Spammer and Hacker, Two Old Friends

Pedram Hayati, Vidyasagar Potdar
Digital Ecosystem and Business Intelligence Institute
Curtin University of Technology
Perth, WA, Australia
pedram.hayati@postgard.curtin.edu.au, v.potdar@curtin.edu.au

***Abstract*—Spammers[1] are always looking for new ways to bypass filters and spread spam content. Currently, spammers have not only improved their spam methods but have also moved towards exploiting software security vulnerabilities in order to spread their spam content. Spammers use weaknesses inside web applications to inject their spam content into legitimate websites, redirect users to their own campaign, misuse web users resources, and hide their footprints. In this paper, we investigate security vulnerabilities that are exploited by spammers. We explain these security vulnerabilities, list their importance and provide a scenario of how spammers can exploit them. Additionally, we discuss two possible solutions to counter problems by patching and secure software development. The result of our work highlights importance of concerning security best-practices in developing secure software which lack of that would result to demotion of website popularity, blacklisting of website and lose of users' trust.**

I. Introduction

Web spam refers to webpages that are created to manipulate search engines or deceive users [8], [7]. Web spam is defined as webpages that are created deliberately to trick search engine into offering unsolicited, redundant and misleading search result as well as mislead users to visit unwanted and unsolicited webpages. Spam contents in blogs, wikis, online forums, comments are example of spam in the web. Web spam has become a major problem for quality, sustainability, and trust of content on the Internet. There has been many works done in existing literature to address spam concerns by developing new and robust methods for spam detection (finding spam patterns within content) and spam prevention (identifying and stopping spammers from entering the system). However, as anti-spam filters improve to increase their capability to detect spam, spammers find new ways to bypass anti-spam filters [10]. An annual report released by Cisco in the December 2008 shows that nine in ten email are spam [2]. The report indicates that spam is still a very large problem that has become more and more difficult to solve. Currently, most techniques that have been proposed to detect spam is done by finding spam patterns within content or they attempt increase barriers of entry such as increasing CPU consumption and the time required to spam [10]. However, to the best of our knowledge, literature lack of any explorations concerning how spammers are exploiting security vulnerabilities in software (e.g. web applications) to achieve their purposes. For instance, by utilising *Trojan horses*, *computer viruses* and *worms*, spammers exploit genuine user computers to send unsolicited emails [6]. On the server-side, spammers abuse websites by manipulating their functionality. Another report from Cisco shows that over 90% of threats originate from legitimate domains in 2008 [2]. Google services (e.g. Google Docs) are exploited by spammers to redirect users to unsolicited websites [14]. These are examples of many other legitimate websites that have been compromised by spammers which indicates that spammers have stepped into security world. This problem not only makes web hard for spam detection and prevention but also makes spamming[2] a big security threat for systems [13]. Figure 1 illustrates an example of a legitimate website that has been exploited by spammers for keyword stuffing and Figure 2 shows an spam email sent from a legitimate website resulting of "abuse of functionality" vulnerability.

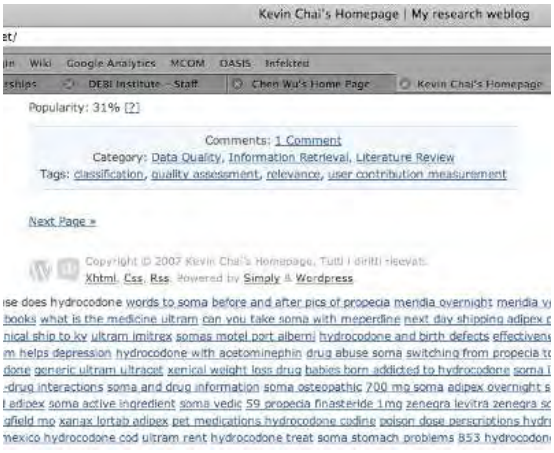

Fig. 1. An attacked blog by spammer

In this paper we identify and analyze security server-side and client-side vulnerabilities which are exploited by spammers to publish spam content and security risks that spammers pose to web. We explain these security vulnerabilities, list their importance and provide a scenario of how spammers can exploit them. According to our understanding there are no apparent work done in this domain. The remainder of the paper is organized as follow. In Section II we attempt to study spammers and hackers motivation to acquire an understanding of techniques they may use to achieve their desired results. A comprehensive identification and analysis of

[1] a person who distributes spam content

[2] the art of sending/publishing spam

978-1-4244-2346-0/09/$25.00 ©2009 IEEE

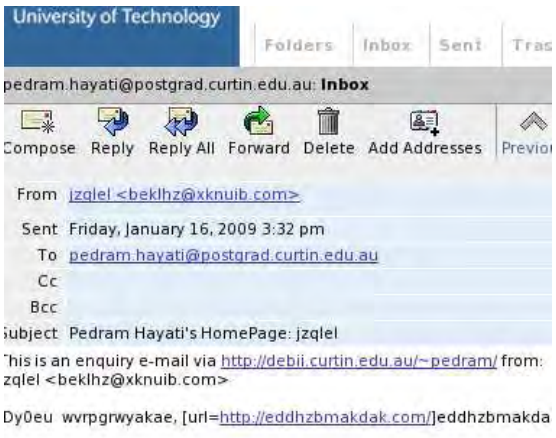

Fig. 2. An spam email made by Form-to-Mail script

security vulnerabilities is discussed in Section III. In Section IV we discuss possible solutions to overcome spam concerns and we conclude our paper in Section V.

II. Spammers and Hackers Motivations

In this section we will study the motivations of spammers and hackers. This makes it easier for us to understand techniques and methods they employ. Although spammers and hackers sometimes have similar desires, we try to differentiate these motivations into two categories. We will begin by evaluating spammer motivations in the next section.

A. Spammer Motivations

According to [10] we can classify spammer motivations into 5 categories:

1) *Generating revenue*: by advertising products and services through emails, blog comments, webpages etc. For instance, spammers can receive payment from pay-per-click programs (e.g. Google Adsense).
2) *Increasing webpage ranking in search engines result*: by misleading search engine indexing algorithms, spammers' webpage can achieve a higher ranks in the search results to route more traffic their campaigns.
3) *Product/service promotion*: Companies pay spammers to promote their product and services through bulk email and/or web advertisements.
4) *Stealing information*: Users' personal information is exploited by spammers in order to publish more user-specific spam content. For example spammer can get backdoor access to users computer and run *Trojan horses* to steal users' username and password.

Hacker motivations will be discussed in the next section along with their similarities to spammer motivations.

B. Hackers Motivations

There is a large amount of existing research focused towards studying hackers' motivations [11]. Among many motivations, the main motivations for hacker activities are:

1) Greed,
2) Revenge,
3) Challenge,
4) Boredom,
5) and Opportunity

which are basically psychological motivations [11], but there are other motivations in the literature that are similar to spammers motivation discussed in previous section. These motivations include:

1) *Stealing users information*, hackers are always looking for new ways to break into the systems and steal user information [13].
2) *Testing and widespread propagation of malicious codes*, the more victims they found for hosting their codes the better results the can get to refine their codes and techniques for future usages [11][17].
3) *Hosting malicious bots*, nowadays, hackers' attacks do not happen from one place. They spread their malicious bots (e.g. Botnet) and utilise them for concerted attack such as *distributed-denial-of-service (DDoS)* [6].

These similarities make the environment beneficial for spammers to step into the hacker world in order to reach their desires. Therefore, spammers are seen to have utilise hacker techniques to

1) send spam emails,
2) hide their footprint to bypass filters (e.g.: blacklisting),
3) spread their malicious codes for performing mass attacks in future [6],
4) steal users' private information (e.g.: email account username and password),
5) and inject malicious code in legitimate websites.

In the next chapter we identify and analyzed these techniques in addition to discussing how spammers can exploit them.

III. Spam and Software Security Vulnerabilities

Spam not only annoys users and wastes their time but also poses as a security risk on businesses [17][13]. In this section we study security vulnerabilities that are exploited by spammers in order to reach their purposes. We have classified security vulnerabilities into two categories, *server-side* and *client-side*. The former refers to security weaknesses that exist on web servers and web applications (e.g. SQL Injection, Directory listing etc) while the latter refers to vulnerabilities on user computers (e.g. Remote-Code Execution etc).

A. Server Side Vulnerabilities

In this section we list and analyze server-side vulnerabilities that are utilised by spammers. We classify each security vulnerability into three parts - Vulnerability explanation, importance and sample scenario of how a spammer can use the vulnerability.

1) Cross-Site Scripting: Cross-Site Scripting or XSS is a technique used by attacker in order to inject malicious code inside a website which later would be executed in users' browsers [3]. Attackers are able to hijack sensitive

user information (e.g. cookies, sessions etc), redirect users to different websites, and perform other malicious activity on users system. Spammers can utilised this vulnerability to redirect users to their own fraudulent web sites (e.g. phishing attack).

A security report on *Microsoft Outlook Web Access* login panel published in 2005 allowed an attacker to redirect users to other locations for phishing purposes [5]. A sample scenario for this attack is describe as follows. An attacker sends a message to a genuine user with a URL to a users trusted domain. This URL contains malicious code and because the URL is known and trusted by user, he/she would navigate to the URL and enter their login credentials which would then execute the attacker's malicious code. The malicious code can redirect user to other websites.

2) Content Spoofing: In *Content-Spoofing* attackers inject their content into user-trused legitimate website [3] so the injected content would be trusted by genuine users. Spammers can trick users by putting their own content (e.g. they can put their own advertisement links and redirect users to their own websites).

Most of dynamically created frame-based HTML pages are vulnerable to this attack. For instance consider the below code in Figure 3.

Fig. 3. Content Spoofing Vulnerability

```
<iframe name="frame_content" src="
    <?php
        print($_GET['content']);
    ?>
"></iframe>
```

the content of *iframe* source from URL variable *content* (e.g. *http://www.foo.com/?content=foo.html*). Spammers can alter the URL to *http://www.foo.com/?content=http://www.spammerwebsite.com/phishing.html*. Hence, users would see content of *phishing.html* inside the legitimate website.

3) Abuse of Functionality: Abuse of functionality is a group of vulnerabilities that allow attackers to mislead a website functionality in different manner. Attacker can abuse functions of website in order to annoy users and defraud them system [3]. The vulnerable website can serve spammer as middle-software, hence spammer are able to hide their own resources. Some of the *Online Contact Forms* has a recipient email address as parameter along with other parameter when user submit a form. By changing recipient email address spammer can send email address to other recipient. So this vulnerability server as a anonymous and legitimate SMTP server for spammer [3].

4) Domain Name Service (DNS) Cache Poisoning: This vulnerability allows attacker to inject false information inside DNS server. it can be used to route users to IP address with the same domain name [20]. This vulnerability in another variation can occur inside *Proxy server* as well. The only difference is that in proxy server cache poisoning, an attacker injects fake content inside the proxy server cache so users would see the fake content when they navigate to a specific website [18]. By exploiting this vulnerability spammer can route users to their fake pages for phishing and advertising campaigns.

A DNS Cache Poisoning vulnerability reported on *Microsoft Windows 2003* and *2000* allowed attackers to inject false records into the DNS Server [19]. A sample scenario for spammer could be as follows. A spammer changes the record for *www.hotmail.com* inside the DNS server and maps it with his/her own website IP address. When a user navigates to *www.hotmail.com* they would be redirected to spammers webapage instead of Hotmail.

5) Browser Cache Poisoning: This vulnerability is similar to DNS cache poisoning. The only difference being that an attacker can inject fake content into browser cache and it would last there until browser cache is cleared. Meanwhile, the user would see poisoned cache whenever he/she requests that particular website [12]. Spammers can leverage this vulnerability to steal user information or perform phishing by injecting fraudulent pages inside the browsers cache.

A recent vulnerability on *Apache* server allows attacker to run XSS or inject arbitrary *HTTP* headers. By injecting HTTP headers spammers can perform browser cache poisoning. Figure 4 demonstrate sample injection input for browser cache poisoning.

Fig. 4. A sample code for Browser Cache Poisoning

```
%0d%0aContent-Type: text/html
%0d%0a%0d%0aHTTP/1.1 200 OK
%0d%0aPragma: no-cache
%0d%0aContent-Type: text/html
%0d%0a%0d%0a
<html>
    <font color=red>
        hey
    </font>
</html>
HTTP/1.1
```

By sending *Pragma: no-cache* spammer would force the browser to cache the content that follows *Content-Type: text/html*.

B. Client Side Vulnerabilities

In this section we study client side vulnerabilities that are used by spammers to publish their spam content, redirect users to their own websites (e.g. pay-per-click programs) and steal user information. Most of the client-side vulnerabilities can be categorized as *Remote Code Execution* (discussed in the next section) and which can occur inside any software running on the users computers such as instant messengers, email clients and web browsers.

1) Remote Code Execution: A remote code execution vulnerability allows attackers to run remote code inside the user's system. Successfully exploiting this vulnerability can allow an attacker to steals user information and redirects user to malicious website [4]. Spammers can leverage this vulnerability inside user web browsers or mail clients to redirects them to malicious websites (e.g. phishing websites). Recently a security report on the *Opera Web Brower* was published which allowed an attacker to execute remote arbitrary code on user browsers [16]. Also another security report on *Microsoft Outlook* showed that an attacker can execute remote code by manipulating *MailTo* URIs [15].

In this section we classify security vulnerabilities and analyzed them both from a security and spam view points. There are other areas such as *Spywares*, *hijackware*, *malwares* that can be used by spammers, however these are out of the scope of this study.

IV. Discussion

Although this paper presents a problem statement and does not provide solutions for the problems, this section will provide a brief discussion on possible solutions that could be used to this counter problem. We can classify solutions into two approaches, patching each vulnerability separately and adopting a secure software development process. The former approach is so-called "*afterthought*" solution which means that software needs to be patch against each vulnerability that exists after the software comes on to the market. The evaluation of software source code is necessary in order to fix such vulnerabilities. This approach does not provide comprehensive solution to this problem and has many flaws. In the latter approach, security concerns are addressed during software development process.

In [9] a UML model for input validation is proposed. By validating software input against security rules, software would be robust against input tampering attacks, *remote command injection, cross site scripting, buffer overflows etc....*
Secure Software Development Methodology is another solution for addressing security concern in the early stage of software development [1]. This methodology has 4 phases for speculating security requirement, design, implementation and test. Hence, software vulnerabilities would be addressed in before software becomes available on the market.

V. Conclusion

The overall aim of the work presented in this paper is to study spam concern from different view point. One area that spammers has focused recently is using security vulnerabilities inside software to originate their attacks. Hence they can hide their digital tracks and bypass current filters (since attacks originated from legitimate systems). We identified and analyzed a number of vulnerabilities and the way spammers exploit these vulnerabilities to achieve their desires. Additionally, we presented 2 possible solutions to this problem which include patching each vulnerability separately and/or using a secure software development process to address security vulnerabilities inside software. The result of our work highlights importance of concerning security best-practices in developing secure software which lack of that would result to demotion of website popularity, blacklisting of website and lose of users' trust

References

[1] A. Apvrille and M. Pourzandi. Secure software development by example. *IEEE Security & Privacy*, 3(4):10–17, July/August 2005.

[2] Cisco. Cisco 2008 annual security report. World Wide Web. http://www.cisco.com/go/securityreport, Dec 2008.

[3] W. A. S. Consortium. Threat classification. Word Wide Web. http://www.webappsec.org/projects/threat/, 2005.

[4] S. Focus. Microsoft internet explorer html objects variant memory corruption vulnerability. Word Wide Web. http://www.securityfocus.com/bid/30610/discuss., 2008.

[5] S. Focus. Microsoft outlook web access login form remote uri redirection vulnerability. Word Wide Web. http://www.securityfocus.com/bid/12459/info., 2008.

[6] D. Geer. Malicious bots threaten network security. *Computer*, 38(1):18–20, Jan. 2005.

[7] Z. Gyongyi and H. Garcia-Molina. Web spam taxonomy. *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '05)*, 2005.

[8] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. *Proceedings of the 30th International Conference on Very Large Databases*, 2004.

[9] P. Hayati, N. Jafari, V. Potdar, S. Mohammadrezaie, and S. Sarenche. Modeling input validation in uml. *19th Australian Software Engineering Conference (ASWEC2008)*, 1:664–672, March 2008.

[10] P. Hayati and V. Potdar. Evaluation of spam detection and prevention frameworks for email and image spam - a state of art. *The 2nd International Workshop on Applications of Information Integration in Digital Ecosystems (AIIDE 2008)*, Novermber 2008.

[11] P. Hoath and T. Mulhall. Hacking: Motivation and deterrence, part i. *Computer Fraud & Security*, 1998(4):16–19, 1998.

[12] A. Klein. Browser cache poisoning using ie and caching servers. Word Wide Web, http://www.securiteam.com/securityreviews/5OP0M15IKO.html,, May 2006.

[13] N. Krawetz. Anti-spam solutions and security. Word Wide Web electronic publication, 2004.

[14] B. Krebs. Spamhaus: Google now 4th most spam-friendly provider, 2009.

[15] G. MacManus. Microsoft outlook mailto command line switch injection. Word Wide Web. http://labs.idefense.com/intelligence/vulnerabilities/display.php?id=673., 2008.

[16] Opera. Advisory: Specially crafted addresses can execute arbitrary code. Word Wide Web. http://www.opera.com/support/search/view/901/., 2008.

[17] M. Reardon. Spam seen as security risk. World Wide Web, February 2004.

[18] SecurityFocus. Squid proxy malformed http header parsing cache poisoning vulnerability. Word Wide Web, http://www.securityfocus.com/bid/12433/info,, 2007.

[19] SecurityFocus. Windows dns cache poisoning by forwarder dns spoofing. Word Wide Web, http://www.securityfocus.com/archive/1/465882,, 2007.

[20] L. Yuan, K. Kant, P. Mohapatra, and C.-N. Chuah. Dox: A peer-to-peer antidote for dns cache poisoning attacks. *Communications, 2006. ICC '06. IEEE International Conference on*, 5:2345–2350, June 2006.

978-1-4244-2346-0/09/$25.00 ©2009 IEEE