Spam and the Ongoing Battle for Safe Communications

Frank S. Rietta
Peachtree Communication Systems, Inc.
frank@peachtreecommunication.net

Abstract

Managing spam is an increasingly critical function for businesses of all sizes. According to the February, 2007, issue of the Communications of the ACM, the overall volume of spam has increased from 10%, in 1998, to 80%, today [1]. The onslaught of spam significantly degrades the reliability of e-mail for business communication. Many web sites provide web-based e-mail forms instead of listing e-mail addresses. Spammers and phishers have been using increasingly sophisticated techniques to attack these forms. These adversaries use Google to locate vulnerable web sites [5, page 104] and then use botnets[3] to attack them. In order to effectively fight spam, it is important to use multiple tools. This paper examines the overall threat facing every blogger, web site operator, and e-mail user.

1. Introduction to Spam and Various Countermeasures

1.1 Overview

Spam is a big problem that causes significant grief to millions of internet users. Since it first became a major problem in 1997, e-mail based spam has mutated many times in an ever-increasing arms race between spammers and anti-spammers [1]. Spam has traditionally been associated with Unsolicited Bulk Commercial E-mail (UBCE), but the problem is not limited to e-mail. Instant messaging systems have IM Spam (SPIM) [1], chat rooms have SPAT, voice over ip systems have SPIT[7], search engines have spamdexing[8], and blogs have it too. It is cheap to send and profitable[6]. It is important for business owner to take reasonable measures to reduce the detrimental effect, but one must be diligent to avoid entanglement in the search for the final ultimate solution to the spam problem [4].

This paper reviews some of the techniques in common usage to fight e-mail spam. It then introduces a Security-Enhanced Contact Form (SECF) and a few of its countermeasures. Finally, it discusses observations on spambot interaction with SECF and evidence of botnet activity.

1.2 Rule-Based, Machine Learning, and Collaborative Filtering

To fight e-mail spam, systems that implement a combination of rule-based, machine learning, and collaborative filtering are quite effective. This class of system is represented by Apache's Spam-Assassin, a freely available open-source hybrid spam filter. It uses many regular-expression-based signatures of spam characteristics, combined with a simple baysian model trained on both good mail and bad mail, and supports network checks. The supported network checks include checking various DNS block lists and CloudMark's SpamNet (formerly known as Vipul's Razor). While baysian filtering has been around for a while, emerging techniques may perform even better. For instance, compression-oriented algorithms have made use of Dynamic Markov Coding and Prediction by Partial Matching techniques to detect spam in an encoding-agnostic way [1]. These models may prove to be resilient against many spammer tricks used to bypass filters through obfuscation.

1.3 Human-Solvable Puzzles

Completely automated public turing tests to tell computers and humans apart (captchas) are human-solvable puzzles that authenticate a web site visitor as human

Software Industry Conference 2007 July 12-14, Denver, Colorado.

rather than machine [2, 1]. The most familiar form is a difficult to read image that must be transcribed at nearly all major web sites. Captchas are typically classified as gimpy, bongo, pix, sound, baffle text, or pessimal print [2]. Good captchas have published source code available for peer-review, like good cryptography, and are currently effective against most automated attacks. They are also very unfriendly to human users who may have difficulty reading the puzzles.

2. A Security-Enhanced Contact Form (SECF)

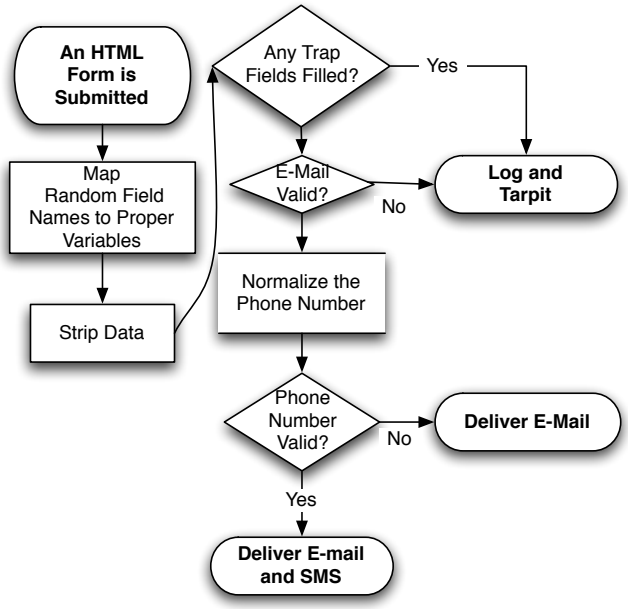
Around 2002, I removed all text e-mail addresses from my web site to prevent spam. For customers to contact me via e-mail, I implemented a simple PHP web form that took the input, formatted it, and e-mailed it the my contact address. The contact e-mail address was only listed in the source code to prevent spam harvesters from picking it up. As an additional benefit to my customers, I had the script generate a text message (SMS) to my mobile phone if a phone number was left. This allowed me to easily provide a call back within a few minutes even when I was away from my home office or computer.

The simple form stopped all spam for a few years and then the attacks began. With each new attack, I improved the security of the script. Below is a summary of broad changes made to keep up with the increasingly complex spambot behavior:

- 2002: Validate, via RegEx, that a single well-formed e-mail address is provided.
- 2005: Strip fields, other than the body, of any content following the first line.
- 2006: Reduce SMS problems by normalizing phone numbers and throwing out invalid ones.
- 2007: Tar-pit bogus submissions to slow down the spammer networks.

The current iteration of the SECF implements several features to defend against automated spammers. Figure 1 shows how submission data flows through the form handler. The core design requirement has been to avoid introducing any burden on a human visitor while making it very difficult for spambots. For this reason, the system does not implement a captcha, but instead relies on randomized field names and trap fields. Because a contact form contains several fields, strict input validation greatly reduces the attack surface. See figure 2 for some of the PHP functions used to validate input, strip spurious input, and tar-pit the spambots.

Figure 1. The data flow through SECF.php, a spam-hostile HTML contact form that uses strict input validation, randomized field names, and trap fields.



2.1 Security Features

No single security feature is sufficient to consistently identify spam submissions without throwing out too many valid submissions. Therefore, SECF implements several smaller metrics and throws out a message only if certain exceptions are triggered. When the message is thrown out it is logged and the request connection is sent to the tar pit function.

2.1.1 Strict Validation for all Input

The programmer must remember to A.R.M. the program against attack. That is, when writing applications, validate all input strings. There are three, and only three, options when given a piece of data:

- Accept it!
- Reject it!
- Modify it!

It might seem obvious that all input must be validated. Too often, webmasters and programmers are fo-

Figure 2. Strictly validating all user-supplied input is important. These PHP functions cut out the junk.

```
function isEmailValid($email) {
   return eregi(
      "^[a-z0-9\._-]+@+[a-z0-9\._-]"
         + "+\.+[a-z]{2,3}$",
  $email
   );
} // end isEmailValid

function oneLine($data) {
   $lines = explode("\n", $data);
   return $lines[0];
} // end oneLine

function delaySpammer () {
   // Delay between 5 and 35 seconds
   sleep(rand(5, 35));
   // Send the spambot to its localhost!
   header("Location: http://127.0.0.1/");
   exit(1);
} // end delaySpammer
```

cused on getting a working application under time pressure and may not implement the best security practices. SECF validates its input. If a field contains unexpected content and error is thrown and no message is delivered. See 2 for the code used to ensure that a single, correctly-formed e-mail address is provided.

2.1.2 Submission Delay

A UNIX time-stamp is generated, and stored in a hidden field, when the HTML form is initially fetched. When the submission arrives, the number of seconds between the fetch time and submission time is computed. Some spam messages had a delay of only 1 or 2 seconds, but others were delayed for 30 seconds or more. The submission delay is not a reasonable filter by itself, but helps serve as an additional data point. A duplicate fetch time indicates out-of-band distributed of cached copies. Submission delays of up to 11.2 days have been also been logged. Table 3.2 shows how a cached copy is shared among multiple IP addresses, indicating a botnet-based spam network.

2.1.3 Randomized Field Names and Trap Fields

My historical data shows that spambots spoof the user agent field of the popular Internet Explorer, FireFox, and Opera web browsers. The bot typically downloads the HTML and fills out the form fields using a simple fuzzy matching algorithm. It puts a name in a field that is called something like “Name”, an e-mail in a field similar to “Email”, a number in a field that is similar to “Phone”, etc. It will typically place the body of its message in any field that it does not recognize.

SECF uses randomized field names in place of normal ones. For example, the “Name” field may really be “oonJif24x1”, the “Email” field really “osdn323xs”, etc. The handler script maps these random fields to the proper variables and validation continues as always. The fields were manually, statically randomized in the current iteration and that has been sufficient to prevent most spam engines because the data supplied to the e-mail field does not pass data validation.

Going one step further, SECF deploys hidden trap fields. The current generation of spambots do not all distinguish between hidden and visible fields and will place content in the trap fields entitled “Name”, “Email”, “Subject”, and “Message”. Any submission with content in any trap field is immediately logged and tar-pitted. See figure 1 for a summary of the submission handler process.

2.1.4 Log and Tar-pit

Starting at the end of March, 2007, SECF logs all rejected messages in a database and tar-pits the spammer’s socket connection for a random period of time. Recall that a message is rejected outright only when any of the trap fields contains data. This is a protection against triggering a tar-pit against a visitor who might have simply entered a bad e-mail address. The resulting data is the basis for the discussion on spambot behavior.

3. Observed SpamBot Behavior

3.1 Before Trap Fields

Throughout the second half of 2006 spams were hitting the contact form multiple times per day from various IP addresses with highly similar content. The user-agents were consistently spoofed as Internet Explorer, Firefox, and Opera. The submission delays ranged from 1 second to 32 seconds. Randomized field names were implemented in SECF in early January 2007, completely stopping the spamming that had been getting through. No spam was delivered through the form and there was no known case of a human visitor being rejected.

Table 1. Below is the log evidence of suspected botnet-based spamming activity. Multiple IP addresses, from three Regional Internet Registries shared a single HTML form that was fetched on March 25, 2007.

Date	Delay (Days)	RIR
2007-03-26	0.9	ARIN
2007-03-27	2.02	RIPE
2007-03-28	2.89	RIPE
2007-03-29	4.03	RIPE
2007-03-29	4.03	ARIN
2007-04-01	6.78	ARIN
2007-04-02	7.97	ARIN
2007-04-05	10.93	APNIC
2007-04-05	11.21	ARIN

3.2 After Trap Fields

The trap fields were implemented on March 8, 2007. Eighteen distinct trapped messages were logged between March 18, 2007, and April 29, 2007. These messages were delivered by fourteen distinct IP addresses such that one IP address delivered nine messages, another delivered two messages, and the remainder each came from various IP addresses. Among the messages, only eight unique fetch times were present.

It is highly likely that a small botnet, or a leased portion of a larger botnet, is being used to deliver these spam messages. A single cached time-stamp, that represents a moment in time down the the second, is being shared among a bunch of unique IP addresses from different regional internet registries (RIR). Table 3.2 shows the data for trapped messages received that were all based on a single, cached copy that was fetched on Sunday, March 25, 2007 at 20:37:15 GMT.

4. Conclusion

Placing a custom-designed e-mail form on a web site is no longer sufficient to protect against inbound spam. The spammers are using sophisticated crawlers to locate and reverse HTML forms. The purpose is to spam the particular operator of a particular web site. This is a break from the past where spammers only went after broadly deployed commercial and open-source software. I have shown that cached copies of the reversed forms are even being distributed among multiple botnet nodes. Those nodes independently submit form spam to the target over multiple days without fetching a new copy of the form.

All web sites are adversarial environments and must be treated as such in the design, development, and implementation of all dynamic systems. Those tasked with hardening web forms should consider the various techniques available. In some instances, randomized field names and trap fields may work as well as traditional captchas against automated attacks for the present time.

References

[1] Joshua Goodman, Gordon V. Cormack, and David Heckerman. Spam and the ongoing battle for the inbox. *Commun. ACM*, 50(2):24–33, 2007.

[2] Clark Pope and Khushpreet Kaur. Is it human or computer? defending e-commerce with captchas. *IT Professional*, 7(2):43–49, 2005.

[3] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 291–302, New York, NY, USA, 2006. ACM Press.

[4] V J S. You might be an anti-spam kook if... `http://www.rhyolite.com/anti-spam/you-might-be.html`, Nov 26 2006.

[5] Joel Scambray, Mike Shema, and Caleb Sima. *Hacking Exposed Web Applications, Second Edition (Hacking Exposed)*. McGraw-Hill Osborne Media, 2006.

[6] Bruce Schneier. Crypto-gram newsletter: The economics of spam. `http://www.schneier.com/crypto-gram-0402.html#9`, Feb 15 2004.

[7] Bruce Schneier. Schneier on security: Combating spam. `http://www.schneier.com/blog/archives/2005/05/combating_spam.html`, May 13 2005.

[8] Tanguy Urvoy, Thomas Lavergne, and Pascal Filoche. Tracking web spam with hidden style similarity. In *Proceedings of the Second International Workshop on Adversarial Information Retrieval on the Web - AIRWeb 2006*, pages 25–31, Seattle, WA, USA, 2006. Department of Computer Science and Engineering, Lehigh University.