением уже не справляется, но с ним великолепно работает слегка переделанный SubRip (кстати говоря, распространяемый совершенно бесплатно). Идея заключается в следующем: графический файл разбивается на символы (алгоритм выделения символов, даже наложенных на сложные и малоконтрастные текстуры, давно отработан), затем изображение показывается пользователю, который должен ввести соответствующий ему символ (или даже несколько символов, если во входном тексте несколько букв слипаются в сплошное «месиво»). Для каждого символа создается битовая матрица, и в дальнейшем это изображение распознается автоматически. То есть программа как бы самообучается в процессе работы и обращается к пользователю все реже и реже. Поскольку возможных начертаний одного и того же символа существует не так уж и много (естественно, в контексте распознаваемой страницы, а не «начертаний вообще»), а самих символов и того меньше, обучение занимает совсем немного времени, и на выходе получается довольно качественно распознанный текст, пусть

Готовые защитные продукты

- **HTML Protector** – обладает способностью выборочно шифровать HTML-содержимое (Java/VBScript, e-mail-адреса против mail-грабберов, линки, текст, изображения), борется с Teleport и подобными ему программами, позволяет запрещать off-line-просмотр, выделение текста, сохранение изображений и печать; при необходимости разрезает изображения на кусочки и/или накладывает на них водяные знаки, конвертирует их в swf, а также препятствует отображению страницы во фрейме чужого сайта или копированию его фрагментов, поддерживает IE, FireFox и Opera: http://antssoft.fileburst.com/htmlprotector.zip.
- **HTML Power** – шифрует всю HTML-страницу целиком или только ее часть, борется с Teleport и подобными ему программами, препятствует сохранению страницы на диск, картинки не режет, водяные знаки не накладывает, но зато умеет блокировать правую кнопку мыши, выделение текста, сохранение изображений и вывод на печать: http://www.pullsoft.com/htmlpower.zip.
- **Encrypt HTML Pro** – шифрует всю HTML-страницу целиком или только ее часть, с Teleport не борется, изображения не режет, сохранению страницы не препятствует, но блокирует правую кнопку мыши, контекстное меню, выделение текста и печать: http://www.mtopsoft.com/download/enchp.zip.

Внимание:

- Не стоит путать шифрование, используемое этими программами, с шифрованием, описанным в статье: зашифрованное содержимое расшифровывается и выводится в браузер в текстовом виде и потому может быть сохранено на диск универсальной программой, представляющей собой обыкновенное расширение для браузера и считывающей содержимое окна через API-функции в обход пользовательского интерфейса. Такую программу (для каждого браузера) достаточно написать всего один раз, и она будет работать как часы. Напротив, описанный механизм шифрования с переводом текста в графику не допускает универсального взлома, требуя к себе индивидуального подхода!
- Зашифрованное содержимое (неважно каким путем) недоступно поисковым роботам, что крайне негативно отражается на посещаемости, поэтому шифрование имеет смысл использовать только для создания «закрытых клубов» и «тайных сообществ», в которые попасть можно только по «приглашению», а простой человек с улицы их навряд ли найдет.
- Шифрование также позволяет обходить разнообразные adult-фильтры, блокирующие доступ к контенту определенного содержимого (к которому относятся не только порноресурсы, но и некоторые политические сайты), впрочем, для этих целей существуют и другие, гораздо более продвинутые методики (например, VPN-тунели или https-proxy сервера).

не без ошибок, но все же пригодный для последующей работы с ним (см. **рис .8**).

Таким образом, взломать предложенную защиту все-таки возможно, но… для этого нужно быть хакером, а «потуги» обычных пользователей она выдержит вполне успешно.

Защита от телепортеров

Большинство веб-мастеров очень не любят, когда пользователи выкачивают весь сайт целиком, а потом спокойно почитывает его в off-line или прожигают на болванки, сбываемые в ближайших ларьках.

Существует множество программ, предназначенных для автоматического скачивания и обладающих весьма продвинутой системой фильтров, способных качать только то, что нужно, и не качать одни и те же страницы несколько раз подряд. Тем не менее их достаточно просто обмануть (см. **рис .9**).

Первое, что приходит в голову, – это запрашивать подтверждение при скачивании каждого файла в виде графической картинки с искаженными символами, которые требуется ввести (см. **рис. 10**). Конечно, это утомляет честных посетителей, но зато делает «телепортирование» сайта практически невозможным. Можно выводить картинку на экран, перекладывая задачу распознавания на плечи человека, и если таких людей будет много, то копирование займет вполне разумное время. При желании можно даже привлечь аудиторию с какого-нибудь порнографического сайта с гиперпосещаемостью, если, конечно, «телепортирование» стоит того.

Другой механизм предполагает генерацию случайных аплинков, что приводит к «зацикливанию» телепортера. Если веб-содержимое генерируется PHP (а в большинстве случаев это так), совсем несложно сделать, чтобы ссылка на предыдущую страницу (или «home page») каждый раз генерировалась случайным образом и «телепортер», обнаружив, что такой ссылки еще нет в его базе, повторял загрузку «home» несчетное количество раз. Чуть-чуть усовершенствовав алгоритм, можно добавить и случайную генерацию даунлинков.

Самый простой пример: каждый линк представляет ссылку на скрипт с параметром, идентифицирующим эту ссылку. Идентификатор состоит из двух частей, первая часть – случайным образом генерируемый ключ, которой расшифровывает (по XOR) вторую часть. Этот механизм, несмотря на всю его простоту, не позволяет «телепортеру» отличать ссылки на уже скачанные страницы от тех, что еще предстоит скачать. Теоретически (!) «телепортер» может анализировать содержимое скачанных страниц, выделяя среди них дубли, но что это ему дает?! Качать-то все равно придется… в бесконечном цикле!

Пара слов о поиске украденного

Как ни защищай свое имущество, его все равно уведут, и веб-содержание начнет расползаться по Сети. Это невозможно предотвратить, но найти грабителей – в общем-то пустяковое дело. Например, наложить на каждую фотографию свой логотип. Прием распространенный, но очень глупый и проблемы не решающий. Логотип, расположенный в незначащей части изображения (где-нибудь в углу) элементарно вырезается. А если расположить его посередине да еще выбрать размер покрупнее, на такую фотографию никому смотреть не захочется!

От «визуальных» логотипов лучше сразу же отказаться, а использовать водяные знаки, цифровые подписи, скрываемые в изображении методами стеганографии. Тогда, обнаружив «позаимствованный» материал на чужом сайте, по крайней мере, можно будет доказать факт кражи. Весь вопрос в том: как найти того, кто украл? Если незадачливый похититель недодумается переименовать фотографию, с высокой степенью вероятности она будет найдена Google или другим поисковиком. Аналогичным образом осуществляется и поиск похищенных текстов, причем тексты ищутся еще быстрее и легче. Можно попробовать выделить уникальную фразу, взятую в кавычки (многие именно так и поступают), однако в этом случае Google зачастую выдает негативный результат (что неоднократно проверено на практике), поэтому лучше выделить из своего текста несколько уникальных слов (комбинация которых в других текстах практически не встречается) и попробовать найти их, не используя кавычек. Замечено, что Google плохо находит те слова, которые встречаются только на одном конкретном сайте (например, Капитан Кракрыпырзззкллл), и кириллические слова, содержащие в себе цифры или спецсимволы. В общем, поиск – дело тонкое, но тот, кто ищет, тот всегда найдет!

Кстати, некоторые стеганографические алгоритмы (и утилиты, их реализующие) переживают различные трансформации изображения и даже полиграфическую печать (естественно, не отечественного качества). С другой стороны, практически все современные печатающие устройства оснащены специальными защитами от несанкционированного тиражирования банкнот и других ценных бумаг. Хакеры давно разгадали алгоритм, по которому принтер отличает банкноту от всего остального. Не вдаваясь в технические подробности, достаточно отметить, что там используется многоуровневая защита, в том числе основанная и на практически неразличимых (глазу) малоконтрастных окружностей и желтых точках на светлом фоне. В Сети имеется множество утилит, позволяющих обходить защиту, просто вырезая эти метки из изображения, но… их можно применять и в мирных целях – вносить в свою фотографию эти метки, после чего никакой принтер не станет их печатать! К сожалению, в силу своей изначальной криминальной ориентации данные утилиты постоянно меняют адреса, и дать постоянный URL невозможно. Остается только засесть за Google и искать, искать, искать…

Философское заключение

Потребность в охране веб-контента назрела уже давно, но адекватных методик защиты до сих пор предложено не было. И дело тут совсем не в открытости формата HTML. Защищать пытаются и eBook, и аудио/видеоконтент, но все с тем же неизменным (не)успехом! Если цифровое содержимое можно просмотреть (прослушать), то, очевидно, его можно и скопировать! Единственная надежда – на аппаратную защиту, вживленную непосредственно в «железо», но аппаратная защита наших дней сводится все к той же программе, упрятанной в микропрограммную прошивку (которую можно перешить), да и уровень технического развития уже позволяет создавать копирующие устройства самостоятельно.

Прежде чем вкладывать деньги, время, средства в защиту веб-контента, следует хорошо подумать: какие убытки это принесет и не превысят ли они «упущенную выгоду» от плагиаторов и пиратов. Практически все виды защиты, упомянутые в статье, базируются на JavaScript, а это значит, что мы теряем пользователей, отключающих JS по соображениям безопасности, а так же всех тех, кто пользуется браузерами, в которых JS не было еще с рождения (к таким браузерам, в частности, относится мой любимый Lynx). Про совместимость различных реализаций лучше вообще не говорить, и тут приходится тестировать не только весь зоопарк имеющихся браузеров, но и различные версии каждого браузера! А это уже серьезно. Это требует создания тестовой лаборатории и существенных расходов. Так что вопрос: защищать или не защищать – остается открытым.