Proposing a Secure URL Shortening Service by using Blackboard Architecture

Dr. Reem J. Ismail

reemaljanabi@yahoo.com

University of Technology - Computer Sciences Department

Abstract: *URLs can get long, unattractive, and break when sent via e-mail. To shorten an Internet address, we proposed a secure URL shortening service which will take a long Web address and create a shorter address that will not break in an e-mail posting by using the Blackboard architecture.*

Security is considered in the proposed service to prevent hacking because shortened links may stop working or requests can be redirected to advertising websites and/or websites containing malware. The Blackboard is updated continuously by KS and it holds all the data concerning the URL shortening by storing all the intermediate solutions until the final solution of the problem of URL shortening is found. The knowledge sources that are proposed in the Blackboard system include: (URL Validation, URL Statistics, and Secure URL shortening).

As a result the proposed secure URL shortening service will give the user more trust to use the service with security, availability, and confidentiality consideration as compared to the available used services.

Keywords: URL shortening, blackboard architecture, Secure URL shortening

1. Introduction

An organization that constantly uses very long or complex URL’s will often find that customers expect not to understand a URL provided to them in any electronic communication. Subsequently, the attacker can abuse this “trust” through a combination of social engineering and deliberately broken long or incorrect URL’s, to cause the customer to follow a shortened URL. This short URL would be supplied by a third-party site and is used to obfuscate the true destination [1].

With the advent of social networks and Twitter, on which messages are required to fit into 140 characters, compression of URLs becomes more and more crucial. In order to turn long URLs into shorter ones, so-called URL shortening services (USS) are offered by various entities. When users want to shorten a long URL, they submit the long URL to a shortening service that returns a short URL which typically does not exceed 30 characters. The users then include the short URL instead of the long URL e.g. in a Twitter, short message or e-mail [2].

Users requesting short URLs from a USS may be attacked; so the shortening service is hacked, two different threats emerge to users of that service: Shortened links may stop working and requests can be redirected to advertising websites and/or websites containing malware [2].

For example, a customer may be used to following complex links such as:

https://privatebanking.mybank.com/privatebanking/ebankver2/secure/customersupport.aspx?messageID=3324341&Sess=asp04&passwordvalidate=true&changepassword=true

The attacker could easily create a fake web site at a URL such as:

http://www.attackersite.com/fake/mybank/support and register it as: http://tinyurl.com/4outd

In this paper we propose a secure URL shortening service which will take a long Web address and create a shorter address that will not break in an e-mail posting by using the Blackboard architecture. As a result the proposed secure URL service will give the user more trust to use the service with security, availability, and confidently consideration.

2. Secure URL Related works

In *Character Frequency Analysis* the attackers who are searching for secret URLs are primarily interested in URLs which have just recently been shortened, because the destination URL of older shortened URLs might not exist anymore. For the attackers it is therefore interesting to know the structure of a target service's shortened URLs. If all the shortened URLs a service hands out in the last 24 hours start with the same letter, it might be a good idea to start enumerating URLs with this prefix [2].

All shortened URLs from the top 10 shortening services on Twitter found have the following structure:

http://<host>/<path>

The <host> part is the USS's host name and <path> is the string uniquely identifying the short URL [2].

In [1], the author proposes to use short web addresses to prevent using shortening services. By following a few simple best practices, organizations can easily strengthen the security of their environments against many of these attacks and make it much more difficult for an attacker to confuse customers or clients.

In [3], the author analyses implications on security and privacy that are caused by the use of URL shortening services. It is evaluated what risks exist for the privacy of users, the security of their machines and for the security of server machines involved. A number of experiments are conducted for empirically analyzing the identified risks.

In [4], the authors provide a first characterization on the usage of short URLs. Specifically, the goal is to examine the content short URLs point to, how they are published, their popularity and activity over time, as well as their potential impact on the performance of the web.

3. The Architecture of the Blackboard [5]

The blackboard model was first proposed by Newell in 1962 and later incorporated by Reddy and Erman into the Hearsay and Hearsay II projects, both of which dealt with the problems of speech recognition. Englemore and Morgan explain the blackboard model by analogy to the problem of a group of people solving a jigsaw puzzle:

"Imagine a room with a large blackboard and around it a group of people each holding over-size jigsaw pieces. We start with volunteers who put on the blackboard (assume it's sticky) their most "promising" pieces. Each member of the group looks at his pieces and sees if any of them fit into the pieces already on the blackboard. Those with the appropriate pieces go up to the blackboard and update the evolving solution. The new updates cause other pieces to fall into place, and other people go to the blackboard to add their pieces. It does not matter whether one person holds more pieces than another. The whole puzzle can be solved in complete silence; that is, there need be no direct communication among the group. Each person is self-activating, knowing when his pieces will contribute to the solution. No a priori established order exists for people to go up to the blackboard. The apparent cooperative behavior is mediated by the state of the solution on the blackboard. If one watches the task being performed, the solution is built incrementally (one piece at a time) and opportunistically (as an opportunity for adding a piece arises), as opposed to starting, say, systematically from the left top corner and trying each piece".

As Figure (1) indicates, the blackboard framework consists of three elements: a blackboard, multiple knowledge sources, and a controller that mediates among these knowledge sources. Notice how the following description describes the key abstractions identified from the problem space. According to Nii, "the purpose of the blackboard is to hold computational and solution-state data needed by and produced by the knowledge sources.

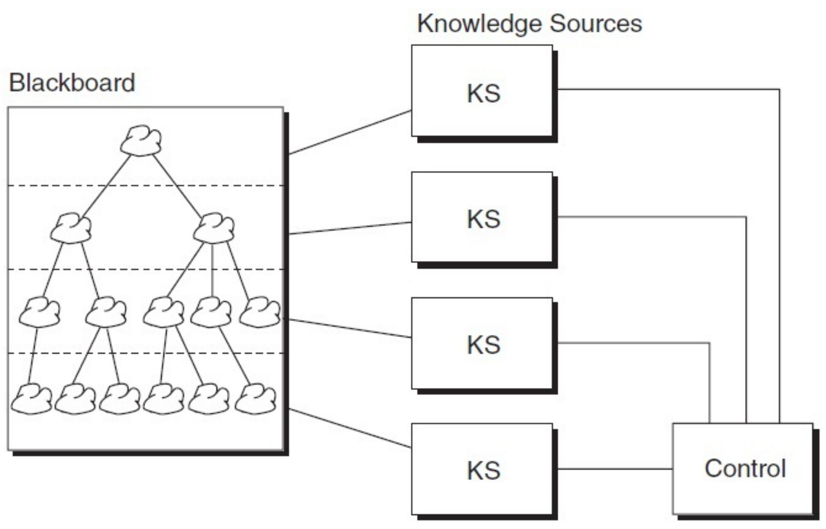


Figure (1): Blackboard Framework

The blackboard consists of objects from the solution space. The objects on the blackboard are hierarchically organized into levels of analysis. The objects and their properties define the vocabulary of the solution space".

As Englemore and Morgan explain, "The domain knowledge needed to solve a problem is partitioned into knowledge sources that are kept separate and independent. The objective of each knowledge source is to contribute information that will lead to a solution to the problem. A knowledge source takes a set of current information on the blackboard and updates it as encoded in its specialized knowledge. The knowledge sources are represented as procedures, sets of rules, or logic assertions".

Some of the main benefits of blackboard architectures and the systems built using a blackboard framework are as follows, [6]:

Flexibility of configuration. The knowledge sources within a blackboard system are not tied together in a fixed or rigidly applied manner. Instead they communicate and cooperate via the common blackboard. This means that any knowledge sources can be added to the system without having to specify its existence in any other knowledge source.

Flexible problem solving. A blackboard architecture is independent of any particular task and can therefore be used as the basis of very different applications, from route planning, to image processing, speech recognition and diagnosis. Cooperation is driven by the current state of the problem, rather than through any particular control strategy, and control of this cooperation is distributed between the knowledge sources themselves. These knowledge sources then determine which they are appropriate and the scheduler determines which of the appropriate knowledge sources to actually activate.

Selection of knowledge sources. Because more than one knowledge source may be able to perform the same function, the scheduler can select that problem solver which will provide the most benefit to the emerging solution. This can improve both problem solving efficiency and the quality of the eventual solution.

Multiple problem solvers. The blackboard architecture promotes the integration and multiple problem solving modules (knowledge sources). Each of these knowledge sources can potentially have its own representation and inference mechanism. This means that the twin advantages of multiple reasoning systems are maintained, namely: resilient behavior and efficient behavior.

Management of multiple levels of abstraction. The blackboard architecture explicitly supports the management of multiple levels of abstraction. Not only can the blackboard be divided hierarchically, but the knowledge sources can reflect this hierarchy. Problems of moving between the levels of the hierarchy can be overcome by providing knowledge sources whose sole task is to move information between two particular abstraction levels.

Opportunistic cooperation. Cooperation in a blackboard system is explicitly opportunistic; knowledge sources can post partial solutions to the blackboard in the hoped that some other knowledge source will be able to take these partial solutions and progress further down the path to the final solution.

4. The Proposed Secure URL Shortening Service

The components of the proposed secure URL shortening service that correspond to the architecture of the blackboard components are as shown in figure (2).

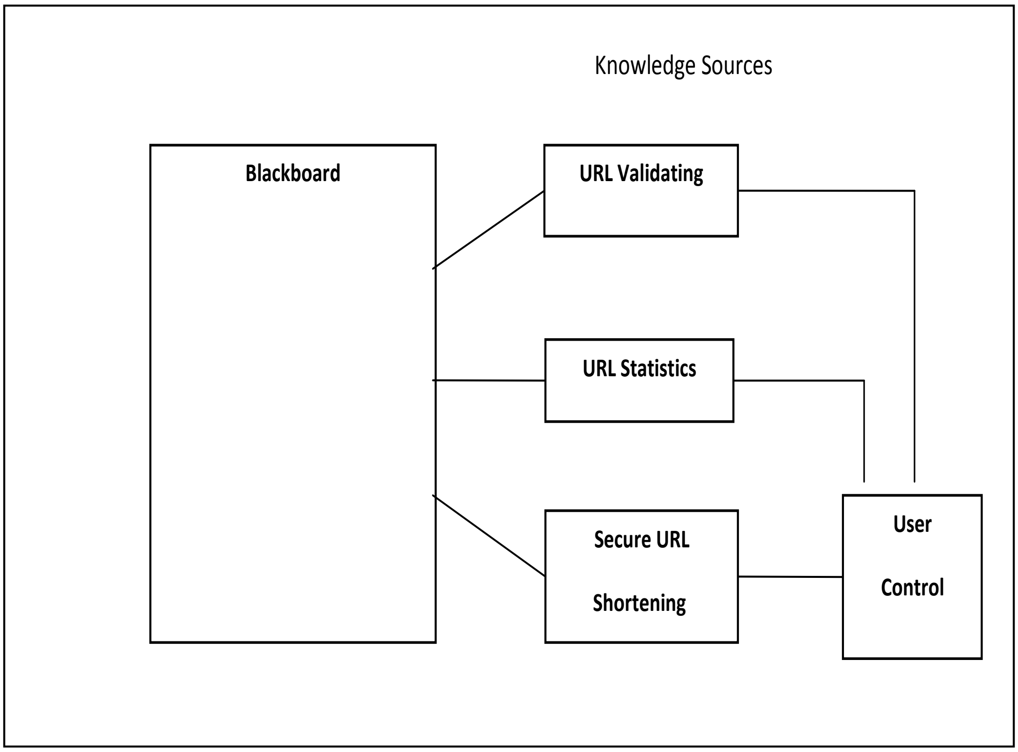


Figure (2): The Proposed URL-Blackboard Framework

4.1 Knowledge Sources (KS):

Each (KS) is represented as a procedure in the system and is called sequentially by the control, the output of the top KS is considered as an input to the next KS in the system, figure (2). Each KS gives a sub solution to the URL shortening problem to result the final solution; the following are the KS that are used in the proposed service:

1. **URL Validation:** This KS will check if the URL is long enough to be shortened or not; since not all URL need to be shortened, as described in algorithm (1). As in [2] short URL does not exceed 30 characters.

Algorithm (1): URL Validation
Input: URL
Output: Shortened URL or not

```
Begin
        Step1: Enter URL

        Step2: If  length(URL) > 30 Then

                        Go to Algorithm (2); URL need to be shortened

                Else

                        Exit; Use the same URL

                End If

End
```

2. **URL Statistics:** This KS will count how many links are there in the URL depending on separators "/ ", as described in algorithm (2).

Algorithm (2): URL Statistics
Input: URL, output of Algorithm (1)
Output: Number of words (*no_word*) in URL, vector (*index*) which has the start index for each word and vector (*word*) which has all words in URL

```
Begin
        Step1: no_word = 0

                For i = 1 To length (URL)

                        ch = Mid(URL, i, 1)

                        If ch = "/" Then
```

```
                    no_word = no_word + 1
                    index(no_word) = i
                End If
            Next i
        Step2: print "Number of paths: " & no_word
        Step3: For i = 1 To no_word
            words(i) = Mid(URL, index(i) , index(i + 1) - index(i))
            print "Word in URL " & word(i)
            Next i
End
```

3. **Secure URL Shortening:** This KS will produce a secure structure of URL shortening service, as described in algorithm (3). The user enters a secret key by which the secret shortening is done. The secret key is converted to ASCII code to use it as index for URL as explained in step (2), then some characters of the original URL are selected depending on the secret key index as explained in step (3). This key will give the user evidence that the shortened URL is not attacked by hackers so he can use it safely.

Algorithm (3): Secure URL shortening
Input: URL, secret key (*key*), output of Algorithm (2)
Output: Secure URL

```
Begin
        Step1: Enter secret key (key)
        Step2: Convert secret key to ASCII code to use it as index
for URL
```

```
          For i = 1 To Len(key)
                    ch = Asc((Mid(key, i, 1)))
                    index(i) = ch Mod 26
          Next i
Step 3: For i = 1 To Len(key)
                    find = Mid(URL, index(i), 1)
                    shorty = shorty & find
          Next i
          Print “secure URL is” & shorty
Step 4: Produce final short URL:
                    babyurl.com & word(1) & shorty
```

‘Note that word(1) will contain the name of the original web site which is important in our proposed structure

End

4.2 Blackboard:

The Blackboard is updated continuously by KS. The Blackboard holds all the data concerning the URL shortening by storing all the intermediate solutions until the final solution of the problem of URL shortening is found. The intermediate solutions are as shown in the text box of figure (3), which are: the original URL, all URL links, and final syntax of URL after shortening it.

4.3 User Control: The user will control and monitors the Blackboard’s states. The controlling is done by the user by executing each procedure of KS sequentially starting from **URL**

Validation then **URL Statistics** and finally **Secure URL Shortening** (the executing of the procedure is done by clicking the buttons that are shown in figure (3))**.** At each procedure executing the user monitors the output that is produced by it.

To study the weakness in the URL shortening services, we select the most popular services. Several URL shortening services have been found, e.g. bit.ly; tinyurl.com; etc. Table (1) shows four URL shortening services, they all are used as a shortening service to convert the same long URL which is:

URL Example:
http://search.conduit.com/results.aspx?q=hearsay+ii+A+speech+recognition+system+blackboard+&Suggest=&stype=Results&FollowOn=True&SelfSearch=1&SearchType=SearchWeb&SearchSource=32&ctid=CT2233703&octid=CT2233703

Table (1): Several URL shortening services that are tested in our study

URL Shortening Services	**URL Shortening**
bit.ly	http://bit.ly/Ufy0c2
tinyurl.com	http://tinyurl.com/8y5tsdx
goo.gl	http://goo.gl/J13Lb
mcafe.ee	http://mcaf.ee/93ojk

All the USSs that are listed in table (1) use the following structure, [2]:

http://< USS_name >/<random_path>

where:
< USS_name > is the URL shortening services
<random_path> is a random characters produced by the service
Instead, our proposed structure will be:

http://<USS_name>/<original_host>/<secure_path>

where:
<USS_name> represent the USS name, our proposed service name is “babyurl.com”.
<original_host> is the original site of URL that links to its original web page.
<secure_path> here the proposed service will use a secret key entered by the user to produce a path.

For example, the above URL will be shortened securely by using the secret key (***mybest***) in our proposed service as follows:

http:// babyurl.com /search.conduit.com/corudu

Here, the secret key is converted as an index to select some characters from the original URL as described in algorithm (3), so the shortened URL length is as the secret key length which is 5 characters as the example above shows. The user can change this secret key to be any word or phrase when using the service.

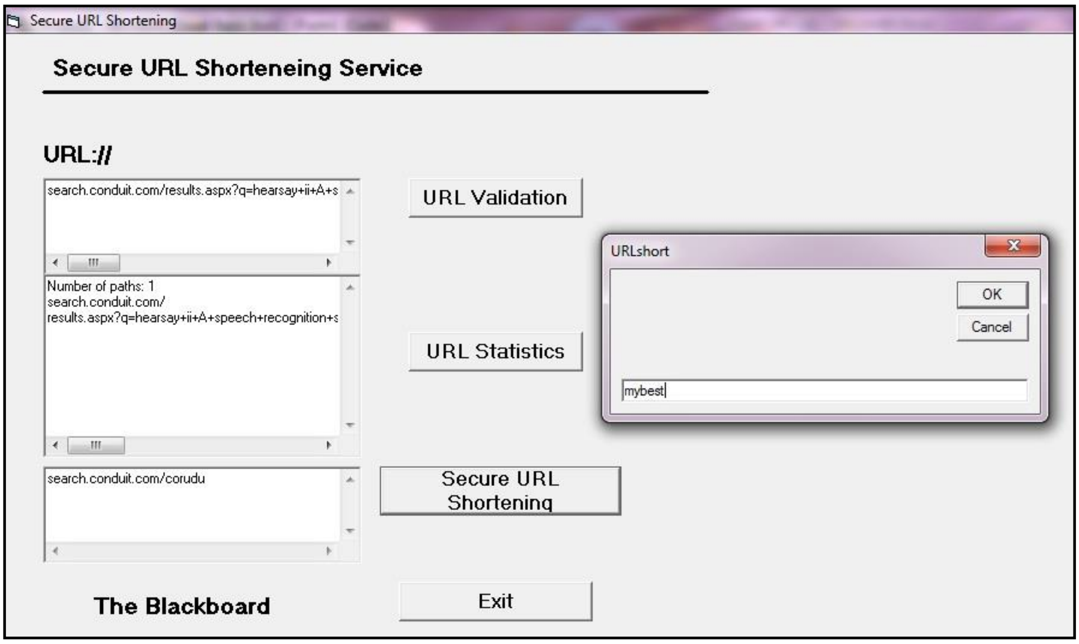
5. Results and Implementation of the Proposed Secure USS

As shown in Table (2), the proposed secure USS is compared with other URL shortening services with concern to the property of service like: URL shortening structure, Availability, Security, etc.

Table (2): Comparing between proposed USS service and others

Property of service	**URL shortening services**	**Proposed URL shortening service**
URL shortening structure	**http://< USS_name >/<random_path>**	**http://<USS_name>/<original_host>/ <secure_path>**
String length of URL after shortening	Each USS has fixed length size of URL after shortening which is between 5 to 7 characters	Variable character length because it depends on the secret key length that is entered from the user. This variable length will decrease the possibility of guessing the short URL by the attacker
Relation between old long URL and new short URL	Produces short random string that differs completely from the original long URL	Produces short random string that is selected from the original long URL
Availability	If short URL is attacked and changed the access to the original web site will not allowed because the shortening string is random	If short URL is attacked and changed the access to the original web site will still be allowed because the proposed URL shortening structure has <**original_host**> in it
Security	Have no security	Have security by using secret key which is known by the user to used it as index to produce **<secure_path>**
Shortening method of URL	It is unknown for the user of the service	It is known for the user of the service so he/she can control his final short URL by using his/ her own secret key

The proposed service is implemented by using visual basic programming language. Figure (3) shows the “Interface of the Proposed Secure URL Shortening Service”.



```
search.conduit.com/results.aspx?q=hearsay+ii+A+s
```

```
Number of paths: 1
search.conduit.com/
results.aspx?q=hearsay+ii+A+speech+recognition+s
```

```
search.conduit.com/corudu
```

```
mybest
```

Figure (3): The Interface of the Proposed

Secure URL Shortening Service

As shown in figure (3) the user will enter or easily copy and paste the original long URL and then press the "URL Validation", if the entered URL does exceed 30 characters then the user click "URL Statistics" and then "Secure URL Shortening" button, here the service will notify the user to enter the secret key to use it to produce the final secure URL shortening structure and then posting it in a short message or e-mail.

6. Conclusions

The proposed secure URL shortening service will give the user more trust to use the service with security, availability, and confidentiality consideration because the proposed URL shortening structure will have the following features:

1. Putting the original host name in the shorten URL will give the user more trust to use the service.
2. The availability is considered; when the shorten URL is attacked the user can still access the original site and its original web page.
3. Using <secure_path> instead of <random_path> will give more security by using secret key.
4. By using secret key the user can check the confidently of the shortened URL so he can use it safely.
5. The user will know how the shortening is done, in previous services the shortening is unknown and is done randomly.
6. The user can choose his/her own secret key and change it when needed.
7. The secret key is converted as an index to select characters from the original URL, so the shortened URL is obtained.
8. The available URL shortening services replace the entire URL with a new short URL which has no relation between the two. In our propose service we replace the entire URL by selecting some characters from the original URL.

References

1. Gunter Ollmann; "***Security Best Practice: Host Naming & URL Conventions***", Security considerations for web-based applications, Next Generation Security Software Ltd., 2005.
2. Alexander N., Johannes B., Ulrike M.; "***Security and Privacy Implications of URL Shortening Services***", IT Security Group, RWTH Aachen University, 2011.
3. Alexander Neumann ; "***Analyzing Security Implications of URL Shortening Services***", Diploma Thesis, RWTH Aachen University - Research Group IT-Security, 2011.
4. Demetris Antoniades, Iasonas Polakis and Georgios Kontaxis;" ***we.b: The web of short URLs***", Hyderabad, India March 28 – April 1, 2011.
5. Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen and Kelli A. Houston; "***Object Oriented Analysis and Design with Applications***" (on AI of cryptanalysis), 3rd edition, Addition- Wesley, 2007.
6. John Hunt; "***Blackboard Architectures***", JayDee Technology Ltd., 2002.

Note: the original proposed USS name is like the name of the (author name + short .com) but we use babyurl.com for paper evaluation only…

اقتراح خدمة سرية لاختصار عنوان صفحة الانترنت باستخدام هيكلية السبورة

م.د. ريم جعفر اسماعيل
reemaljanabi@yahoo.com
الجامعة التكنولوجية - قسم علوم الحاسوب

المستخلص:

قد يكون عنوان صفحة الانترنت طويل مما يؤدي الى قطعه عند استخدامه للارسال في البريد الالكتروني لذا فانه سوف يحتاج الى استخدام خدمة اختصار العنوان. لذا تم اقتراح خدمة سرية لاختصار عنوان صفحة الانترنت والتي تقوم بتحويل العنوان الطويل الى عنوان مختصر عن طريق استخدام هيكلية السبورة (blackboard architecture).

أن الخدمة المقترحة تؤمن سرية للخدمة وذلك بسبب الهجوم الذي يحدث لروابط صفحة الانترنت والذي يؤدي الى تعطيل الموقع الاصلي او تحويله الى موقع اخر مشكوك به. ان هيكلية السبورة المستخدمة سوف تحوي جميع المعلومات المتعلقة بخدمة الاختصار المقترحة عن طريق استخدام مجموعة من مصادر المعرفة (knowledge source) والتي تشمل: صلاحية العنوان و احصائيات و تقنيات سرية لتقصير عنوان موقع الانترنت.

(URL Validation, URL Statistics, and Secure URL shortening)

ان خدمة الاختصار المقترحة لعنوان صفحة الانترنت لها مميزات بان تكون ذات سرية اكثر ووثوقية ومتاحية للمستخدم مقارنة بالطرق المستخدمة الموجودة الاخرى.

الكلمات الرئيسية: تقصير عنوان موقع الانترنت و هيكل السبورة و تقنيات سرية لتقصير عنوان موقع الانترنت.