SPS: a simple filtering algorithm to thwart phishing attacks

Daisuke Miyamoto, Hiroaki Hazeyama, and Youki Kadobayashi

Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, Japan
{daisu-mi, hiroa-ha, youki-k}@is.naist.jp

Abstract. In this paper, we explain that by only applying a simple filtering algorithm into various proxy systems, almost all phishing attacks can be blocked without loss of convenience to the user. We propose a system based on a simple filtering algorithm which we call the *Sanitizing Proxy System (SPS)*. The key idea of SPS is that Web phishing attack can be immunized by removing part of the content that traps novice users into entering their personal information. Also, since SPS sanitizes all HTTP responses from suspicious URLs with warning messages, novice users will realize that they are browsing phishing sites. The SPS filtering algorithm is very simple and can be described in roughly 20 steps, and can also be built in any proxy system, such as a server solution, a personal firewall or a browser plug-in. By using SPS with a transparent proxy server, novice users will be protected from almost all Web phishing attacks even if novice users misbehave. With a deployment model, robustness and evaluation, we discuss the feasibility of SPS in today's network operations.

1 Introduction

In this paper, we argue that by applying only a simple filtering algorithm into secure proxy servers and browsers, we can thwart almost all phishing attacks. We propose an architecture based on the simple filtering algorithm which we call *Sanitizing Proxy System* (SPS).

A phishing attack is a scam to deceive novice users into disclosing their personal information in various ways [1, 2]. Recently, the number of phishing attacks has grown rapidly. According to trend reports by Anti-Phishing Working Group (APWG) [3], the number of reported phishing attacks was 15,050 in June 2005, increasing from 6,957 in October 2004. The Gartner Survey reported 120 million consumers lost 929 million dollars through phishing attacks[4].

Generally, a phishing attack is composed of two phases: attraction and acquisition. First, *email spoofing* [5] attracts users using a 'spoofed' email, as if it were sent by a legitimate corporation. To acquire the users' personal information, the spoofed email leads users to execute the attached crime-ware, such as a keylogger or a redirector, or to access a 'spoofed' Web site, the so-called "phishing site". The acquisition method using a phishing site is defined as *Web spoofing* [6]. Web spoofing can be categorized by techniques of stealing personal information, downloading crime-ware, cross site scripting (XSS), and deceit. Downloading and XSS cases employ technical subterfuges. On the other hand, in the deceit case only social engineering, that is, the misbehavior of users is employed.

Although countermeasures against technical subterfuges have been studied and practiced [7, 8], there is no sufficient solution against deceit cases of Web spoofing. Ideally, to avoid the deceit cases, every user should distinguish between phishing sites and legitimate sites by themselves, and should pay attention to phishing attacks while browsing Web sites. Considering the growth rate of phishing attacks, however, many novice users are likely to disclose their personal information into phishing sites. Although several proposals have tried to alert novice users about Web spoofings by warning messages [9–18], the effectiveness of these proposals for novice users is suspect [19]. Several application firewalls [7, 8] try to protect users from the deceit cases with URL filtering and/or stateful anomaly detection, they, however, cannot completely rule out phishing attacks because of a trade-off for maintenance costs, error rate and convenience of users.

Our proposal, Sanitizing Proxy System (SPS), focuses on blocking the deceit cases of Web spoofing, and on alerting novice users of phishing attacks. The key idea of SPS is *removing part of the content which enables novice users to input their personal information.* The characteristics of SPS are summarized in the following points:

- *Usability of the filtering algorithm*: SPS employs a filtering algorithm, *two-level filtering.* The two-level filtering is composed of *strict URL filtering* and *HTTP response sanitizing.* Combining two filtering methods, SPS prevents novice users from sending their personal information to phishing sites while allowing novice users to browse other contents of phishing sites.
- *Flexibility of the rule-set*: On filtering HTTP responses, SPS distinguishes between verified legitimate sites and other suspicious Web sites based on a rule-set written by the operator of SPS.
- *Simplicity of the filtering algorithm*: Two-level filtering is a very simple algorithm, which can be described in roughly 20 steps. Hence, it is easy to apply the SPS functions into existing proxy implementations, browser plug-ins, or personal firewalls. In section 3, we show SPS based on two different open-sourced proxy implementations to proof the simplicity and availability of the two-level filtering algorithm.
- *Accountability of HTTP response sanitizing*: SPS prevents novice users from disclosing their personal information to phishing sites by HTTP response sanitizing, that is, by removing malicious HTTP headers or HTML tags from HTTP responses. Through the sanitized Web pages, SPS alerts novice users that the requested Web page contains suspicious parts which are under the threat of phishing attacks at the time.
- *Robustness against both misbehavior of novice users and evasion techniques*: An SPS built-in proxy server can protect novice users from almost all deceit cases of Web spoofing, regardless of novice users' misbehavior and evasion techniques by phishing attacks. Furthermore, applying SPS both in application firewalls and in browser plug-ins can block any Web spoofing phishing attacks.

– *Feasibility of deployment*: It is feasible to deploy SPS into today's Internet operational environment in spite of the maintenance cost of the rule-set.

The rest of this paper is organized as follows: In section 2, we describe related work pertaining to phishing attacks. In section 3, we specify the detail of SPS design, and discuss the feasibility of SPS. In section 4, we verify the function of SPS and evaluate the processing overhead of SPS. Finally, section 5 concludes our paper.

2 Related Work

There are many research contributions to characterize or model phishing attacks [1, 2, 20–22]. Generally, a phishing attack is separated into two distinct phases, attraction and acquisition. In many cases, phishers attract users by *email spoofing* [5]. Email spoofing attracts and leads users to one of the acquisition tricks. One trick attracts users and suggests executing an attachment crime-ware, such as a virus or Trojan. The other leads users to access a phishing site. The acquisition trick, which employs some phishing sites, is called *Web spoofing* [6]. Web spoofing can be categorized into three types; downloading, cross site scripting (XSS), and deceit. Downloading and XSS use such technical subterfuges as the acquisition trick. Downloading attracts users to download and install some of crime-ware, such as the key-logger or redirector. Once users install these crime-ware, phishers can steal a users' personal information arbitrarily. XSS exploits the vulnerabilities of a legitimate site to forward personal information to a phishing site.

While technical means of Web spoofing have been discovered so far, most Web spoofing only employs social engineering, that is, misbehavior and carelessness of users. In the deceit case, a spoofed email convinces users to access a URL leading to a phishing site. The phishing site is well-designed to be a look-alike of the targeted legitimate site; attracted users easily believe the phishing site is the legitimate site, and the users are likely to disclose or input their personal information to the phishing site, without verifying URL or SSL certification. Merwe et al. pointed out a responsibility shift from businesses to users, and suggested the necessity of education both for the users and for the businesses [20]. Many consortia or security vendors have published guidelines against phishing attacks[1, 2, 23].

On technical issues, sufficient countermeasures have not been proposed yet, although several countermeasures against phishing attacks have been studied and practiced in both the users and the businesses [1, 2]. Several spam-filtering methods are effective for email spoofing because email spoofing can be included in the spam email category. Focusing on the email spoofing itself, a lightweight trust architecture has been proposed and evaluated [24, 25]. Although several security tools [9–18] support users to alert the phishing attacks or to confirm URL and SSL certification, Wu et al. indicate that these tools are not effective for protecting novice users [19]. By testing three security toolbars, 34% of test users disclosed their personal information to emulated phishing sites. The reasons why

34% of test users were attracted to the phishing sites are as follows: (i) there are some users who ignore warning messages when the phishing site attracts them good enough, and (ii) there are poorly designed legitimate sites which is hard to distinguish from phishing sites.

Application Firewalls have the possibility to block phishing attacks[7, 8]. An application firewall filters several application layer protocols. The coverage of the application firewall is broad and it includes emails, Web applications, instant messaging, p2p applications, and etc. By using signatures or rule-sets, application firewalls can filter spoofed emails or virus-mails, protect servers from XSS attacks, or prevent users from downloading crime-ware.

It is difficult, however, to thwart deceit cases of Web spoofing by application firewalls. In URL filtering, there is a trade-off for convenience, safety and maintenance cost. Strict filtering, which allows users to access to only legitimate Web sites, diminishes the convenience of the user, because users cannot browse anything about filtered pages. Also, strict filtering hinders administrators who want to catch up to the new phishing sites, although most URLs will be meaningless sooner or later because of the average life-span of the phishing sites [3]. On the other hand, loose filtering, which denies well-known illegitimate Web sites, can reduce maintenance costs and improve the convenience of users, in spite of sacrificing safety. URL filtering methods alone cannot prevent users from disclosing their personal information while allowing them to browse Web pages.

Some application firewalls try to deal with the case of deceit in Web spoofing by combining URL filtering and stateful anomaly detection. Stateful anomaly detection have the possibility of detecting new varieties of phishing attacks by tracking and analyzing every TCP session in real time. Combined with loose URL filtering, stateful anomaly detection can ease the maintenance cost of the rule-set, and improve security while preserving the convenience of the user. Stateful anomaly detection is, however, likely to be hindered by high false positives and false negatives. By improving the accuracy of the detection algorithm, the detection algorithm becomes complex and requires a high frequency CPU and large amounts of memory to analyze all of the TCP flow in real time. ASIC, FPGA, or a Network Processor may accelerate throughput, but these special hardware requires simple algorithms to be written in limited steps.

3 SPS: Sanitizing Proxy System

SPS focuses on the case of deceit in Web spoofing. The key idea of our proposal requires *removing part of the content which enables novice users to input their personal information.* If HTML contents does not contain any input form, a user never inputs, that is, never discloses his or her account name, password or PIN code to phishing sites. In this section, we first analyze the behaviors of novice users and of phishers to formulate requirements for SPS. Next, we describe the details of the SPS design including assumptions and requirements, and discuss the feasibility of SPS from several aspects.

3.1 Assumptions

We assume behaviors of novice users and of phishers as follows:

- Novice users cannot verify a Web site by themselves, or cannot distinguish phishing sites and legitimate sites.
- Novice users may be credulous to access a phishing site easily.
- Novice users may ignore warning messages about phishing attacks, if those attacks are highly attractive to the user.
- Novice users may disclose their personal information without being aware of phishing attacks.
- Phishers deceive novice users in various ways, with changing URLs or tricks in the short term.
- Phishers try to hijack the names of legitimate Web sites which have been verified by some trustworthy third parties.
- Phishers try to evade countermeasures against phishing attacks.

3.2 Requirements

According to our assumptions in 3.1, the requirements for SPS are as follows:

1. *Alerting novice users that they have visited the phishing site.* The alert will help novice users to understand why they cannot input their personal information. Without any alerts, novice users will be confused and wonder what is happened to their browser.
2. *Forcing novice users not to disclose their personal information into phishing sites.* SPS should prevent novice users from disclosing their personal information to phishing sites.
3. *Distinguishing between phishing sites and legitimate sites.* Because SPS must allow novice users to input their personal information into legitimate sites, a distinguishing mechanism between phishing sites and legitimate sites is necessary.
4. *Being independent from misbehavior of novice users.* The SPS must not be controlled by novice users, because phishing is caused by novice users' careless mistakes.
5. *Being as simple as possible.* A simple algorithm facilitates developers to incorporate it into various platforms, and it will have opportunities to be applied in various environments, in user-side applications, or in an Internet Service Provider (ISP) / Security Service Provider (SSP) solutions. Because of tunability and availability, the filtering algorithm for SPS should be as simple as possible.
6. *Making the SPS robust against phishers.* Phishers will try to escape from SPS filtering by various evasion techniques. The architecture of SPS should be designed as robust as possible against evasion techniques.
7. *Being easy to deploy.* In order to compete with the rapid growing of phishing attacks, the quick deployment of SPS must be considered. Hence, the benefits produced by SPS should overwhelm the operational costs.

Algorithm 1 SPS main routine

```
 1: procedure SPS_PROXY main routine
 2: for all  httpRequest from client do
 3:     send httpRequest to server
 4:     receive httpResponse from server
 5:     if server.URL == VALIDATED then
 6:         send httpResponse to client
 7:     else
 8:         apply HTTP_RESPONSE_SANITIZING to httpResponse
 9:         send httpResponse to client
10:     end if
11: end for
```

Algorithm 2 HTTP Response Sanitizing

```
1: procedure HTTP_RESPONSE_SANITIZING
2: if httpResponse.httpStatusCode == MALICIOUS_HTTP_HEADER then
3:     httpResponse.httpStatusCode ← HEADER_SANITIZED_MESSAGE
4: end if
5: if httpResponse.htmlContent includes MALICIOUS_HTML_TAGS then
6:     replace MALICIOUS_HTML_TAGS to TAG_SANITIZED_MESSAGES
7: end if
8: return httpResponse
```

3.3 Two-Level Filtering Algorithm

In this section, we describe the design of a two-level filtering, the key component of SPS. The two-level filtering is composed of *strict URL filtering* (SUF) and *HTTP response sanitizing* (HRS). To filter out the suspicious URLs, the SUF delineates URLs to safe or suspicious ones at first. Then, HRS removes any input forms on every HTTP responses from the suspicious URLs.

Strict URL Filtering

Although the SUF is similar to the URL filtering of application firewalls [7, 8], SPS employs the SUF to tell if each HTTP response came from one of the suspicious URLs. SPS does not use the SUF for an actual blocking method as an application firewall do. Because the SUF does nothing but to check the source of the HTTP response, the algorithm can be described simply, as shown in Algorithm 1.

The SUF rule-set is a set of prioritized allow/deny rule, combined with a varying degree of URL specification, as shown in Figure 1, The rule-set contains the top directory URL of each validated Web site. It also includes URLs which are under the threat of phishing attacks in validated Web sites. By using this rule-set, SPS distinguishes safe and validated URLs from other suspicious URLs.

HTTP Response Sanitizing

```
filter{
    url "http://validurl.com/suspicious/allowed_directory/" {
        priority 1;
        allow;
    }
    url "http://validurl.com/suspicous/" {
        priority 2;
        deny;
    }
    url "http://validurl.com/" {
        priority 3;
        allow;
    }
    url "http://validurl.net/" {
        priority 3;
        allow;
    }
    default {
        deny;
    }
}
```

Fig. 1. Pseudo SPS Rule-set

After delineating valid URLs from suspicious URLs by SUF, SPS applies HRS to every HTTP responses from suspicious URLs. In the context of XSS, a sanitizing method is often employed. Sanitizing the XSS converts special characters to safe characters, since special characters may lead to unexpected actions. For example, sanitizing XSS replaces "⟩" to "&gt;". On the contrary, HRS replaces malicious HTTP headers and HTML tags with warning messages. Table 1 shows the HTTP headers and HTML tags which have the possibility of being used by phishing attacks. Usually, these headers and tags are used to construct input forms for user accounts or passwords. Therefore, a Web page using these headers and tags require users to input their personal information, such as account names or pass-phrases. To prevent phishing attacks completely, it is better to remove these headers and tags from Web pages or from HTTP responses if possible.

HRS alerts novice users about the malicious parts by a sanitized the Web page, as in Figures 2 and 3. The warning messages on sanitized pages are useful for troubleshooting on browsers. It also helps to educate novice users as to what is vulnerable or what type of pages are suspected as phishing sites. Because it replaces any malicious HTTP headers and malicious HTML tags, HRS is robust to possible evasion attacks as long as no phisher hijacks legitimate URLs, as discussed below:

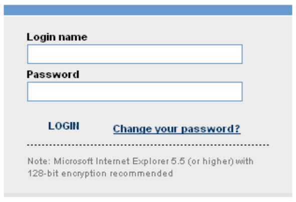
– *Evasion techniques by Authentication Request Headers*:
  HRS sanitizes the malicious HTTP Header listed on Table 1, which pops up a dialog box on the browser and asks the user to input his/her ID and password. Phishers cannot escape HRS by Authentication Request Headers.

Table 1. Malicious HTTP Headers

`401 Unauthorized`
`402 Payment Required`
`407 Proxy Authentication Required`
`WWW-Authenticate`

Table 2. Malicious HTML Tags

`<script>`	`<isindex>`	`<form>`
`<input>`	`<textarea>`	`<select>`
`<applet>`	`<object>`	`<embed>`



Note: Microsoft Internet Explorer 5.5 (or higher) with 128-bit encryption recommended

Fig. 2. Malicious Parts Before Sanitization

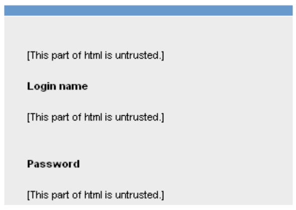

[This part of html is untrusted.]
Login name
[This part of html is untrusted.]
Password
[This part of html is untrusted.]

Fig. 3. Malicious Parts After Sanitization

- *Evasion techniques by CGIs*:
  HRS removes `<form>`, `<input>`, `<textarea>`, `<select>` and `<isindex>` tags (as shown in Table 2) which compose an input form by only HTML. This process is applied to dynamically generated contents as well as static contents. Hence, phishers cannot evade HRS by using CGI.
- *Evasion techniques by Active Components*:
  HRS replaces the `<script>` tags which can dynamically generate documents, which in turn may contain malicious tags. Phishers cannot escape from HRS even with dynamic scripts.
- *Evasion techniques by Dynamic Scripts*:
  HRS replaces the `<script>` tags which dynamically generate malicious tags as shown in Table 2 on a Web page. Phishers cannot escape HRS by dynamic scripts.

In addition to the SUF algorithm, HRS can be written in a simple pseudo code as shown in Algorithm 2, which requires only 8 steps.

The effects of the Two-Level Filtering

SPS is usable for novice users and maintainers, because the two-level filtering can ease the trade-off for convenience, safety, and maintenance costs. While preventing novice users from phishing attacks, SPS does not prohibit novice users from browsing any Web pages, except forbidding them to disclose their personal information to unknown URLs. Therefore, SPS can improve both the convenience and safety of users. Also, because SPS uses the rule-set for strict filtering, the maintainers of SPS are no longer bothered by registering new phishing sites on the rule-set to protect users.

Considering Algorithms 1 and 2, the two-level filtering algorithm is a very simple algorithm, which can be described in roughly 20 steps. We tried to implement SPS in two different open-sourced proxies; tinyproxy [26] and privoxy [27].

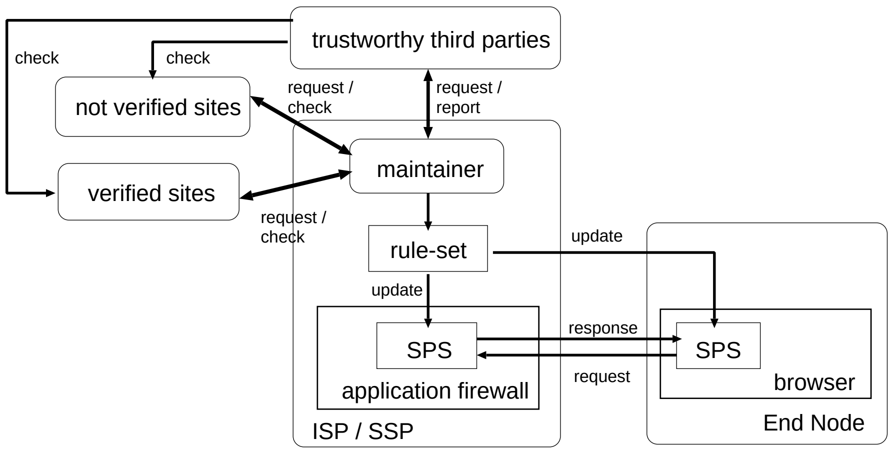


Fig. 4. Deployment Model of SPS

We implemented SPS by adding only 46 lines to the 37,449 lines of privoxy;213 lines to the 17,621 lines of tinyproxy. Hence, SPS has the portability to various proxy implementations with only small changes.

Applying SPS into proxy servers is also efficient. As shown in Figure 4, the SPS can be applied both in a proxy server and in a local proxy as a browser plug-in or personal firewall. According to APWG reports, 94.5% of the port number employed by phishing attacks is 80, the well-known HTTP port [28]. Therefore, applying SPS into proxy servers can prevent novice users from almost all phishing attacks. Of course, this proxy server solution is affected by the proxy setting on the browser used by a novice user. If SPS is implemented as a transparent proxy server, it can be protect novice users from phishing attacks while remaining completely independent from the misbehavior of novice users. Also, combining the SPS with application firewalls by ICAP [29] or built-ins can block not only deceit cases, but also other technical subterfuge cases of phishing attacks. Furthermore, SPS can be implemented in the form of browser plug-in. SPS browser plug-in can even sanitize SSL/TLS-encrypted HTTP responses.

3.4 Deployment Issues

In this section, we explain the deployment of SPS. We previously explained the key idea of SPS and its variations of SPS in section 3.3. When deploying SPS in real network operations, we should consider the trade-off of users' convenience, safety, and maintenance costs.

Essentially, HRS has no false positives or false negatives, and has robustness against evasion techniques. The false positives and false negatives of SPS are caused by the accuracy of the rule-set for SUF. Also, the accuracy of the rule-set affects convenience of users. If some legitimate sites have not been registered in the rule-set by a loose operation, users will complain that the maintainer of SPS has to register the legitimate sites. Therefore, the success of deploying SPS depends on the cost of achieving a sophisticated and accurate rule-set regarding legitimate sites. If such a sophisticated and accurate rule-set can be achieved and shared by both users and businesses, it would be helpful novice users in order to find a safe and trustworthy legitimate site.

We consider a deployment model of SPS as shown in Figure 4. A double-check mechanism among ISPs, CDNs, SSPs, or consortia such as APWG or OWASP, will prevent phishers from registering phishing sites to the rule-set. Also, SSPs, and consortia have already started searching new phishing sites. Hence, the double-check mechanism potentially exists in today's network operations.

We present preliminary *supply and demand* analysis of users, ISPs/SSPs, and Web sites as follows:

– *Users*: Novice users already know their responsibility in dealing with their personal information carefully, however, they do not know or do not operate protecting methods against phishing attacks. Potentially, novice users need a solution to protect them from phishing attacks.
– *ISPs or SSPs*: ISPs or SSPs already know their responsibilities to protect their users or customers from phishing attacks. To reduce maintenance costs and to minimize incidents, they explore a light-weight solution to block phishing attacks completely.
– *Web sites*: Web sites already know their responsibility to eliminate the vulnerabilities of their Web sites; they also ask their customers to pay attention to spoofed emails. Since Web sites may lose their business opportunities due to the customer's fear for phishing attacks, they want a solution to protect them from the influences of phishing attacks.

According to the potential needs, the existing double-check mechanism, the effectiveness of the SPS, and the availability of the SPS, we conjecture that SPS can be deployed in real network operations without major difficulty.

3.5 Robustness against evasion attacks

In this section, we explore the evasion attacks against the SPS and describe the robustness of SPS.

– *Deceiving Novice Users*: Phishers may deceive novice users into changing their proxy settings on browsers. Even then, a transparent SPS proxy, mentioned in section 3.3, can prevent novice users from phishing attacks with evasion techniques.
– *Deceiving the Rule-Set Maintainers*: Phishers may try to deceive the maintainer of SPS on some ISPs into listing phishing sites as valid URLs. According to the deployment model, it is hard for phishers to register their phishing sites into the rule-set because of the double-check mechanism.
– *Rank Pollution*: Phishers may try to pollute rankings on several search engines. If rank pollution has occurred, users who find the URL of legitimate sites with search engines would be lead to phishing sites. SPS always sanitizes suspicious URLs, therefore, SPS can prevent novice users from phishing attacks with rank pollution.

Table 3. Client environments

Operating System	Browser
Windows XP Professional	Internet Explorer 6.0
Windows XP Professional	Mozilla Firefox 1.0.6
Mac OS X 10.4.2	Internet Explorer 5.2.3
Mac OS X 10.4.2	Safari 2.0
Sun OS 5.9	Mozilla 1.4

– *DNS Cache Poisoning*: Phishers may try to evade SPS by pharming hosts files or by pouring dirty DNS records. SPS can protect novice users from pharming hosts files on novice users, because SPS resolves by its own hosts files or DNS servers. The pharming on the SPS proxy server never occurs unless a phisher cracks the SPS proxy server.
  SPS, however, cannot prevent novice users if the cache records on DNS servers are spoofed. Although DNS cache poisoning is one of the problems on the Internet, it is a security issue of DNS, therefore, it is outside of the scope of this paper.

According to these discussions, the SPS is robust against evasion attacks except for DNS cache poisoning.

4 Evaluation

4.1 Functional verification

To verify that the SPS meets design requirements, we tried browsing the various emulated phishing sites in several client environments. We prepared 5 client environments (Table 3) and 20 emulated phishing sites. These emulated phishing sites are made from Web pages of legitimate enterprises: Banks, Brokerage, Online payments and Online shops. Notice that the emulated phishing sites built in the test-bed network, are not allowed to be persistently accessible.

As the results of verifications, SPS could identify the phishing sites, sanitized all suspicious part of each HTML content mentioned in section 3.3, and alerted its presence to test users in all environments in Table 3.

4.2 Measurement of processing overhead

In this section, we evaluate the processing overhead of SPS. The processing overhead is generated by the SPS, when the SPS checks URLs and sanitize the contents. Also, the processing overhead is an important index for deployment, because novice users are not satisfied as long as the processing overhead is too big. To evaluate processing overhead, we measure the time spent to download content in various cases, and compare the response speed with and without SPS.

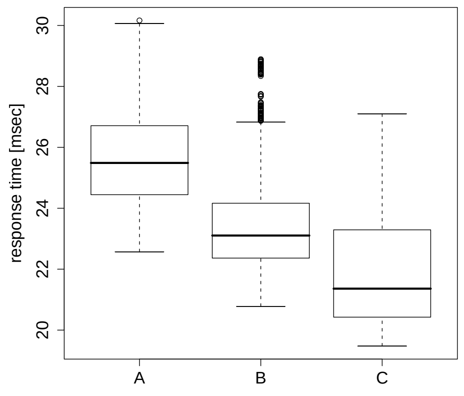


Fig. 5. Base case

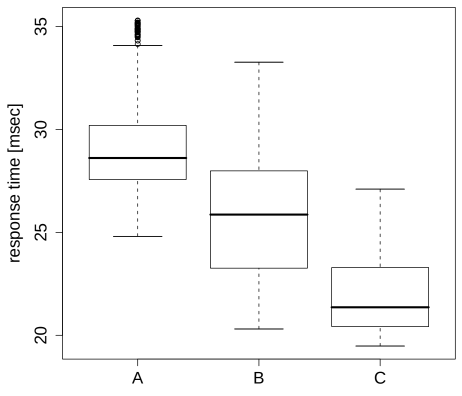


Fig. 6. Increment valid URL

Our test environment is comprised of a client host, server host, and SPS host. We prepare GNU Wget [30], a well-known HTTP client program, in the client host running Linux 2.4.22, the 1.7GHz Celeron processor and 256MB RAM. We also prepare Apache [31], another well-known HTTP server, in the server host running FreeBSD 4.11, the 800MHz Pentium III processor and 256MB RAM. The SPS host runs FreeBSD 4.11, the 1.8GHz Pentium4 processor and 512MB RAM.

First, we show the base case that there is only one valid URL and the content size is 10Kbytes. Next, we show the performance overhead of increasingly valid URLs and the content size. Moreover, we show the performance overhead of the content compression. Content compression is sometimes a useful HTTP extension to save bandwidth between a client and a server, but is not suitable for SPS, because SPS must decompress transactions to sanitize contents.

Base case

Figure 5 describes the base case of processing overhead. 1% of top and 1% of bottom data are omitted for accuracy. We assume that pattern A shows the processing overhead for phishing site, so that when the client uses SPS, a requested URL is invalid and sanitizing is also needed. Also, we assume that pattern B shows the processing overhead for the legitimate sites, so that the client uses SPS and the requested URL is valid. Pattern C shows that the client does not use SPS.

The processing overhead for the phishing site is 3.59 millisecond, which is obtained by subtracting from the average of pattern A to the average of pattern C. We consider the 3.59 millisecond of overhead not to be critical, because it is not sensible, and there are too many factors of long delays on the Internet. On the other hand, the processing overhead for the legitimate site is 1.45 millisecond, which is obtained by subtracting from the average of pattern B to the average

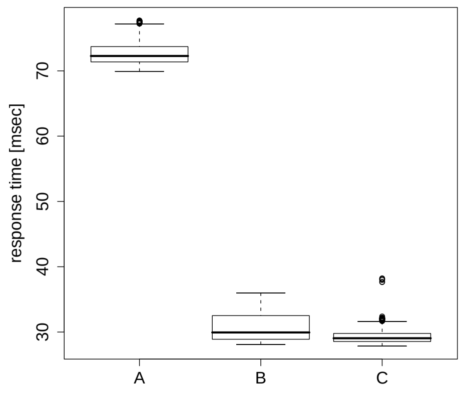


Fig. 7. Increment content size

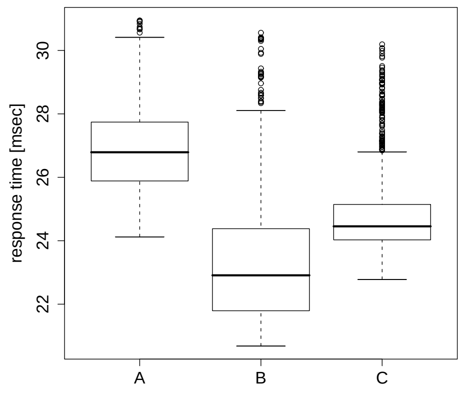


Fig. 8. Content compression

of pattern C. We may assume that the processing overhead can be reduced by increasing the legitimate sites, such that SPS does not need to sanitize.

The performance overhead of the increasing legitimate sites

To evaluate the performance overhead of the increasing legitimate sites, we change the number of valid URLs from 1 to 100. The result is described in Figure 6. Interestingly, the processing overhead is similar to the base case, even if the valid URLs are increased. In this case, the average of overhead for the phishing site is 6.99 millisecond and the average of overhead for the legitimate site is 3.90 millisecond. In summary, the processing overhead may not be affected sensitively by the increment of valid URLs.

The performance overhead of the increasing content size

In order to evaluate the performance overhead of the increasing content size, we change the content size from 10Kbytes to 100Kbytes. The result is described in Figure 7. The processing overhead for phishing is much higher than the base case, and the average of overhead for phishing site is 43.37 millisecond, otherwise the average of overhead for the legitimate site is 1.38 millisecond.

It can be assumed that SPS needs processing time to sanitize rather than validate the URL, and the overhead is affected sensitively by content size.

The performance overhead of content compression

To evaluate the performance overhead of the content compression, we use the HTTP contents encoding scheme [28]. The result is described in Figure 8. It is interesting to compare pattern B and C, because the processing overhead is reduced by the SPS. Otherwise the client decompresses the content by him or herself in pattern C, but in pattern B, the client does not need to decompress because the SPS sends the content to the client after decompression. Also, the processing overhead for the phishing site is not much higher than the base case, and the loss is only 0.43 millisecond. We may assume that the performance overhead of content compression is not high.

As a result, we conclude from the measurements that processing overhead for legitimate sites is low enough, even if the valid URLs are increased. The processing overhead for phishing sites is high when the contents are too big. This is not a problem because novice users never want to browse phishing sites quickly.

5 Conclusion

We contribute a simple filtering algorithm and an architecture based on a simple filtering algorithm as a robust countermeasure against Web phishing attacks. In this paper, we verified and evaluated our proposal, and showed the results of verifications and evaluations. Our proposed SPS is sufficiently robust and feasible on the Internet.

If SPS is deployed and the double-check mechanism works well, then, a sophisticated and accurate rule-set, that is, a list of verified legitimate sites will be produced. This list will be useful for rating Web sites that tell users which site is safe, trustworthy, and legitimate. Also, the result of such rating of legitimate sites will be available for constructing a countermeasure against Rank Pollution. This is one of future work on against phishing attacks.

References

1. Kumar, A.: Phishing - A new age weapon. Technical report, Open Web Application Securitry Project (OWASP) (2005)
2. Tally, G., Thomas, R., Vleck, T.V.: Anti-Phishing: Best Practices for Institutions and Consumers. Technical report, Anti-Phishng Working Group (2004)
3. Anti-Phishing Working Group: Phishing Activity Trends Report - June, 2005 (2005)
4. McCall, T., Moss, R.: Gartner Survey Shows Frequent Data Security Lapses and Increased Cyber Attacks Damage Consumer Trust in Online Commerce (2005)
5. Drake, C.E., Oliver, J.J., Koontz, E.J.: Anatomy of a Phishing Email. In: Proceedings of CEAS '04. (2004)
6. Felten, E.W., Balfanz, D., Dean, D., Wallach, D.S.: Web spoofing: An internet con game. In: Proceedings of NISSC '97. (1997)
7. Blue Coat Systems, Inc.: Spyware Prevention - Blue Coat Systems. Inc. - proxy servers (2005)
8. F5 Networks, Inc.: Trafficshield Application Firewall (1998-2005)
9. Dhamija, R., Tygar, J.: The Battle Against Phishng: Dynamic Security Skins. In: Proceedings of SOUPS 2005. (2005)
10. Dhamija, R., Tygar, J.: Phish and HIPs: Human Interactive Proofs to Detect Phishing Attacks. In: Proceedings of HIP '05. (2005)

11. Kirda, E., Kruegel, C.: Protecting Users Against Phishing Attacks with AntiPhish. In: Proceedings of 29th COMPSAC 2005. (2005)
12. Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D., Mitchell, J.C.: Client-side defense against web-based identity theft. In: Proceedings of NDSS '04. (2004)
13. Wenyin, L., Huang, G., Xiaoyue, L., Min, Z., Deng, X.: Detection of Phishing Webpages based on Visual Similarity. In: Proceedings of WWW 2005. (2005)
14. CoreStreet: SpoofStick (2005)
15. malwareremover: Phishing Sweeper (2005)
16. Herzberg, A., Gbara, A.: TrustBar: Protecting (even Naïve) Web Users from Spoofing and Phishing Attacks. Cryptology ePrint Archive, Report 2004/155 (2004)
17. Netcraft: Netcraft Toolbar (2005)
18. Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D., Mitchell, J.C.: SpoofGuard: Preventing online identity theft and phishing. Cryptology ePrint Archive, Report 2004/155 (2004)
19. Wu, M., Miller, R., Garfinkel, S.: Do Security Toolbars Actually Prevent Phishing Attack? In: Proceedings of SOUPS 2005, poster session. (2005)
20. Van der Merwe, A., Loock, M., Dabrowski, M.: Characteristics and Responsibilities Involved in a Phishing Attack. In: Proceedings of ISICT 2005. (2005)
21. Jakobsson, M.: Modeling and Preventing Phishing Attacks. In: Proceedings of FC '05, Pishing Panel. (2005)
22. Jakobsson, M.: Distributed Phishing Attacks. In: Proceedings of DIMACS Workshop on Theft in E-Commerce. (2005)
23. Anti-Phishing Working Group: Anti-Phishing Working Group: Resources (2005)
24. Adida, B., Hohenberger, S., Rivest, R.L.: Fighting Phishing Attacks: A Lightweight Trust Architecture for Detecting Spoofed Emails. In: Proceedings of DIMACS Workshop on Theft in E-Commerce. (2005)
25. Hohenberger, S.: Separable Identity-Based Ring Signatures: Theoretical Foundations For Fighting Phishing Attacks. In: Proceedings of DIMACS Workshop on Theft in E-Commerce. (2005)
26. Kaes, R.J., Young, S.: Tinyproxy (2004)
27. Privoxy Developers: Privoxy (2001-2004)
28. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force (1999)
29. Elson, J., Cepra, A.: Internet Content Adaptation Protocol(ICAP). RFC 3507, Internet Engineering Task Force (2003)
30. Free Software Foundation, Inc.: GNU Wget (2005)
31. The Apache Software Foundation: The Apache Software Foundation (1999-2005)