

untraceable email cluster bombs

On Agent-Based Distributed Denial of Service

We describe a vulnerability that allows an attacker to perform an email-based denial-of-service attack on selected victims, using only standard scripts and agents. As we will describe, the attack can also target the SMS and telephony infrastructure. What differentiates the attack we describe from other, already known, forms of DDoS attacks is that an attacker does not need to infiltrate the network in *any manner* – as is normally required to launch a DDoS attack. Not only is the attack easy to mount, but it is also almost impossible to trace back to the perpetrator. We describe the attack, some experimental results, and some countermeasures.

The attack involves Web-crawling agents that, posing as the victim, fill in forms on a large set of third-party Web sites (the “launch pads”), causing them to send emails or SMSes to the victim or have phone calls placed. The launch pads do not intend to do any damage – they are merely tools in the hands of the attacker. What makes the attack difficult to prevent is that the launch pads perform the same type of operations as normally desired, thereby giving their administrators no indication that they are part of an attack. This also makes legislation against unwanted emails, SMSes, and phone calls a meaningless deterrent: Without the appropriate technical mechanisms to distinguish valid requests from malicious ones, how could a site be held liable when used as a launch pad? To further aggravate the issues, and given that our attack is a type of DDoS attack, it will not be possible for the victim (or nodes acting on its behalf) to filter out high-volume traffic emanating from a suspect IP address, even if we ignore the practical problems associated with spoofing such addresses.

The attack we describe is an extension of the recent work by Byers, Rubin, and Kor-mann (“Defending Against an Internet-Based Attack on the Physical World”) in which an attack was described where victims are inundated by *physical* mail. While the underlying principles are the same, the ways the attacks are performed and what they achieve are different. Moreover, the defenses proposed in the two papers vary considerably, given both the different threat situations and the different goals in terms of systems to be secured.

Our attack takes advantage of the absence in the current infrastructure of a (non-interactive) technique for verifying that the submitted email address or phone number corresponds to the user who fills in the form. This allows an automated attacker to enter a victim’s email address in a tremendous number of forms, causing a *huge* volume of emails to be sent to the victim. (For concreteness, we focus on the email-based attack, noting the similarities to the SMS and phone call attacks.)

Note here that the “double opt-in” defense routinely employed against impersonation of users is not useful to avoid the generation of network traffic. Namely, some sites attempt to establish that a request emanated with a given user by sending the user an email to which he is to respond in order to complete the registration or request. However, as far as our email-based attack is concerned, it makes little difference whether the emails sent to a victim are responses to requests or simply emails demanding an acknowledgment.

by Markus Jakobsson

Dr. Markus Jakobsson is a principal research scientist at RSA Laboratories, and an adjunct associate professor at New York University. He is engaged in research related to wireless security, privacy issues, and protocol vulnerabilities.



mjakobsson@rsasecurity.com

and Filippo Menczer

Dr. Filippo Menczer is an associate professor of informatics and computer science at Indiana University, Bloomington. His research spans Web, text, and data mining, intelligent agents, and complex systems, and is supported by a Career Award from the National Science Foundation.



fil@indiana.edu

It would not be very difficult to mount an attack with, say, a hundred thousand or a million forms.

In a first phase, an agent would harvest forms from the Web by posting appropriate queries directly to some search engine. The agent can then fetch the hit pages to extract forms. For example, at the time of this writing, MSN reports about 5 million hits for the query “free email newsletter” (all terms required) and over 800,000 hits for “send free SMS.” However, search engines often do not return more than some maximum number of hits (say, 1,000). One way for the attacker’s software to get around this obstacle is to create many query combinations by including positive and/or negative term requests. These combinations can be designed to yield large sets of hits with little overlap.

Once a potential page is identified, it must be parsed by the agent to extract form information. The page value must match a string like “email.” Such a heuristic identifies potential launch pad forms with high probability.

In a second phase, forms are filled in and submitted by the agent. This can be performed right after a form is found, or later, using many already harvested forms. Heuristics can be used to assign values to the various input fields. These include the victim’s email address and, optionally, other information such as name, phone, etc. Other text fields can be left blank or filled with junk. Fields that require a single value from a set (radio buttons, drop-down menus) can be filled with a random option. Fields that allow multiple values (check boxes, lists) can be filled in with all options. The request can then be sent.

The program could be executed from a public computer, in a library, for example, or a coffee shop. All that is required is an Internet connection. The program could be installed from a floppy disk, downloaded from a Web or FTP server, or even invoked via an applet or a virus.

While it is possible for a site to determine the IP address of a user filling in a form, not all sites may have the apparatus in place to do so. Even if the first phase of the attack takes place from an identifiable computer using a search engine, it is difficult for the search engine to recognize the intent of an attacker from the queries, especially considering the large numbers of queries handled. And it is impossible for a launch pad site to determine how its form was found by the attacker, whether a search engine was used, which one, and in response to what query. In other words, the second phase of the attack cannot be traced to the first (possibly traceable) phase.

We will now report on a number of contained experiments carried out in April 2003 to demonstrate the ease of mounting the attack and its potential damage. We are interested in how many email messages, and how much data, can be targeted to a victim’s mailbox as a function of time since the start of an attack. We also want to measure how long it would take to disable a typical email account.

Clearly, these measurements and the time taken to mount an attack depend on the number of forms used. It would not be very difficult to mount an attack with, say, a hundred thousand or a million forms. However, much smaller attacks suffice to disable a typical email account by filling its inbox. Furthermore, experimenting with truly large-scale attacks would present ethical and legal issues that we do not want to raise. Therefore we limit our experiments to very contained attacks, aiming to observe how the potency of an attack scales with its computational and storage resource requirements. We created a number of temporary email accounts and used them as targets of attacks of different sizes. Each attack used a different number of Web forms, sampled randomly from a collection of about 4,000 launch pads, previously collected.

In the collection phase of the attack, we used a “form-sniffing” agent to search the Web for appropriate forms based on hits from a search engine. The MSN search engine was used because it does not disallow crawling agents via the robot-exclusion standard. (We wanted to preserve the ethical behavior of the agent used in our experiments; an actual attacker could use any search engine, since the robot-exclusion standard is not enforceable.) This was done only once.

The collection agent was implemented as a Perl script using no particular optimizations (e.g., no timeouts) and employing off-the-shelf modules for Berkeley database storage, HTML parsing, and the LWP library for HTTP. The agent crawled approximately 110 hit pages per minute, running on a 466MHz PowerMac G4 with a shared 100Mbps Internet connection. This configuration is not unlike what would be available at a copy store. From our sample we measured a harvest rate of 40% (i.e., 40 launch pad forms per 100 search engine hits) with a standard error of 3.5%. At this harvest rate, the agent collected almost 50 launch pad forms per minute, and almost 4,000 forms in less than 1.5 hours. If run in the background (e.g., in the form of a virus), this would produce as many as 72,000 forms in one day, or a million forms in two weeks – probably in significantly less time with some simple optimizations.

The second phase, repeated for attacks of different size, was carried out using the same machinery and similarly implemented code. A “form-filling” agent took a victim’s information (email and name) as input, sampled forms from the database, and submitted the filled-in forms. The agent filled in approximately 116 forms per minute. The email traffic generated by our attacks was monitored until the size of the inbox passed a threshold of 2MB. This is a typical quota on free email accounts such as Hotmail and Yahoo. No other mail was sent to the victim accounts, and no mail was deleted during the experiments. When an inbox is full, further email is bounced back to senders and, for all practical purposes, the email account is rendered useless unless the victim makes a significant effort to delete messages. We call kill time the time between the start of an attack and the point when the inbox size reaches 2MB.

In Figure 1 we can observe that for the three smaller attacks (using 514, 1026, 2050 forms), the kill time occurs well after the attack has terminated. For the largest attack (3911 forms), kill time occurs while the attack is still being mounted. This means that the time to disable a standard email account is less than one hour if around 4000 forms are used. If a million forms were filled in parallel by several agents, or by one agent with high bandwidth, we can see from the figure that a standard account can be disabled in well under a minute.

A first line of defense consists of a simple preventive step by which Web sites can avoid being exploited as launch pads in our attack, but without abandoning Web forms. Web sites would use the following simple strategy. After the form has been filled out, the Web site dynamically creates a page containing a mailto link with itself as an addressee. Legitimate users would send the message to validate their request. The email to the Web site would then be used by the site’s mailing list manager to verify that the sender matches the email address submitted via the Web form. Although the address of the sender is not reliable, because it can be spoofed in the SMTP protocol, the sender cannot spoof the IP address of its legitimate ISP’s SMTP server. The site can thus verify that the email address in the form request matches the originating SMTP server in the validation message.

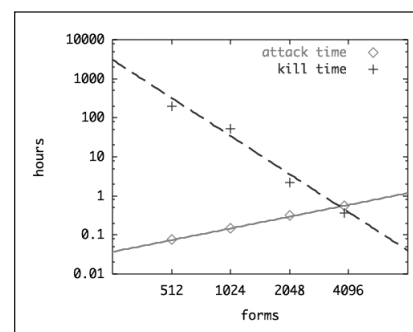


Figure 1

If legislation is enacted that makes sites liable for any attacks mounted using their facilities, then even poorly behaved sites might wish to employ protective measures to avoid being the defendants in lawsuits by victims of the attack we have described.

There are three caveats to this strategy. First, messages via open relays must be discarded by the site. Second, if an attacker could guess that a user in a given domain requests information from some site, she could request information from the same site for other users in the same domain, potentially spoofing the validation created by the addressee. To prevent such an attack, the validation message created by the site should contain a random number with sufficient entropy that it is hard to guess. Third, one could still attack victims who share their ISP's mail server. In general this would be self-destructive, but a disgruntled employee might use such an attack against his employer. In this case, however, the attack could be traced. With these caveats, our preventive strategy would afford the same security as forms that now request email confirmation, but without sending any email to victims.

The above technique works for forms where a party requests information to be sent to herself, but it does not cover common services such as sending newspaper articles or postcards to others. Sites wishing to allow this can use alternative defenses – namely, well-behaved sites may make the harvesting of forms more difficult by not labeling forms using HTML, but rather, using small images. This would increase the effort of finding and filling in the forms. Given the relative abundance of available forms, potential attackers would then likely turn to sites where no image analysis has to be performed to find and fill in the form. Doing this has no impact on human users, except to a very small extent on the download time of the form.

If legislation is enacted that makes sites liable for any attacks mounted using their facilities, then even poorly behaved sites might wish to employ protective measures to avoid being the defendants in lawsuits by victims of the attack we have described.

In our full paper, available from <http://www.markus-jakobsson.com>, we describe further experimental results, along with descriptions of secondary lines of defense that can be used by end users to avoid becoming the victims in attacks where potential launch pads have not taken sufficient precautions to prevent a successful attack from being mounted.