

Defending Against an Internet-based Attack on the Physical World

Simon Byers
AT&T Labs – Research
180 Park Avenue
Florham Park, New Jersey
byers@research.att.com

Aviel D. Rubin
Johns Hopkins University
Information Security Institute
Baltimore, Maryland
rubin@jhu.edu

David Kormann
AT&T Labs – Research
180 Park Avenue
Florham Park, New Jersey
davek@research.att.com

ABSTRACT

We discuss the dangers that scalable Internet functionality may present to the real world, focusing on a simple yet impactful attack that we believe may occur quite soon. We offer and critique various solutions to this class of attack and hope to provide a warning to the Internet community of what is currently possible. The attack is, to some degree, a consequence of the availability of private information on the Web, and the increase in the amount of personal information that users must reveal to obtain Web services.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous

General Terms

Computer Security, Internet Threats, Automated attacks

Keywords

Computer Security, Internet Threats, Cybercrime

1. INTRODUCTION

One of the things that makes attacks on Internet services more dangerous than attacks in the physical world is the automation that is possible in the world of computers. An example of this can be seen in the resistance of many to move elections online. The worry is that in the physical world, attacks do not scale very well, but as soon as a physical world process is moved online, malicious parties can potentially exploit vulnerabilities in an automated and exhaustive fashion. All of the published studies related to online elections come to the same conclusion. Namely, that the technology is not ready for Internet voting, and that at this time, the security risks are too great [6, 14, 13, 9]. We believe that the risks of online services are not limited to electronic voting.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WPES'02, November 21, 2002, Washington, DC, USA.
Copyright 2002 ACM 1-58113-633-1/02/0011 ...\$5.00.

Over the last several years, we have enjoyed several new Internet services that have dramatically improved our quality of life. The most obvious examples of these are search engines, whose indices cover just about any information that is known in the world, and Web portals, which provide a window to the wealth of information and services available online. In this paper, we demonstrate that we have been living in a state of bliss, spoiled by the lack of any concerted attacks that utilize these new services, search engines in particular. We demonstrate that the current Internet services offer an avenue of attack against physical world processes, and we argue that the only means to defend against these attacks is to dramatically alter the functionality of services such as search engines, to the point where they become much less useful.

As more organizations move online, and real-world processes become automated, it is inevitable that more personal user information is stored in databases. There are companies whose business it is to harvest this information, cross-index it, and compile lists of people and information about them. This information can consist of medical history, personal buying habits, grocery purchases, as well as a history of browsing sessions. The automatic nature of data collection, as well as the ability to invoke programmable scripts against this data makes for an unfortunate environment for users concerned with their personal privacy.

While we concentrate on an example involving search engines and a post office, we recognize that our discussion is but one example of a general vulnerability. As more functionality moves online, and as more services are automated, there is a greater risk that cyber attacks can cause problems that are manifested in the physical world. Defending against these new attacks often requires taking large steps backwards in the convenience offered by technology and often introduces compromises of personal privacy.

2. A WORD ON DISCLOSURE

As security researchers, we often find ourselves faced with a dilemma. When we discover a serious flaw that leads to a vulnerability, do we disclose it to the world, or do we sit on it with the hope that nobody will use it to launch an attack? Our philosophy is that each case needs to be handled independently. If one conceives of an attack that is not likely to happen on its own, and for which there is no prevention, then it is irresponsible to advertise it, and even worse to provide a recipe or exploit code.

However, there is also a risk in not disclosing vulnerabili-

ties for which there are known solutions. By not educating people who are in a position to defend against an attack, it can be more damaging to bury knowledge of a vulnerability than to announce it.

In all cases, sound judgment must be applied and a decision made as to how to minimize the likely effect of the vulnerability. We first conceived of the attack in this paper in September of 2000, but until now have chosen not to publicize our work. The recent availability of an Application Programming Interface (API) for search engines makes the attack much more likely. In addition, we have developed enough countermeasures, that we feel that the disclosure of this paper is important so that proper steps can be taken to prevent the attack from happening. Not everyone will agree with our decision to disclose the attack, but if we knew about it and did nothing, and then the attack was launched, we would be guilty of negligence. It is our judgment that the time has come to reveal this threat.

2.1 Search Engine APIs

Recently search engines have launched developer APIs, as front ends to their databases. We believe that this has important effects relevant to the attack presented here:

- It will cause people to think more about the legitimate and nefarious uses of programmatical interfaces to Web indices.
- It implements a programmatical interface to search engines.
- It facilitates access to private information about people.

It is the release of developer APIs and their facilitating of the attack that convinces us that it is time for disclosure. A press release has claimed 10,000 developers signing up for use of a particular API and release of attack in executable code form would lead to millions of people being able to run the attack as “script kiddies”.

3. THE ATTACK

The attack involves causing large amounts of physical junk mail to be sent to a targeted individual at their physical address. We describe this attack by walking the reader through its implementation. This consists of using three simple tools that we wrote, and applying them to Web pages written in HTML, utilizing the HTTP protocol. The three tools are a URL search interface, a HTML form parser and a HTTP request constructor/submitter. In general using tools such as these leads to powerful analysis methods and can also be used to quickly build new and interesting Web services. As with all powerful tools there tends to be the capability for so called “dual-use”, where one of the uses is potentially damaging or subversive and the other is too good to live without. We have found uses for these tools in such diverse areas as Internet topology measurement, fraud detection, business process monitoring and infrastructure analysis, all of which can bring great gains to businesses willing to employ sufficiently skilled technical staff. The attack described here is a prime reason for wishing to restrict general use of personal address data, whether it be telephone number, snail mail addresses, or even IP addresses and email addresses.

3.1 URL searching

This tool is simply a programmable interface to a major search engine. It allows Web searches to be made and large amounts of results to be processed automatically by other programs. As an example, we search for the terms

```
request+catalog+name+address+city+state+zip.
```

This returns thousands of business catalog request forms that exist on the Web. This is a standard tool in any serious online data mining analyst’s toolkit, and the versions we use generally output URLs on stdout for use in Unix pipelines. A simple yet functional instance of this tool can be written in fewer than 100 lines of Perl. We frequently use more subtle and complex versions but have not developed our tools beyond standard for this application. Figure 1 shows sample output of this search.

3.2 Form Parsing

The results of the search are then processed by a standard form parser, the second tool. This takes each URL passed to it, performs a HTTP GET request to retrieve the content and then analyses the structure of the HTML to extract the URLs of any form resources, along with their advertised and hidden inputs. All such forms found are then stored in a standardized manner to be passed on for further processing. Figure 2 displays the Web page found at the url

```
http://www.whitehatsecurity.com/catreq.html
```

This is the highlighted URL from the list of URLs in the example in Figure 1. (We have altered the domain name for demonstration purposes.)

Figure 3 shows the representation our tool constructs of the form to be found in our sample Web page. The first line lists the HTTP method and resource URL, while subsequent lines show the various inputs which are recorded showing their type, name and possible or default values.

Note that this entirely typical catalog request form is easily machine parsable and even gives the option of sending more than one catalog to the victim. The form parsing program is a standard tool that we use in various legitimate applications and is also approximately 100 lines of Perl. Professional spammers employ sophisticated programs of this nature in their address gathering activities.

3.3 HTTP POST request submission

Now, given each of the forms found and documented in the previous steps an attacker can submit data of his choice to each one. The attacker takes a given name, address, city and state and for each form, then builds and submits the POST request, filling in the victim’s personal information into the correct boxes. Note that the construction of these Web requests does not have to be perfect. For example, not all inputs need to be filled out in the form, only those that are required for processing or proper addressing. The generated Web requests then trigger each of the respective companies to mail out a physical catalog. The use of the victim’s telephone and fax machine in rich cases such as the example form above provide an additional avenue for attack.

Following is some sample code, written in Perl, and using the LWP module, that constructs and issues the HTTP POST request that submits a name and address to the catalog request resource shown in Figure 2. This is a hand configured example to illustrate the method and would not

```

http://www.madirect.com/madvet/madcatreq/madcatreq.asp
http://www.storcase.com/catalog/default.asp
http://www.victoriantrading.com/catalogrequest/ordercatalog.html
http://www.whitehatsecurity.com/catreq.html
http://www.videoguys.com/catalog.htm
http://www.tech-line-inc.com/catalog.htm
http://www.learn2.unh.edu/forms/main_catalog.html
http://www.adventureplanet.com/ap_home/ap_catalog.asp
http://www.stano.night-vision.com/html/catreq.html
http://www.pitsco.com/Forms/catrequest2_hdr.html
http://www.isupress.edu/maillinglist.html
http://www.microcenterorder.com/request_cat.phtml
http://www.execprog.org/catalog/cat-desc.htm
http://www.sporthill.com/catalogsent.htm
http://www.concentric.net/~phanes/phanes-info.html
http://www.timber-press.com/contact/catrequest.cfm
http://66.120.237.202/legend/inforequest.asp
http://www3.mailordercentral.com/petdoors/inforequest.asp
http://www.outreach.washington.edu/request/dlrequest.asp
http://www.iusb.edu/~cted/common/catreq.htm
http://www.nicholsgardennursery.com/catalogrequest.htm
http://www.tomar.com/requestcat.asp
http://www.glorybee.com/catalog.html
http://www.accucut.com/international/catalog.htm
http://www.taperesources-store.com/store/inforequest.asp
http://www.heifer.org/forms/request_gift_catalog.cfm
http://www.advantech.com/catalog_request.asp
http://www.port-specials.com/catalog.cfm
http://www.biomol.com/store/orderCatalog.asp
http://www.trainerswarehouse.com/shop/inforequest.asp
http://www.sharperimage.com/us/en/account/requestcatalog.jhtml
http://www.spoon.com/catalogrequest.phtml
http://www.bandmill.com/catalog/index.asp?site=bandmill
http://www.teamencounter.com/requestCatalog.asp
http://www.firecatalog.com/customer_service/catrequestimport.html
http://www1.edresources.com/iwsupport/er/catalogreq/maillst.cfm
http://www.technoscout.com/general/catalog/catalog.asp
http://www.audioadvisor.com/ordercatalog.asp
http://www.grainger.com/Grainger/help_catalog_request.jsp
http://www.glasshouseworks.com/catreq.html
stdout (4%)

```

Figure 1: The output of a scripted catalog request page search. The highlighted URL is used in our parsing example.

be used in a mechanized attack. We have formatted it for easier reading but it is really only three lines of code. It shows us filling in only the information that we think would be needed to actually get the catalog sent. It is probable that many forms on the Web are filled out to this level of incorrectness during real human submissions.

```

$url = 'http://www.whitehatsecurity.com/catreq.html';
$req = POST $url , [
    FNAME    => $victims_name,
    company  => $victims_name,
    Address1 => $victims_address,
    City     => $victims_city,
    State    => $victims_state,
    Zip      => $victims_zip
];
$webdoc = $browser->request($req);

```

The ability to construct arbitrary POST and GET requests is a commonly used functionality for data gathering and analysis on the Web. It can be done in just a few lines of Perl using standard modules and can yield many useful functionalities, including spoofing browsers, circumventing Javascript checks and impersonating an individual through cookie manipulation. These techniques are commonly used in streamlining power usage of the Web and also allow greater freedom for individuals who wish to avoid

the arbitrary privacy invasion practiced by many Web sites such as third party ad providers.

3.4 Variations

The attack presented above targets an individual by causing a large volume of junk mail to arrive at a particular address. A more sophisticated attack targets the post office infrastructure itself. Such an attack could utilize an online telephone directory service to target many individuals who live in the same area and share the same post office. Such an attack could severely interfere with the ability of the post office to handle regular mail. A scenario could be imagined where an attacker would do this to delay the arrival of an important letter, to wreak havoc on the postal system for political reasons, or even worse, to serve as a diversion for a terrorist act, such as the mailing of a contaminated letter.

Another variation on the attack has the potential to cause specific emotional distress to targeted individuals. In this scenario, the attacker picks a victim with a particular belief, or of a particular group such as a race or religion, and performs the attack where the mail that is sent is material that is known to be offensive to the particular group. For example, one could imagine a scenario where a malicious attacker with personal knowledge of the victim causes colorful literature on gourmet chocolates to be sent to the house of

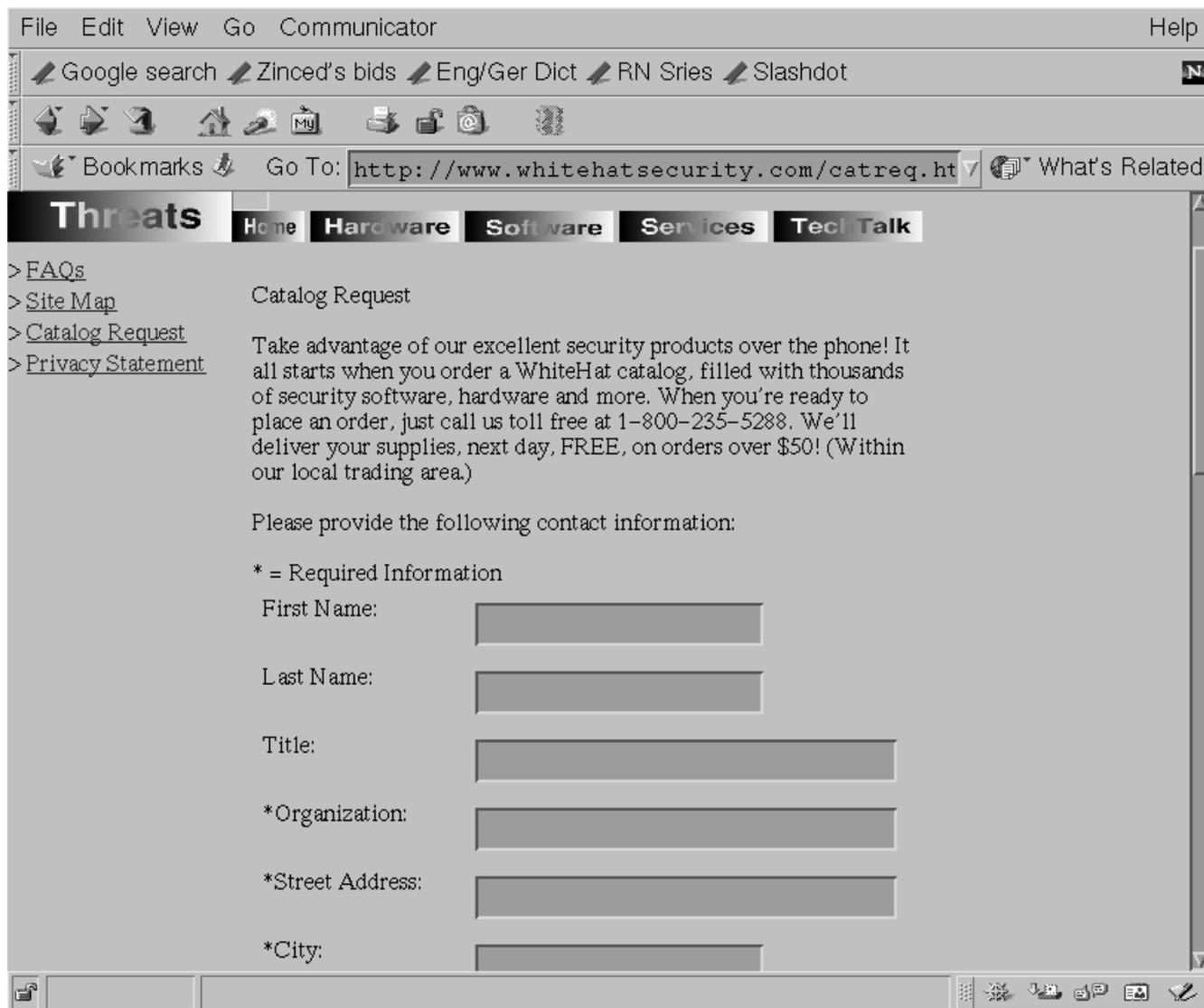


Figure 2: A catalog request form shown in the Netscape browser

an obese person on a weight loss program. The reader can imagine more serious scenarios. Such attacks could be very damaging and cause unimaginable distress to the victim. A more whimsical use of this for an unspecified victim might be to send a wave of gardening catalogs, followed by cookery and finally by weight loss, each spread out over several weeks.

Variations in implementing the fundamental search also exist. We note that spidering a business directory or Web portal such as Yahoo! and parsing the listed business home pages for links to catalog request pages can also be quite fruitful. Many tools exist for dissecting HTML to the desired granularity.

Launching the attack with a subtle crescendo rather than a sudden impact may also have served to hide the attack. The victim may never even realize that they are the target of an attack, but may occasionally remark on how much junk mail they seem to be getting recently.

The basic problem is that the convenience and rich func-

tionality of the Web and its applications have the potential to seriously disrupt our world and our way of life in ways that are becoming easier and easier to implement. In particular, as discussed in Section 4.1, the attacks can be carried out so that the initiator is virtually undetectable.

3.5 Detecting an Attack

Early detection of the attack is non-trivial. Sophisticated traffic analysis running at or near the launch point could sound an alarm, but may not be able to prevent it. To combat this, subtle timing and proxy usage variations could be easily employed by the attacker, we have developed and tested such technology already. It is likely that the postal delivery worker responsible for final delivery would be the first person to notice what is happening, by which time it is far too late to employ much prevention.

We anticipate that the recipient would receive catalogs according to some form of dispersed time transfer function, rather than all on a given day. This is because of variations

```

POST|http://www.whitehatsecurity.com/catreq.html
select|State|WI|MS|OR|MT|KY|DC|UT|DE|WV|LA|WY|PA|NC|ND|FL|NE|NH|VA|NJ|\
RI|NM|TN|CA|NV|NY|GA|VT|IA|TX|AL|ID|MA|O|MD|CO|ME|AR|IL|SC|MI|SD|IN|CT\
|OH|WA|OK|AZ|MN|MO|KS
TEXT|TITLE|
TEXT|FNAME|
RADIO|associatecontact|Yes|No
TEXT|EMAIL|
TEXT|phonePfx|
select|qtycatalogs|1|2|3
TEXT|FaxPfx|
TEXT|phone3|
TEXT|LNAME|
TEXT|phone4|
select|WhereShopped|Store|Catalog\Delivery Service|Both|n/a
TEXT|Fax3|
TEXT|Fax4|
select|employees|50+|0|1|8-49|2-7
TEXT|Zip|
select|Country|USA
select|Aboutwhitehat|PC|SR|BC|SS|V|F|EC|Y|CF|VS|IR|N/A|O
TEXT|company|
TEXT|Address1|
TEXT|City|
select|SicCode|10|50|15|52|70|72a|91|72b|75|58|78|79|99|n/a|00|01|20|4\
0|60|07|08|80|63|81|27|82|73a|65|83|48|73b|49a|86|49b|87a|87b|87c|87d
select|CatalogType|BSELL|FURN|BGUID|N/A

```

Figure 3: The representation of our tool on a form found on a sample Web page.

in the processing of the mailout requests and also the distance they have to travel. It is, though, quite conceivable that a massive number of large pieces of mail could arrive on a given day, dependent on the size of the initial attack. Physical disposal of the mail may prove to be a problem depending on the nature of local recycling facilities. We also note that due to mailing list resale there would be a secondary wave of mailings that would continue for quite some time and that once the name and address combination is tarnished by this attack it might never again be free from junk mailings.

4. ANALYSIS OF THE ATTACK

We conducted some simple tests to gage how well the attack would proceed if launched. A key step where the attack could fail is HTML parsing for form structures. The reasons for this are twofold. First the form might not have fields that an attacker can automatically recognize and use, and second, HTML can be written in various ways that defy some parsers. In tests, our stock form extractor yields actionable results for around 78 percent of candidate Web pages from a relevant search. For these tests we define a form as actionable if we can extract all of the apparent address, city, state and zip code fields via a simple regular expression match on the stored form schema. We did not tune our form extractor for these types of forms in this case although we have done so, with good results, for legitimate applications.

4.1 Tracing the Attacker

This attack can be launched in a way that makes it hard to trace by loading the required software onto a floppy disk and paying cash for a few minutes on a machine in a cyber-cafe. It can be instantiated with multiplicities of tens of thousands quite easily. Once launched it is very difficult to stop as the attack invokes semi-automated processes in

thousands of different places distributed in space, through jurisdictions, and across the Internet. Prevention would be very difficult even given the URLs of all the catalog request forms hit.

Traceback of the attack once detected at the endpoint is simple enough given merging of Internet server logs, but that only gets the launch point. Another possible sanitized launch point is a drive-by launch that uses various open 802.11b networks. Some major cities have several hundred or even thousands of open access points which one might distribute traffic amongst. If the launch access method is anonymous then the perpetrator can only be traced through any existing connection with the victim, i.e. the motive for the attack. If the attack is random, as many Internet DOS attacks may be, then the perpetrator stands a good chance of remaining unidentified.

Another way that the attack could be hidden is if the Web requests are launch using a Web anonymizing tool such as Crowds [12] or Onion Routing [10]. These tools provide a layer of anonymity for Web requests, in effect shielding the identity and location of the requester from observers on the network and end servers.

4.2 Recovery from the Attack

A primary goal of the victim after attack detection and initiation of traceback is prevention of further mail. This should prevent repeat mailing by the primary attack points and also help stem the secondary mailing lists – those who purchased the name and address from the first wave. Informal research we have conducted into systematic mailing list name removal suggests that removing one’s name from a large number of lists is an extremely frustrating and time consuming task. Frequently the removal mechanisms are obscure or inaccurately documented. For example, telephone IVR systems place the removal option in a low priority po-

sition. In addition, many sites use the *removal* procedure as a “sign of life” indicator, telling them that the name and address are valid, and for less scrupulous vendors, this can lead to an increase in the mailings rather than removal from the list. In some cases we discovered that after following removal instructions, we were led down a dead end trail with required menu selections that did not exist, or instructions that were not possible to follow.

5. PREVENTING THE ATTACK

There is no simple way to prevent the attack presented above. We describe several possible approaches to mitigating this or similar attacks and comment on their usefulness.

5.1 Restrictions on Access to Indexing

One possible prevention mechanism is to restrict heavily the use of indexing on the Web. This would mean that search engines might have to ensure that nobody can get a sufficiently scaled view of their index to mount any substantial attack. While this solution may not seem realistic, it is not unreasonable to assume that a legal requirement could be imposed mandating this if attacks such as the one described above become common place and the mail system ceases to function.

We see this approach as undesirable to the greater Web community and ultimately ineffective. With the advent of the search engine APIs, people can build front end programs to their favorite search engine, and any attempt to partition the index could be circumvented with automated multiple queries combined through so-called *keyhole database imaging* techniques [2]. Despite this, we view the potential for restriction on access to indexing as an overreaction that would not be unprecedented by our legislative bodies.

An example of this might be request and results metering. A Web index could limit the number of queries accepted and the number of individual results served to a given client in any given amount of time. One obvious problem with this is that there are large classes of users whose requests are indistinguishable from the point of view of a server because they share a common proxy, as is the case with all AOL users. In addition, request metering is trivially circumvented by an attacker with patience tuning his requests to come in under the radar. The gathering of forms for use in later attacks can be done slowly over a period of time. A technique that we call *keyword salting* can be used to ensure efficient gathering of results. This entails salting a set of keywords specific to the task with randomly chosen external additions that are likely to form an approximate partition. For example, spidering a Web portal for hyperlinked keywords and salting each search with a randomly chosen one from those gathered would ensure that many small searches would not overlap too much in their results.

Simple experiments in keyword salting indicate that a 10-fold salting using some arbitrarily chosen common words can yield an efficiency of around 70 percent. Specifically, in place of a single search, perform 10 searches each with a different added salt, and find the overall number of unique results. The total number of unique results from the salted search is 70 percent of 10 times the original number of hits. We have not yet carried out studies into how tunable the salting words are, but we commonly use this general method in legitimate Web data gathering activities.

5.2 HTML Obfuscation

The attack we describe above would not be as effective if Web sites did not assign recognizable names to the fields storing addresses and phone numbers. However, getting many Web sites to change this is likely to be an uphill battle at best. One of the reasons that we are disclosing this attack is that the preventive measures require cooperation and awareness from a large, diverse set of organizations. The incentives are not perfectly aligned either. The change proposed here, of obfuscating the field names, is something that is required by someone who is not the potential victim of the attack. There is a slight incentive to cooperate, as institutions usually favor protecting their customers, but in many cases, it is not clear that the organizations collecting names and addresses have much to lose if such an attack is launched (besides the cost of mailing out additional materials).

Web sites can comply with the HTML field name obfuscation without much overhead by writing a small bit of code that translates garbage names to more meaningful names at the transition between development and deployment of code. Thus, developers can work with familiar field names such as `Address1`, `City`, and `Zip` and versions such as `dfa4954331b28f23b`, `b19a67514d73127a`, and `16f4b7033ad95` can be visible to the rest of the world.

This defensive strategy may impede the attack at first but a finer and perhaps more effective version of the attack could use an improved form parser that examines the visible text next to each form box as well as the form input names. Further garbling the HTML may make the attack more difficult, it does not completely eliminate it. The attacks can still exhaustively register information at various Web sites that do not employ this defensive strategy. The majority of the registrations would be meaningless, but some nontrivial percentage would be real, and would have the effect of realizing the attack.

5.3 Human in the Loop

One defense against automated attacks is to require that a human being participate in the transaction. To accomplish this, a process must fail a so called *Turing Test* [8], which means that there is a step that cannot be completed by a machine. The most common technique on the Web is to produce an image with some text such that OCR technology is guaranteed to fail, and to then require that the text, or some response to it, be entered into a form. Several sites employ such technology to prevent automated registration or form submission, including www.altavista.com, www.paypal.com, and www.yahoo.com.

There is an active project at CMU called Completely Automated Public Turing Test to Tell Computers and Humans Apart (Captcha), <http://www.captcha.net/> that generates and grades tests that distinguish humans from computers. There is also a paper on printing low quality images such that humans can read them, but OCR fails [4]. In fact, in January of 2002, there was a workshop devoted to this problem titled *Human Interactive Proofs*, and a second workshop is planned. So, this is an active area of research.

There are sites on the Internet that implement a simple check before taking an action, to prevent robots from performing actions. The idea is, for example, to send e-mail to a user's address when he registers, and to require the person to reply to the message or click on a link. While an adver-

sary can automate the reply procedure, this technique raises the bar a bit and makes the attack more complicated.

By placing a *Reverse Turing Test* at every functional interface between cyberspace and the physical world, that is, by requiring that an actual human participate in every transaction that translates from an Internet process to an action in the physical world, the general class of attacks presented in this paper can be mitigated. Such a requirement can be legislated (although we note that the consequences of such legislation is often worse than the attack it is designed to prevent).

5.4 Client puzzles

Client puzzles [5, 7, 1] are techniques designed to force a certain amount of work on a client to prevent a flood of requests. One method is to require a client to exhaustively find a collision in a hash function in some small number of bits, say 30. The server refuses to process a new request until the client solves a new puzzle. Thus, the server can meter access from the client.

For our purposes, we can construct a function which serves both to reduce the rate at which catalogs may be requested and further obfuscate the forms submission process (as in Section 5.2). The server might require the client to compute a unique submission URL by solving a puzzle provided in script form on the Web page. Such a script is easy to add to existing input-validation scripts and requires the attacker to run a Javascript interpreter to submit the form. This increases the complexity of each request and the software that makes the request.

Unfortunately, this scheme is of only limited value. Besides requiring legitimate users to enable Javascript and to do useless computation, the scheme can only moderately slow down an attacker. While such minor slowdown is sufficiently effective to make traditional network denial-of-service impractical, our attacker need only make a few hundred or thousand requests per day (as opposed to per second) to effect the attack.

5.5 Honeypots

Honeypots [3], monitored systems intended to attract attackers for the purpose of early detection and study, have gained popularity recently in the field of network intrusion detection. A *Honeynet* [11] is a network of honeypots with centralized monitoring and reporting. While such systems cannot prevent attacks, they can aid in the early detection of a large-scale attack, allowing steps to be taken to mitigate the effects of such an attack. A simple variation on this concept could be useful in detecting and containing the attack we describe in this paper.

In this strategy, an organization such as the postal service would establish a network of Web servers as honeypots, perhaps located at local post offices, which advertise themselves as offering subscription to a variety of catalogs. Either through cooperation with the major search engines or via more underhanded techniques, the servers would endeavor to obtain high rankings for a search such as those described in 3.1. When a registration occurs at any server, it is communicated to a central monitoring service that, when it detects duplicate or very similar registrations at multiple sites, could take a variety of actions ranging from notifying the victim or local postmasters to alerting a set of registered catalog vendors to be aware of possible fraud.

Such a system is a heuristic tool; it suffers from both false positives (in our case, when a legitimate user registers to receive the nonexistent catalog) and false negatives (when an active attacker either fails to find the bogus catalog registration form or recognizes it as bogus). The distributed nature of a honeynet can help reduce both of these failures. A legitimate client is unlikely to register for dozens or hundreds of catalogs in a short span of time, and having more honeypots increases the likelihood at least one will be found for any given search.

Of course, great care must be taken to ensure that such a system cannot be abused to block or slow the receipt of legitimate bulk mail. One way of mitigating such an abuse might be to for the monitoring organization to work with catalog vendors to ensure that only *new* subscriptions are slowed. The system must also be hardened against abuse intended to create a denial of service against its operators, where an attacker creates the illusion that a massive number of attacks are in progress, with no real intention of actually subscribing any real person to any catalogs.

Enhancements on the basic honeypot are also possible. The monitoring organization could reduce false positives by placing a banner on the pages in a form not likely to be processed by the attacker's software (such as a Java applet or simple image file) which advertises the form's real purpose.

5.6 Stronger Browsing Session Management

While HTTP is a stateless protocol, some defense from the attack can be achieved by enforcing stateful user sessions on the Web. This would greatly complicate the massive submission of user addresses and phone numbers to Web servers required by the attack.

It is clear that the average legitimate user would not submit a Web catalog request form without first loading the page that contains it. It is perhaps also the case that the user would not typically load this page without having been referred there from the business main page or some other relevant part of the Web site. Simple use of the HTTP Referer field might seem to allow a measure of control but that is arbitrarily writable by anybody with a UserAgent capable of implementing the attack (Referer headers cannot be protected via cryptographic MACs in cookies because attackers could easily obtain legitimate MACs with extra queries to the server). Stronger tracking and management of complete browsing sessions by the server would be needed to make this control method effective, and indeed, many content providers currently implement controls of this type.

Such control of user sessions makes deep linking (the ability to link to a URL that is deep within the hierarchy of the Web site) and bookmarks useless. While companies might love the added control and logging that is possible with full user sessions, the elimination of these mechanisms for browsing the Web would represent a great inconvenience to users. Unfortunately, a successful attack of the kind described in this paper could lead to pressure to eliminate deep linking, and in the extreme search engines would only be useful if they returned the root page from a Web site that contains the answer to a query. It would not be useful to index pages that reside deep down in a file system, if Web sites required that pages be accessed after some kind of registration followed by click-through.

From the perspective of user privacy, the elimination of deep linking is problematic. Web sites that control user

sessions have the opportunity to collect information about the users through login screens and survey screens. Also, Web sites can collect information across sessions more easily, and build up databases with user information.

We note that we have had considerable success in spoofing full sessions in order to circumvent the methods discussed here, including management of special session keys and cookies. In the case of the attack under consideration we suspect this could be applied with mixed results, but as noted previously, not every POST request has to succeed for the attack to work.

6. LEGAL ISSUES

We focus the legal discussion on United States law. According to our lawyer, this attack does not appear to fall under traditional notions of computer crime, which typically require some form of unauthorized access to computer resources. See, e.g. 18 United States Code, Section 1030 (“Fraud and related activity in connection with computers”), etc. It is not clear that any of the actions involved in the attack constitutes activity that can be characterized as requiring *unauthorized* access to anything in a typical sense. Nevertheless, the attack might arguably implicate various mail fraud statutes, e.g. 18 United States Code, Sections 1341 (which prohibits schemes to “defraud” using the mail system), 1342 (which forbids the use of the name and address of another in a mail fraud scheme). A large volume attack could also be seen as violating a general “obstruction of mail” criminal provision, e.g. 18 United States Code, Section 1701. The United States Postal Inspectors are authorized to enforce such laws.

If someone were to implement the attack we describe, it is possible that each fake request could result in a count of mail fraud and that the sentences could be consecutive, so the automated attack described in this paper could result in a substantial penalty upon conviction.

7. CONCLUSIONS

Services such as search engines have revolutionized the level of access people have to information. The new world we live in brings the cyber world and the physical world closer together. While in the past cyber attacks were limited to attacking computer systems and data, as the two worlds converge, there exists the threat of physical world attacks originating on the Internet, leveraging scale, automation, and the widespread availability of personal data about people.

In this paper we provide an example of such a physical attack. Awareness of this danger should exist within the post office so that proactive defensive measures can be employed. Aggressive prosecution of perpetrators does not prevent attacks, but may result in consecutive mail fraud sentences. Prevention methods we discuss might prove to be useful, but widespread implementation of some of the countermeasures would interfere important and legitimate Web uses.

The implications of the very existence of powerful search engines are thought provoking after considering the work here, let alone the API. We think that this attack is an important illustration of what may prove to be a large problem in the future. Namely, leveraging of Internet scaled functionality to provide impactful direct physical effects. As more personal information about people becomes available

online, the attacks will be more directed and more powerful. We also feel that responsible disclosure and further study are important in this case to allow solutions to be found to problems that are looming large on the horizon.

8. ACKNOWLEDGMENTS

We thank Ben Lee for helpful discussion on the legal aspects of the attack, Bill Aiello and Lorrie Cranor for helpful comments, and Bennie Pinkas for useful information on reverse Turing tests.

9. REFERENCES

- [1] T. Aura, P. Nikander, and J. Leiwo. Dos resistant authentication with client puzzles. In *Proceedings of the Cambridge Security Protocols Workshop 2000*, LNCS., April 2000.
- [2] S. Byers, C. Silva, and J. Freire. Imaging web databases with restricted query access, 2001.
- [3] B. Cheswick and S. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley Publishing Company, 1994.
- [4] A. L. Coates, H. S. Baird, and R. J. Fateman. Pessimist print: a reverse Turing test. In *Proceedings of the IAPR 2001 International Conference Document Analysis and Recognition (ICDAR 2001)*, September 2001.
- [5] D. Dean and A. Stubblefield. Using client puzzles to protect tls. *10th Annual USENIX Security Symposium*, 2001.
- [6] N. S. Foundation. Report on the national workshop on internet voting, March 2001.
- [7] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In *S. Kent, editor, Proceedings of NDSS 99*, pages 151–165, 1999.
- [8] M. Naor. Verification of a human in the loop or identification via the turing test, 1996.
- [9] N. C. on Federal Election Reform. To assure pride and confidence in the electoral process, August 2001.
- [10] M. R. P. Syverson and D. Goldschlag. Private web browsing. *Journal of Computer Security*, 5(3):237–248., 1997.
- [11] T. H. Project, editor. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Addison-Wesley Publishing Company, 2002.
- [12] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information System Security*, 1(1), April 1998.
- [13] T. D. O. P. The Debate Over Technology. Voting in the information age: The debate over technology, January 2001.
- [14] C.-M. voting technology project. Voting: What is; what could be, July 2001.