An efficient Security Solution for Dealing with Shortened URL Analysis

Jaime Devesa[1], Xabier Cantero[1], Gonzalo Alvarez[2] and Pablo G. Bringas[1]

[1]S[3]Lab, University of Deusto, Bilbao, Spain

[2]Grupo de Criptología y Seguridad de la Información, CSIC, Madrid, Spain

Abstract. With the boom of the Internet, and particularly of social networks, information sharing possibilities have increased. In this context, the so called URL shortening services, consisting of compacting a web link into a much shorter and manageable one, have arisen. However, the popularity of Web 2.0 also causes users to be unprotected against certain types of unwanted contents and attacks motivated by the desire of economic profit, which translates as an exponential increase in security incidents. Moreover, URL shortening services provide attackers a new method of obfuscation to malicious web links, hindering the analysis and detection of unwanted sites. Thus, we propose here a solution to solve the real destination of a shortened URL, analysing it in terms of security.

1 Introduction

Web address shortening is a technique based on making a website accessible through an URL much shorter than the original, so it is easier to remember. The shortened URLs have become an extremely useful method for recommending links, especially through social networks, mobile phones or microblogging platforms with space constraints, as in the case of Twitter.

The technical ease to develop and implement new shortening services, along with the emergence in 2006 of Twitter, have caused a boom in this kind of services. URL shortening has become the reference technique for recommending text links where HTML code cannot be inserted, and recently it is common to find these links in posts, blogs, documents, forums, etc.

Against this background, we present a study of the internal operation of URL shortening services. Second, we review the current state of security regarding these services. Furthermore, we propose a solution to address the aforementioned security risks. Next, we evaluate it with real malicious URLs. Finally, we draw conclusions about our technical solution and discuss the future work.

2 Study of the Internal Operation

Although not new, URL redirection is currently in vogue due to the different services to shorten links: extensive directions and particularly difficult to remember URLs become

Devesa J., Cantero X., Alvarez G. and G. Bringas P..
An efficient Security Solution for Dealing with Shortened URL Analysis.
DOI: 10.5220/0003579800700079
In *Proceedings of the 8th International Workshop on Security in Information Systems* (WOSIS-2011), pages 70-79
ISBN: 978-989-8425-61-4

shorter and therefore much easier to type, with a positive impact on the publicity and access to online services. There are different techniques to achieve this URL redirection, either based on the HTTP protocol or on the interpretation by web browsers of the downloaded page code.

The simplest system uses redirection HTTP status codes 30X. Defined in the Hypertext Transfer Protocol [1], a redirect consist of a response to a specific request by the client, whose code begins with the digit 3, prompting the browser to be directed to a new address indicated in the Location header of the HTTP message.

The protocol defines different response codes: 300, which indicates multiple destinations, such as when a page is available in several languages; 301, when the destination has changed permanently; 302, when the destination exists but must be accessed temporarily by another URL; 303, to specify that this URL is to be accessed via the HTTP GET method; or 307, similar to 302, but with other technical constraints.

We have conducted a study on HTTP response codes offered by various URL shortening services available on the market, based on a total of 242 different services operating at the time of analysis. This study gave the following results, shown in Table 1: 44% of services answered with HTTP code 301 (moved permanently), 39% did so with code 302 (moved temporarily), 1% used code 307, and finally, the remaining 16% returned HTTP code 200, that is to say, they used a different redirection system. Moreover, we have perceived that it is common to use various chained redirections.

Table 1. HTTP responses distribution from a corpus of 242 different shortened URLs, where 84% answered with 30X redirection codes.

HTTP response	Number of responses	Percentage
200	38	16%
301	108	44%
302	94	39%
307	2	1%
TOTAL	242	100%

As reflected in this study, the vast majority, 84% of shortening services, made use of redirection based on HTTP 30X status codes. Still, this percentage is higher if the analysis focuses only on the 10 most used services based on the Alexa ranking [2] as can be seen in Table 2.

However, within the HTTP redirection, the choice of different codes is relevant to the creator of the destination page. The crawling engines used by search engines understand redirection codes differently. If they get a 301 code, it will be understood that the link is the destination, so the page rank [3] is updated correctly. However, when a code 302 or 307 is obtained, which is supposed to be temporary redirection, the robot gives the page rank to the shortened link and not to the final destination, as really should occur. Therefore, when creating a URL shortening service, it is considered bad practice to use any redirect code other than 301. Leading services in the sector, such as *tinyurl.com* or *bit.ly*, properly use redirection code 301 (moved permanently), although the study shows that in nearly 40% of cases this practice is violated.

Table 2. HTTP methods used by Alexa Top 10 URL shortening services.

Shortening Service	Alexa Ranking	HTTP method used	Redirections
bit.ly	164	301	1
tinyurl.com	779	301	2
goo.gl	1.387	301	1
ow.ly	2.752	301	1
su.pr	7.467	301	1
is.gd	9.593	301/302	2
tiny.cc	10.400	301	1
j.mp	11.989	301	1
shorturl.com	16.408	301	1
cli.gs	30.877	200	0

There are other techniques to achieve the redirection of links, used by 20% of the shortening services analysed in the study. These systems are based on the use of *iframes* to insert the target page, the label 'META REFRESH' in 'HEAD' section of the website, or via *javascript*. Thus, these methods are based on HTTP 200 OK status code, which is a standard response for successful HTTP requests. Therefore, these practices are also discouraged, as the page rank is given again to the shortening service and not the destination page. However, these redirection techniques are often used by the creators of URL shortening services, allowing them to easily add on-line advertising to obtain a direct economic return from the service they offer.

Finally, there are also some specific cases of URL shortening services that do not automatically redirect, and the user has to click on the link destination to visit that page. The aim is to present the user an intermediate page full of banners and on-line advertising, although there are others that use these pages to show the user what will the final destination be, offering some relevant information about the content of this page via a content analysis, a snapshot of the website, etc, and leaving the choice of visiting it or not to the user. Once again, these services rely on HTTP 200 OK status code.

3 Security Implications

The dark side of the use of URL shortening services, and the most obvious, is that the user is no longer able to see directly where the browser will be redirected to. Moreover, because URL shortening is frequently used with social networks like Facebook, there is an inherent trust that the link will be legitimate, so users are going to click without taking into account their own security, and may be tricked into visiting malicious web sites. For instance, in 2010, 8 percent of 25 million URLs posted on Twitter pointed to phishing, malware or scam sites listed on popular blacklists [4].

Furthermore, most URL shortening domains are trusted by firewalls, Web filters or spam blocking tools [5], and other analysis solutions fail to detect the real URL's destination. This makes it difficult to identify links that lead to malicious destinations. Hence, it is easier for attackers to distribute shortened links which could redirect users to web sites containing the following security risks:

– Malware, trojans and other malicious programs.

- Code exploiting security vulnerabilities in browsers or systems.
- Web-based attacks.
- Phishing attempts.
- Spam campaigns.

Malware represents a high-priority issue to security researchers and poses a major threat to the privacy of computer users and their information. Former malware writers sought fame, but nowadays their goal has evolved into a notable financial gain. Therefore, malware has become a profitable illegal business and causes great economic losses all over the world [6]. Moreover, the amount, power, and variety of malicious software increases every year as well as its ability to avoid all kinds of security barriers [7]. Indeed, URL shortening services are intensively used by these cyber-criminals to distribute malicious programs [8], as they provide an obfuscation layer, making detection more difficult.

For instance, a new Twitter worm has recently spread fast, compromising user accounts and abusing Google's *goo.gl* link shortening service with the aim of distributing malicious links [9]. Thus, clicking on the shortened URL will take the user to a site that advertises a rogue antivirus solution, which is in fact malicious software.

Moreover, client-side exploits take advantage of vulnerabilities in client software, such as web browsers or media players, allowing cybercriminals to take control of computers visiting a malicious site [10]. In combination with URL shorteners, these attacks gain in sophistication, making thousands of users vulnerable. Furthermore, other web-based attacks such as Cross-Site scripting or SQL injection also benefit from using these shortening services to go unnoticed.

Furthermore, phishing is one of the attack vectors that causes more losses to financial institutions, using legitimate-looking but fake web sites and emails to deceive users into disclosing personal information [11]. According to [12], 0.47% of bank customers fall victim to phishing attacks each year, translating to $2.4M-$9.4M in annual fraud losses per one million clients. Likewise, there is evidence that phishers have been exploiting and abusing URL shortening services actively at least since 2006 [13], camouflaging web addresses which are identified on blacklists, and thus avoiding security filtering solutions.

Besides, spam (unwanted email messages which usually contain commercial contents) made up between 80 and 90 percent of all messages in 2010 [14]. Furthermore, the usage of URL shorteners in spam campaigns is also well known, used to avoid detection of the real URL by intent analysing techniques [15]. First campaigns employed popular URL shorteners like *bit.ly*, although the trend now is to use less known services which do not even have a way to report abuses through their web site. Moreover, spammers choose shortening services offering public APIs, which makes it even easier to integrate them into a botnet. Finally, services providing preview modes are also avoided, unlike the ones offering statistics, which can help spammers measuring certain aspects of the spam campaign they manage.

Therefore, URL shortening services have included some checks based on blacklists to prevent spammers and hackers from using their service, although these barriers have already been circumvented by using legitimate intermediate sites with redirection capabilities, and hence prevent blacklisting or blocking of the shortened link [16]. This tech-

nique is based on taking advantage of bad coding practices on the site's warning page for all links to external sites (i.e. *http://good.com/redirect?url=http://external.com/*). Thus, this page can be used to redirect users to any domain, including malicious pages (i.e. *http://good.com/redirect?url=http://evil.com/*). Creating a short link of this legitimate domain URL will avoid blacklists and redirect the victim to the spam site.

To conclude, we have identified other security implications regarding the shortening service itself: hijacking or compromise of the shortening service, and Deny-of-Service or interruption of service.

First, if the shortening service is compromised, an attacker could redirect all the existing links to a malicious site. That happened in June 2009 [17], when a hacker exploited a security hole in *cli.gs* shortening service's web page, allowing her to edit about 2.2 million URLs and point them to a harmless blog. Moreover, if it had been used to distribute malware exploiting a bug on web browsers, it would have been devastating. In conclusion, having control of so many URLs makes these services a very attractive target, allowing the *bad guys* to make a lot of money.

Second, when a URL shortening service crashes, due to either technical problems or because it is under a Deny-of-Service attack, it causes thousands of short-links posted on Twitter and other social network sites to be unavailable. For instance, on 2nd February 2011, the popular URL shortening service *is.gd* was unavailable for a few hours, effectively breaking thousands of shortened links [18]. Besides, two years ago a Spanish ISP blocked *TinyURL*'s IPs and domains for a week, meaning their clients could not access to thousands of links [19].

4 Countermeasures

Presently, for the purpose of obtaining greater transparency, many URL shortening services have included a preview functionality, so users are aware of the real destination they are going to be redirected to and determine if it is safe enough to visit. Although a lot of shorteners already offer this as an option, in most of them it is not the default. For instance, *TinyURL*'s *http://tinyurl.com/xxx* short link can be previewed by using *http://preview.tinyurl.com/xxx*, and in *BudURL* simply adding "?" at the end of the URL.

In addition, various external web services have emerged with the aim of displaying the real URL. This is the case of *LongURL* [20], which is also available as a browser plug-in. In fact, the problem of these solutions is that they depend on a known list of URL shortening services. Thus, the appearance of a new shortener or even a downtime in the service, makes them useless. Similarly, there are other web pages like *www.expandmyurl.com* providing URL unshortening, although they only work for certain services (*tinyurl.com*, *bit.ly* and *is.gd* in this case).

Moreover, many URL shortening services have become aware of the dangers of malicious URLs to their users' security. Thus, services like *mcaf.ee*, *saf.li*, *safe.mn* or *sameurl.com*, offer a secure shortening service by checking URLs for malware, phishing, spam or even some types of attacks like Cross-site scripting. Therefore, hackers are going to avoid shortening their malicious links with these services, using others with fewer security measures, and hence, increasing the effectiveness of the attack.

Thus, the default behaviour of all URL shortening services should be to display the full URL and ask if the user is sure to be redirected there. Unfortunately, this info is not always enough to assess the risk of the target site.

5 Proposed Solution

In this Section, we present a solution for dealing with shortened URL analysis consisting of two steps: resolution of HTTP redirections and URL analysis.

5.1 Method for resolving HTTP redirections

As has been seen in Sect. 2, most of the URL shortening services, around 85 percent, employ HTTP 30X redirection methods. The main problem of these services is the abuse induced by cyber-criminals to distribute links containing malware, spam, phishing or other undesirable content. Hence, we present a solution to address malicious URL obfuscation using shorteners employing 30X redirections.


```
resolveURL(url)
    status ← getResponseStatus(url)
    case status
        do { 200 : return (url)
             301 :
             302 :
             307 :
                do { location ← getLocationHeader(url)
                     return (resolveURL(location))
             404 : return ('FinalPageDoesn'tExist')
             405 : return ('URLCan'tBeResolved')
             503 : return ('ServiceUnavailable') }
```

Fig. 1. Real URL destination solve algorithm pseudo-code.

We have developed an algorithm, shown in Fig. 1, based on performing HTTP HEAD requests recursively to determine where the redirection points to. According to the response code sent by the remote server, the appropriate action is decided. Thus, a 30X code means that there is another redirection, so the algorithm is repeated taking the location field content of the HTTP header returned by the server as the new URL. Moreover, if the response is a 200 code, there are not more redirections and the final destination has been solved.

Otherwise, if the status response is an error code (404, 503, ...), this issue will be reported. Nevertheless, if this error is a 405 status code, it implies that the remote server can not service HEAD requests. Therefore, in this case the redirection should be solved via GET requests, but previously analysing that URL for any kind of implicit attack (i.e. XSS).

5.2 URL Analysis Services

In order to evaluate the malicious intentions of a URL, we have developed an analyser making use of various open web services available on the Internet. Moreover, we have included our proposed solution for dealing with shortened URLs before analysing them, so we are actually able to classify the real destination and not the short link, obtaining more accurate results as will be shown.

First, VirusTotal (*http://www.virustotal.com/*) is a service developed by *Hispasec Sistemas* which analyses suspicious files and URLs enabling the identification of viruses, worms, trojans and other kinds of malicious content detected by 43 antivirus engines and 6 web analysis toolbars. It offers a public API, so it has been easily integrated in our system. The antivirus engine detection is given by a percentage, where we consider that the file is suspicious if it is above 0%. On the other hand, we evaluate the web analysis toolbars results individually, although we exclude *ParetoLogic* from the analyser as it always detects shortened URLs as malicious. Hence, we employ the following URL analysis tools from VirusTotal: *Firefox*, *G-Data*, *Google Safebrowsing*, *Opera*, *Phishtank*.

Second, WOT - Web Of Trust (*http://www.mywot.com/*) is a service that scores the reputation of a website based on the community opinions. Thus, it grants a percentage value in terms of trustworthiness, vendor reliability, privacy and child safety. Hence, we measure WOTs confidence as the average of these four terms, considering a positive malicious URL detection if the percentage is below 59 (considered as an 'Unsatisfactory' value by the vendor).

Moreover, all these analysis services fail to detect other kinds of web attacks. Therefore, as a proof of concept, we have implemented in our solution a SQL Injection and Cross-site scripting (XSS) attack detectors, based on Regular Expressions and explained in [21].

In summary, we have 9 services running in our analyser: VirusTotal antivirus engine, the aforementioned five URL analysis tools from VirusTotal, MyWOT reputation service, SQL Injection detector and XSS detector. Furthermore, we consider that a URL is malicious if at least one of these services detects it.

These services are deployed as a unique web service that can be easily accessed by using a plugin developed for Mozilla Firefox web-browser. Simply placing the pointer over a web link, the results of the analysis will be displayed telling the user if it is safe to visit it or not.

6 Evaluation

In order to evaluate the proposed solution, we created a malicious URL dataset detailed in Sect. 6.1. Moreover, we developed an URL analyser, explained in Sect. 5.2, conformed by various services available on the Web. Furthermore, we ran four different experiments explained in Sect. 6.2 in order to show the shortcomings of URL analysis solutions when dealing with shortened URLs. What is more, we proved that our solution is able to deal with this problem, being able to analyse the actual URL instead of the shortened one.

6.1 Dataset

For the following experiments, we created a dataset made up of 133 different malicious URLs, considered as already known threats. Table 3 details the different URL types, number of samples and sources of the whole dataset. Furthermore, we ran a script that makes an HTTP HEAD request to each URL to ensure that every URL is on-line.

Table 3. Malicious URL dataset composition.

Type	Number	Source
Malware	45	https://zeustracker.abuse.ch http://www.malwaredomains.com
Phishing	49	http://www.phishtank.com
Spam	19	http://www.spamcop.com
SQLi	20	Diverse Hacking Forums
XSS	20	Diverse Hacking Forums
TOTAL	153	-

Moreover, in order to evaluate our proposed solution, we chose different URL shortening services to be used in the experimentation phase. First, we selected the best positioned services on Alexa ranking [2], i.e. *bit.ly* and *tinyurl.com*. Furthermore, we also chose *is.gd* as it employs a combination of HTTP 301 and 302 methods, while it is widely used, as shown in Table 2. Finally, we employed the services presented in Sect. 4 offering some kind of security measures against malicious links, i.e. *saf.li* and *safe.mn*, although we exclude *mcaf.ee* since this service bans the IP address if it detects that malicious URLs are being shortened. We consider this a good practice, and although we can not measure it, the detection rate is quite high.

6.2 Experiments

We have conducted four experiments, the results of which are summarized in Table 4. To begin, the first experiment consisted of analysing all the *"long"* URLs in the dataset with the solution proposed in Sect. 5.2, measuring the detection rate of malicious links. In the second place, we evaluated the shortening services malicious URL detection rate at the time of creating the short link or accessing it, automating the task by using the offered APIs by each service. Moreover, the third experiment involved analysing with our solution the shortened URLs that have not been detected by the shortening services, but without resolving the real destination. To conclude, in the last run we analysed the same URLs used in experiment 3 employing the algorithm presented in Sect. 5 before performing the analysis, so the real URL is analysed instead of the shortened one.

In the first experiment, mostly all the malicious URLs in the dataset are detected by the analysis system. Instead, the detection rates in the second experiment are not as high as they should (in some cases even null), and moreover, all the services fail to detect web attacks (i.e. SQLi and XSS). Hence, there is evidence that shortening services must improve their security solutions to protect their users. Moreover, when analysing the shortened URLs the detection rates fall to 0, demonstrating that they are a

Table 4. Malicious URL detection rate in %, where s_1=*bit.ly*, s_2=*is.gd*, s_3=*tinyurl.com*, s_4=*saf.li*, s_5=*safe.mn*.

URL	Experiment 1	Experiment 2					Experiment 3					Experiment 4				
		s_1	s_2	s_3	s_4	s_5	s_1	s_2	s_3	s_4	s_5	s_1	s_2	s_3	s_4	s_5
Malware	100	0	13.3	0	2.2	28.8	0	0	0	0	0	100	100	100	100	100
Phishing	97.9	79.5	32.6	0	2	75.5	0	0	0	0	0	90	96.9	97.9	97.9	91.6
Spam	89.4	10.5	73.6	0	0	57.8	0	0	0	0	0	94.1	80	89.4	89.4	87.5
SQLi	100	0	0	0	0	0	0	0	0	0	0	100	100	100	100	100
XSS	100	0	0	0	0	0	0	0	0	0	0	100	100	100	100	100

good way to obfuscate malicious links. Finally, using our redirection algorithm before performing the analysis, throws the same results as in the first experiment, which probes our solution to be a good approach to deal with URL obfuscation using shortening services.

7 Conclusions & Future Work

We have developed an easily implementable solution to protect Internet users during navegation sessions by developing a web service able to assess the security risk of a shortened URL. This web service can be conveniently accessed through a browser plugin, with minimal disruption to web browsing. Experimental results confirm that our system largely exceeds the performance of other partial solutions offered to date.

On the other hand, there are still some limitations. First, we only work with 30X redirection methods, so our algorithm will fail to resolve the final destination if other ways of redirection are used. Second, we access the VirusTotal antivirus engine by consulting the downloaded file md5 hash. Therefore, files are not actually analysed, so it is not effective for unknown samples for VirusTotal service.

As future lines of work, we plan to deal with other redirection methods used by URL shortening services, like the use of *iframes* to insert the target page, the label 'META REFRESH' in 'HEAD' section of the website, or via *javascript*. Moreover, we also project to develop plugins for other web browsers, like *Internet Explorer* or *Google Chrome*, so our solution can be easily accessed by more users. Finally, we are going to incorporate more security analysis services to web service solution in order to improve the results, and therefore, users' security.

References

1. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol–HTTP/1.1 (1999)
2. Alexa Ranking: The web information company (2011) Online: http://www.alexa.com/.
3. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. (1998)
4. Grier, C., Thomas, K., Paxson, V., Zhang, M.: @ spam: the underground on 140 characters or less. In: Proceedings of the 17th ACM conference on Computer and communications security, ACM (2010) 27–37
5. Bradley, T.: URL Shortening Frenzy Comes with Security Risks (December 15th 2009) Online: http://www.pcworld.com/businesscenter/article/184677/url_shortening_frenzy_comes_with_security_risks.html.

6. Computer-Economics: 2007 Malware report: The Economic Impact of Malware (2008) Online: http://www.computereconomics.com/.
7. Kaspersky-Labs: Kaspersky Security Bulletin: Statistics 2010 (2011) Online: http://www.securelist.com/en/analysis/204792162/Kaspersky_Security_Bulletin_2010_Statistics_2010.
8. Fighter, S.: Cyber-criminals Exploiting Shortened URLs for Malware Distribution (April 17th 2010) Online: http://www.spamfighter.com/News-14219-Cyber-criminals-Exploiting-Shortened-URLs-for-Malware-Distribution.htm.
9. Secure List: New Twitter worm redirects to Fake AV (January 20th 2011) Online: http://www.securelist.com/en/blog/11136/New_Twitter_worm_redirects_to_Fake_AV.
10. Petkov, P.D.: Client-Side Security - One year later. Black Hat (2008)
11. Wu, M., Miller, R., Garfinkel, S.: Do security toolbars actually prevent phishing attacks? In: Proceedings of the SIGCHI conference on Human Factors in computing systems, ACM (2006) 601–610
12. Trusteer: Measuring the Effectiveness of In-the-Wild Phishing Attacks (December 2nd 2009) Online: http://www.trusteer.com/sites/default/files/Phishing-Statistics-Dec-2009-FIN.pdf.
13. McGrath, D., Gupta, M.: Behind phishing: an examination of phisher modi operandi. In: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, USENIX Association (2008) 1–8
14. Symantec: State of Spam and Phishing, a monthly report (February 2011) Online: http://www.symantec.com/content/en/us/enterprise/other_resources/b-state_of_spam_and_phishing_report_02-2011.en-us.pdf.
15. MXLab: Increase in usage of URL shorteners in spam campaigns (January 4th 2011) Online: http://blog.mxlab.eu/2011/01/04/increase-in-usage-of-url-shorteners-in-spam-campaigns/.
16. Computer Security articles: Unchecked redirection + URL shortener = Spam (February 8th 2011) Online: http://www.computersecurityarticles.info/security/unchecked-redirection-url-shortener-spam/.
17. Cligs Blog: Cligs Got Hacked (June 15th 2010) Online: http://blog.cli.gs/news/cligs-got-hacked-restoration-from-backup-started.
18. Netcraft: is.gd URL shortener suffers downtime (February 2nd 2011) Online: http://news.netcraft.com/archives/2011/02/02/is-gd-url-shortener-suffers-downtime.html.
19. Genbeta: Tinyurl blocked by Telefonica all this week (October 23rd 2008) Online: http://www.genbeta.com/web/tinyurl-bloqueado-por-telefonica.
20. LongURL: URL, LongBrowse with Confidence and Increased Security! (2011) Online: http://longurl.org/.
21. Mookhey, K., Burghate, N.: Detection of SQL injection and cross-site scripting attacks. Article from: http://www. securityfocus. com/infocus/1768 (2004)