защита web-контента от кражи

крис касперски ака мыщъх, no-email

готовых систем для защиты web-контента достаточно много, но все они слишком сырые, ненадежные, плохо совместимые с браузерами отличными от IE и, как правило, небесплатные. вот почему большинство web-мастеров пытаются смастерить что-то минимально работающее "на коленках", многократно переоткрывая колесо и наступая на одни и те же грабли. в этой статье мы покажем как надо и как не надо защищать web-контент от гнусных посягательств.

введение или защищать или не защищать

Специфика защиты web-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. В памяти непроизвольно всплывает анекдот. Мюллер: правда, что радистка Кэт дала показания? Штирлиц (сухо): показала, но не дала!

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, который сыт не будешь. А у кого-то уже дети растут. Жена… Все это хозяйство требует денег и, чтобы их заработать, приходится охранять свой креатив от желающих обогатиться за чужой счет. Но всегда ли нужно защищаться?! Например, сайтам, ненавязчиво рекламирующих бытовую и офисную технику (фотоаппараты, мобильные телефоны) и живущих преимущество за счет спонсоров и рекламы, защита идет только во вред. Если кто-то хочет скопировать рекламную статью, описание или документацию на новый смартафон — пускай копирует! Он ведь не зря это копирует! А с расчетом где-то разместить! Пусть даже на своем сайте, журнале, книжке. Чем больше людей узнают о товаре, тем круче взметнется кривая продаж!

Тем не менее, существует большая группа людей, практикующих метод "показала, но не дала". Посетителям демонстрируются фотографии или электронные тексты, но за возможность сохранить их на диск/распечатать/вставить в своей 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 мыщъх оказался замешен в проектировании вот такой мерзкой и противной 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 составляет 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.

Рисунок 1 дизайн сайта — единственное, что нельзя защитить

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. Но вот графические элементы, слагающие дизайн, защитить можно (хоть и сложно).

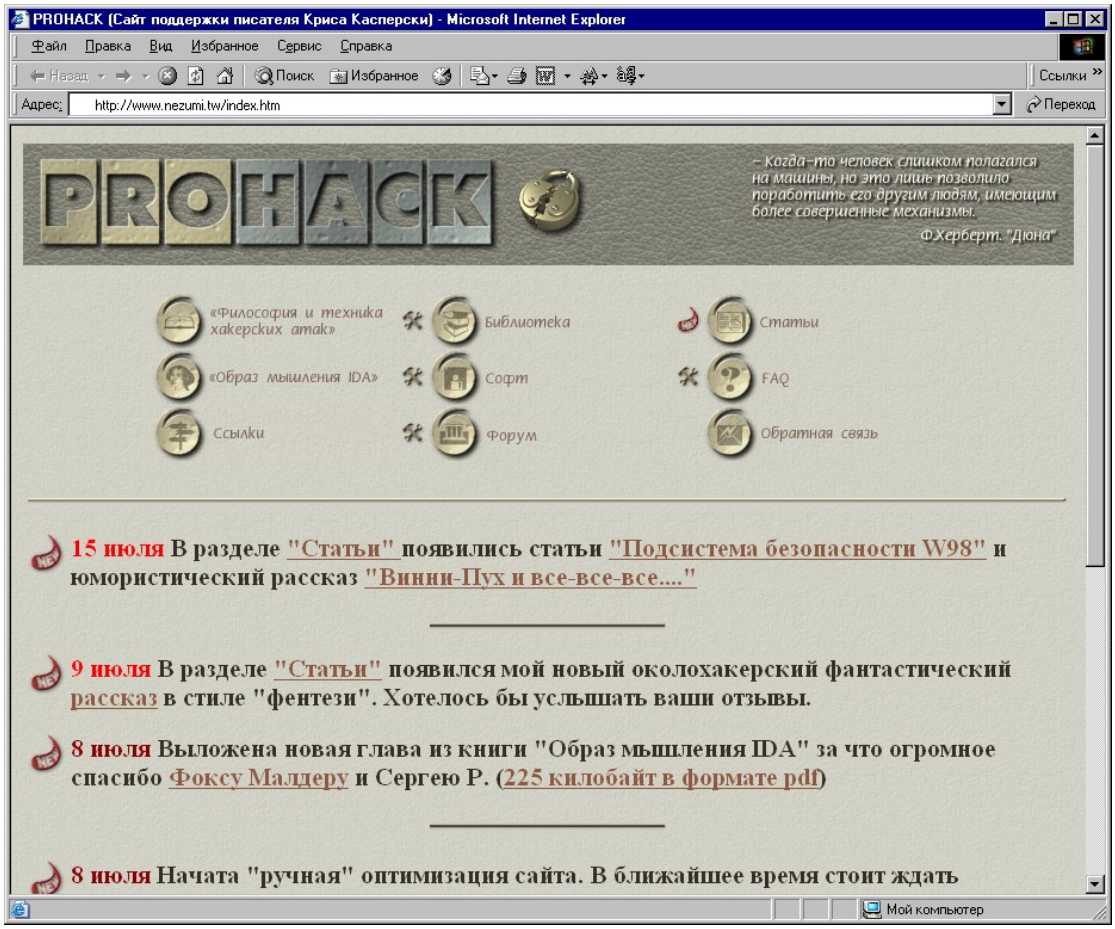

Рисунок 2 …а вот графические элементы дизайна защитить можно!

PHP-движок в защите не нуждается, поскольку правильно настроенный WEB-сервер никогда не отдает программные модули, а всего лишь возвращает результат их работы, представляющий собой WEB-страничку, сгенерированную "на лету", благодаря чему потребность в защите самого HTML-кода практически полностью отпадает. Чисто теоретически, WEB-мастер может запрограммировать хитрый Java-скрипт, 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 (всякие там ниспадающие меню, всплывающие подсказки…), но все это уже давно написано и претендовать на уникальность в этой "отрасли" по меньшей мере странно, а по большей — отдает явным пионерством.

В реальной защите нуждается не дизайн и не PHP/Java-код, а именно само содержимое сайта во всей его текстовой и графической совокупности, ведь посещаемость (в долговременной перспективе) определяется именно содержимым, неприкосновенности которого угрожает: а) сохранение HTML'а на диск, в) копирование фрагментов текста в буфер обмена; г) сохранение изображений правой кнопкой мыши; д) "телепортеры" — программы, сохраняющие весь сайт целиком (или его часть) для off-line просмотра или создания "пиратского" зеркала.

как _не_ нужно защищать текст и картинки

Количество web-мастеров, пользующиеся "грязными" защитными приемам 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 Java-скрипт, перехватывающий click и возвращающий false если event.button == 2.

В простейшем случае исходный текст этой "защиты" выглядит так:

```
<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>
```

Листинг 1 листинг Java-скрипта, блокирующий правую кнопку мыши

А вот результат его работы: выделяем текст мышью, щелкаем правой кнопкой мыши и вместо привычного контекстного меню видим грозное диалоговое окно, ругающее нас матом:

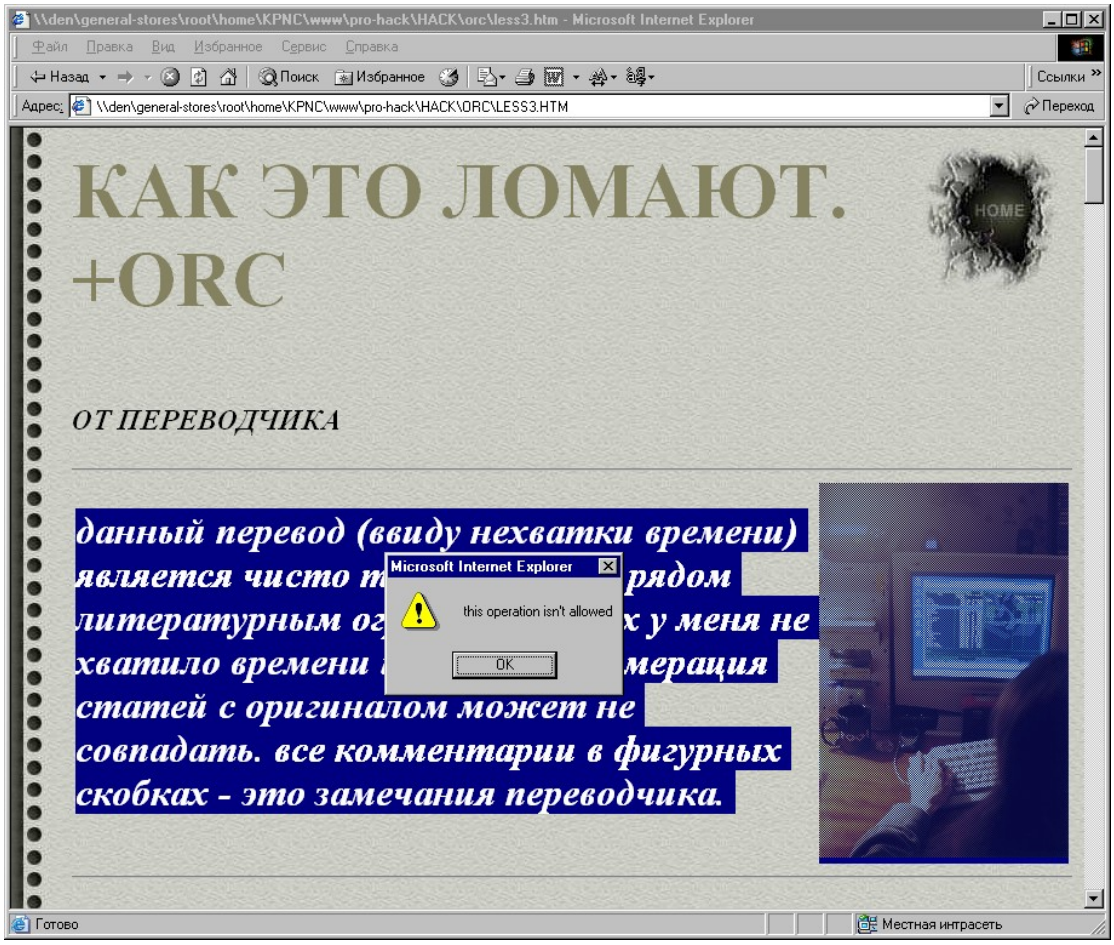

Рисунок 3 защита перехватывает нажатие правой кнопки мыши и вместо контекстного меню выводит диалоговое окно с лаконичным сообщением "this operation isn't allowed"

Если же после выделения текста не трогать мышь, а нажать <CTRL-Ins>, <CTRL-C> или обратится к пункту "копировать" 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 текстовой и графический редактор соответственно.

Кстати, если в последних версиях IE подвести к картинке мышь и некоторое время ее не двигать, возникнет панель инструментов с изображением "дискетки", сохраняющий изображение несмотря ни на какие скрипты:

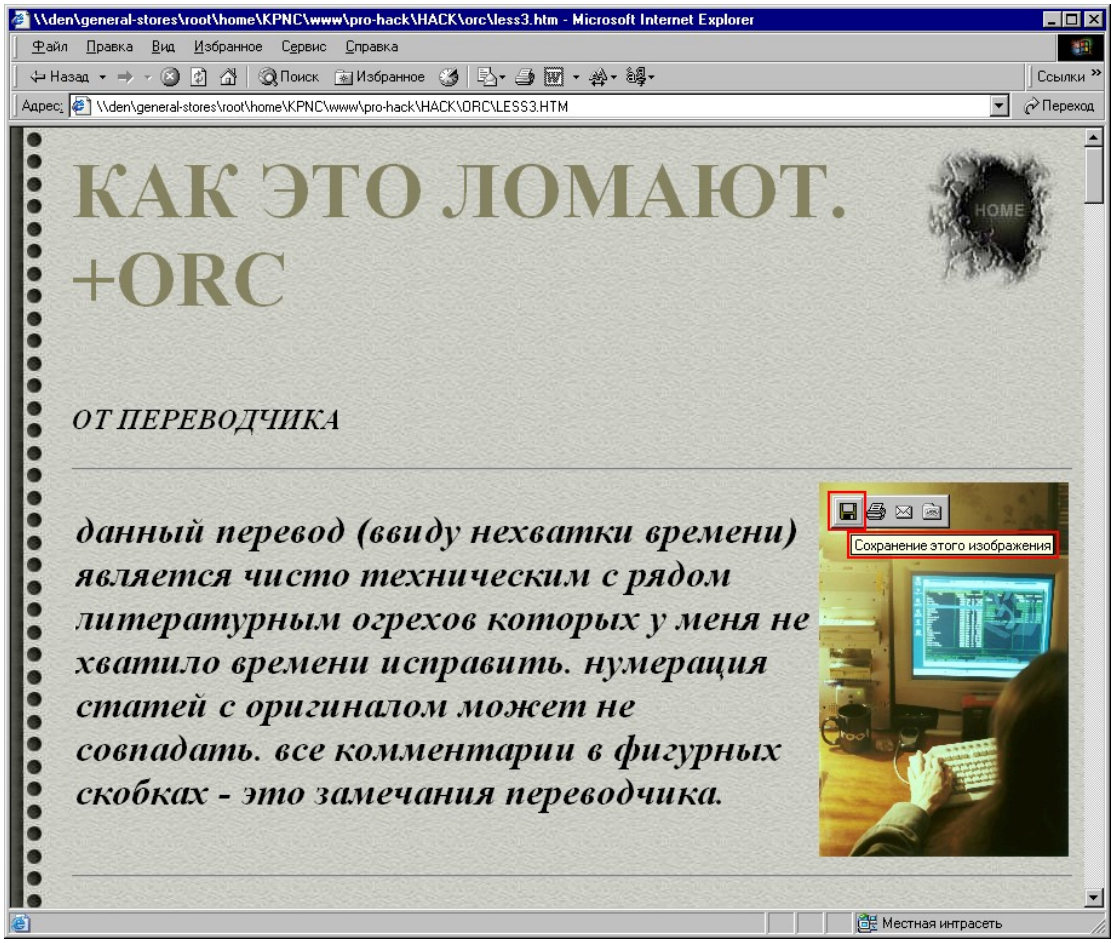

Рисунок 4 обход защиты "правой кнопки" путем наведения мышиного курсора в угол сохраняемой картинки в последних версиях IE

Хорошо, давайте усилим защиту и напишем скрипт, запрещающий не только контекстное меню, но еще и выделение текста, причем не только мышью, но и по комбинации <CTRL-A>, соответствующей пункту меню Правка → Выделить все:

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

	function ops()
	{
		return false;
	}

</SCRIPT>
```

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

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 <SHIF-F10> или клавишей, расположенной слева от правой кнопки <CTRL>.

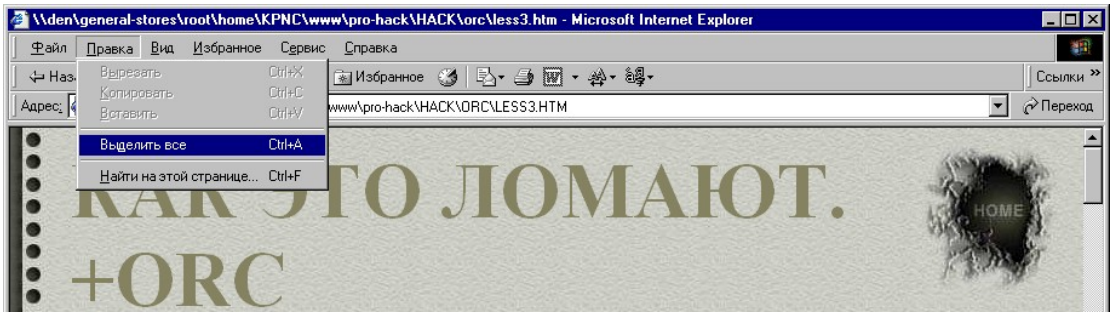

Рисунок 5 заблокированные пункты меню "вырезать", "копировать" и "вставить" в IE, "выделить все" хоть и не заблокировано, но все равно не работает

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 Java-скрипты, возвращая своему любимому браузеру всю его функциональность:

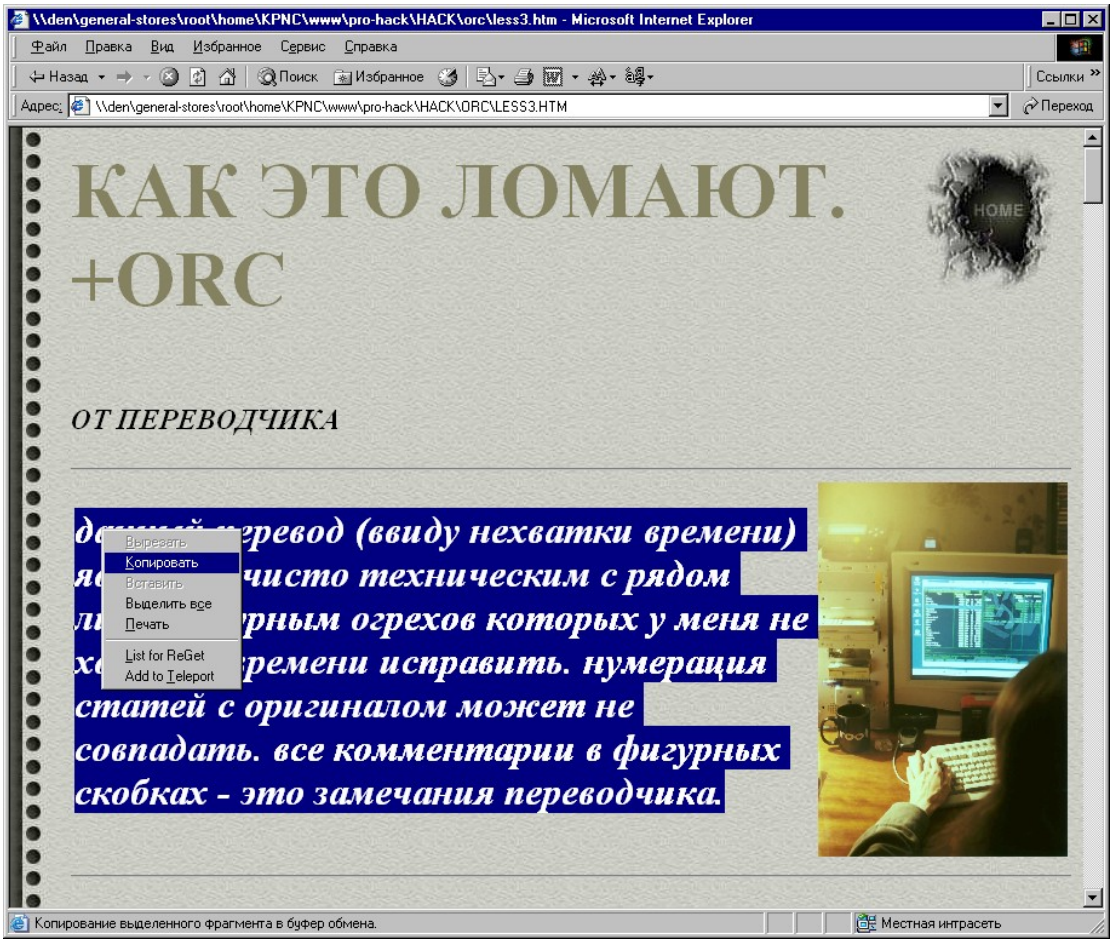

Рисунок 6 обход защиты путем отключения скриптов — правая кнопка сразу же восстанавливает свою работоспособность, а так выделяется и копируется в буфер обмена

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

```
<script>document.write ("данный перевод (ввиду нехватки времени) является чисто
техническим с рядом литературным огрехов которых у меня не хватило времени исправить.
нумерация статей с оригиналом может не совпадать. все комментарии в фигурных скобках -
это замечания переводчика");</script>
```

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

Ниже показан внешний вид защищенной странички с отключенными Java-скриптами (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):

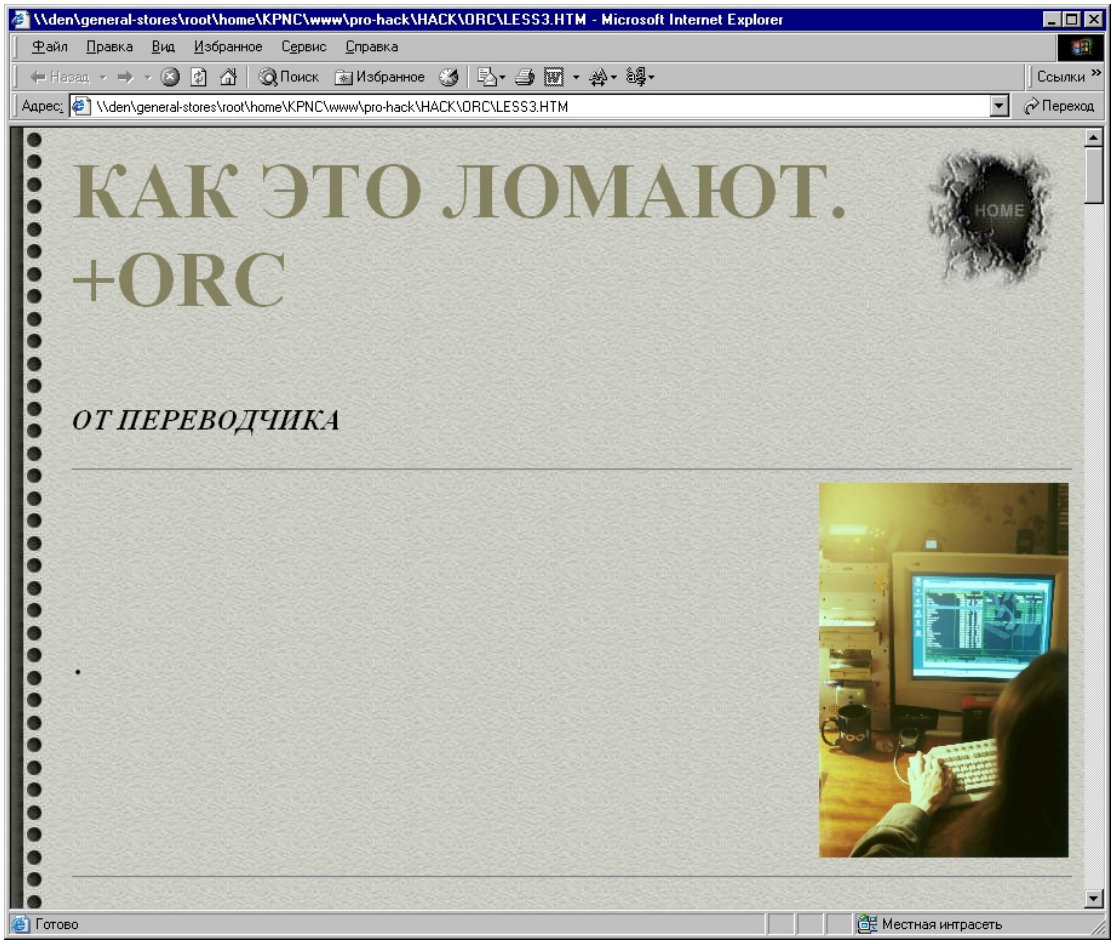

Рисунок 7 внешний вид защищенной странички с отключенными скриптами (исчез ранее присутствующий текст, см. рис. 6)

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: а) 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; б) 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 нестандартным настройками браузера: типа разрешения, масштаба и т. д.); в) 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 (например, если пропущен закрывающий тег </HTML>), 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 web-контента традиционными средствами совершенно бесперспективно. А что если… отображать текстовое содержимое в графическом виде? Идея совсем не так уж и безумна, какой кажется. Ведь преобразование текста в графику не обязательно осуществлять на серверной стороне! Это можно поручить и Java-скрипту (или ActiveX модулю, если это IE).

На сервере мы имеем PHP-модуль, шифрующий текст и передающий его клиенту. Браузер клиента "скармливает" его Java-скрипту/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 web-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 Java-скрипта), хакер без труда напишет программу, расшифровывающую текст, но _не_ превращающую его в графику. Все это так, но… хакеров среди простых пользователей не так уж и много и далеко не у каждого из них есть желание ковыряться в Java-скрипте только затем, чтобы сохранить "первородный" текст. К тому же, подобная защитная методика "нейтрализует" различия между браузерами, поскольку разметкой страницы (в идеале) занимается совсем не бразуер, а наш скрипт! Браузер просто отображает графическую картинку. Прокрутка, изменение размера шрифтов, цвета фона так же обеспечивается скриптом.

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 букетов атрибутов запрета — 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 Java-скрипт, созданный на "коленках" web-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 взлом можно преобразовывать в графику некоторое количество символов еще на сервере (например, 6% или 9%, чтобы не сильно замедлять время загрузки страницы), тогда без OCR'а уже не обойтись.

Популярный FineReader с таким разрешением уже не справляется, но с ним великолепно работает слегка переделенный 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, пускай не без ошибок, но все же пригодный для последующей работы с ним.

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

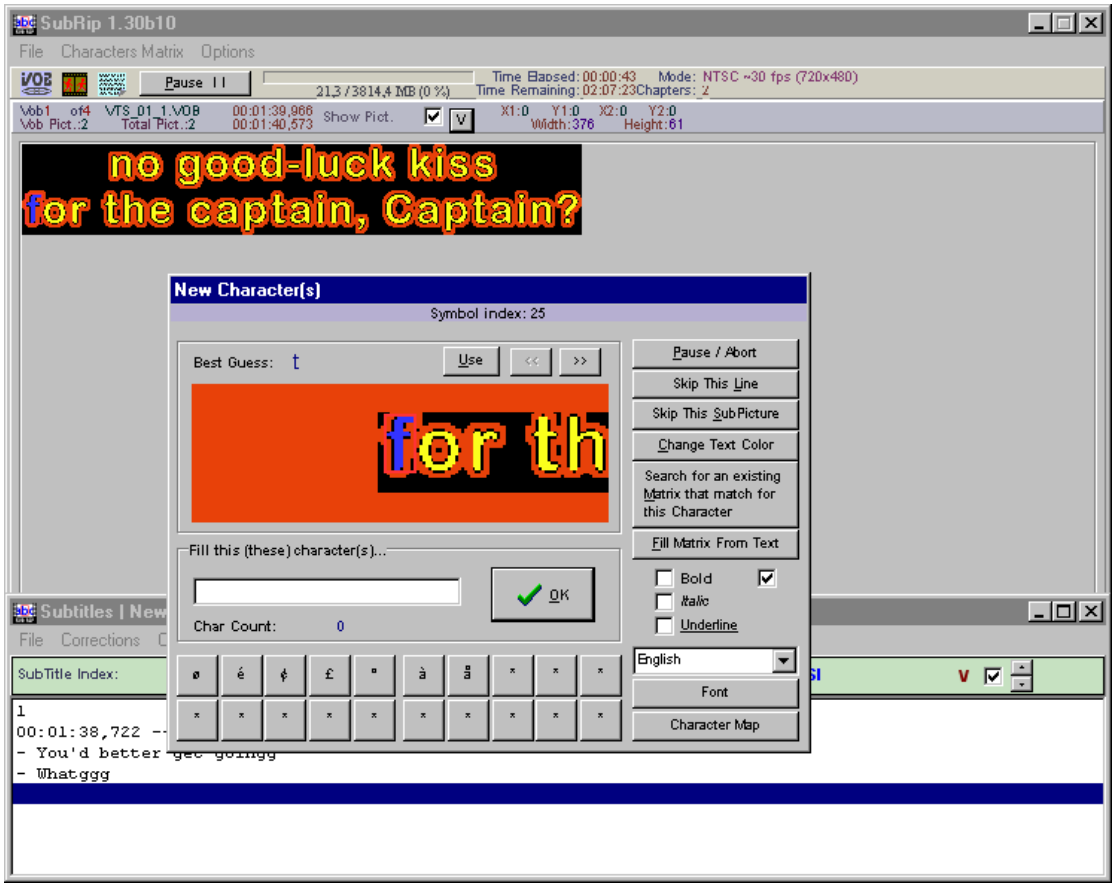

Рисунок 8 внешний вид программы SubRip, обладающий способностью качественно распознавать текст с предельно низким разрешением (вообще-то, он используется для "грабежа" субтитров с DVD, но при желании его можно приспособить и для работы и с другими видами графических текстов)

защита от телепортеров

Основная масса web-мастеров очень не любит, 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 автоматической скачки 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.

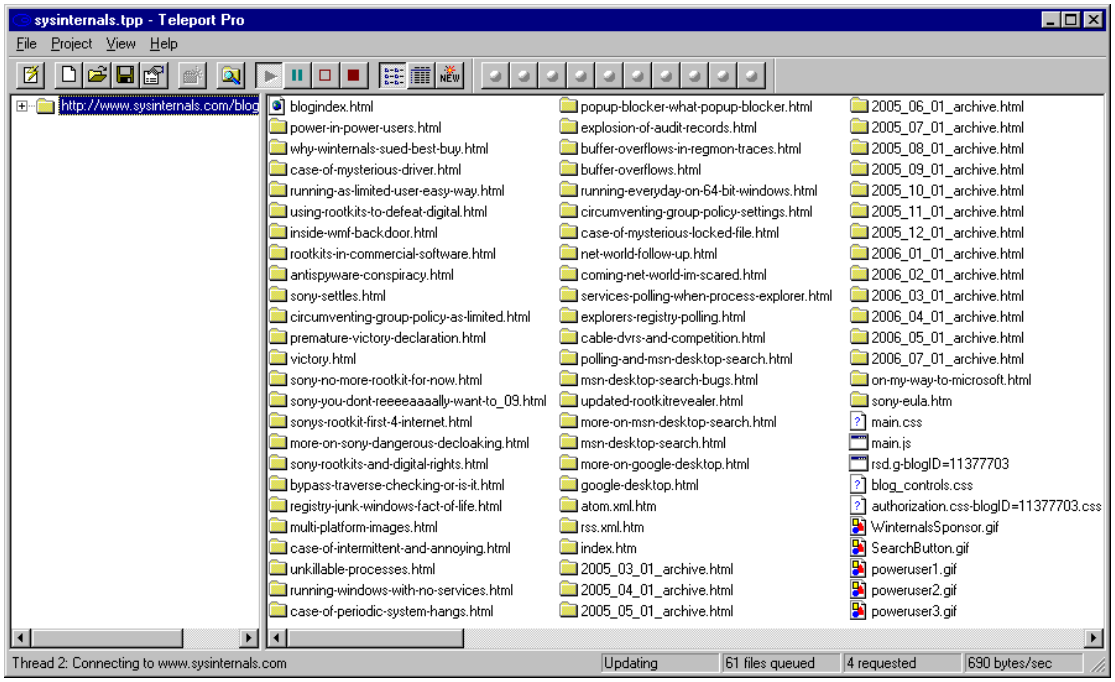

Рисунок 9 сохранение целого сайта на жесткий диск с помощью программы Teleport Pro

Первое, что приходит в голову — это запрашивать подтверждение при скачке 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.

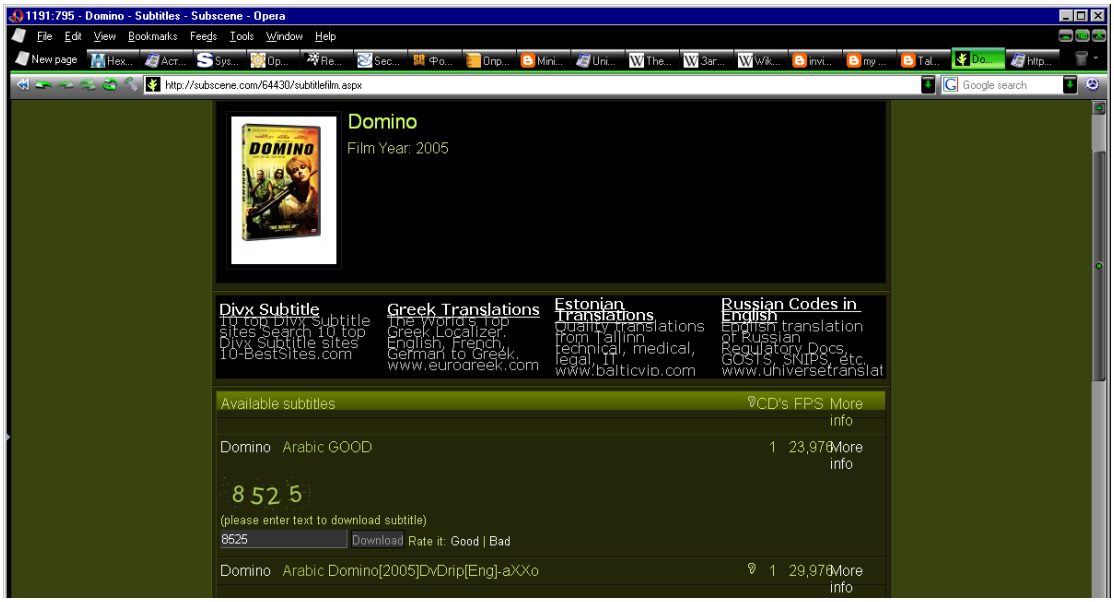

Рисунок 10 использование картинки с искаженными изображениями символов надежно защищает сайт от автоматической скачки Teleport'ом и другими подобными ему программами

Другой механизм предполагает генерацию случайных аплинков, что приводит к "зацикливанию" телепортера. Если web-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 скаченные страницы от тех, что еще предстоит скачать. Теоретически (чисто теоретически!) "телепортер" может анализировать содержимое скаченных 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 начнет web-содержание 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, используя 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 не додумается 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 проверенно на практике), поэтому лучше выделять 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 вырезания 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 web-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, и аудио/видео контент, 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 web-контента следует хорошо подумать: какие убытки это принесет и не превысят ли они "упущенную выгоду" от плагиаторов и пиратов. Практически все защиты, упомянутые в статье, базируется на Java-скриптах, а это значит, что мы теряем пользователей, отключающих Jav'у по соображениям безопасности, а так же всех тех, кто пользуется браузерами в которых Jav'ы не было еще с рождения (к таким браузерам, в частности, относится мой любимый Lynx). Про (не)совместимость различных реализаций Java 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.

>>> врезка ссылка на готовые защитные продукты

- **HTML Protector**:
  - обладает способностью выборочно шифровать HTML-содержимое (Java/VBasic скрипты, e-mail адреса против mail-грабберов, линки, текст, изображения), борется с Teleport'ом 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 так же 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'ом 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'ом 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 данной 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 считывающее 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 так же 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-тоннели или https-proxy сервера).