# Project Proposal: How to build a secure URL shortener?

Akshay Nehe
Stony Brook University
anehe@cs.stonybrook.edu

Vamshi Muthineni
Stony Brook University
vmuthineni@cs.stonybrook.edu

## 1 INTRODUCTION

URL Shorteners are increasing in use with rising social media usage and online services. Communication platforms like Twitter, SMS, etc, have message size limits. Long URLs are a problem in such scenarios. The shortened URLs replace long cumbersome URLs, improving readability and convenience. On the other hand, they may mask the underlying destination and guide a user to an unexpected destination webpage.

Not long back, researchers have shown that short URLs were susceptible to brute-force attacks [1] and ran the risk of exposing personal data of unsuspecting users. Since then, the majority of URL Shortening Services (referred as USS henceforth) have taken measures to protect against such attacks. We will review top 3 (by usage) USS and the security measures they have implemented. We will also look at the design factors to be considered when implementing such a service. We will attempt to quantify the secureness of a service based on certain metrics. This review will be of help to someone who is uninitiated in the topic of URL Shorteners. Also, it will serve as a guide to someone who needs to implement such a system. Lastly, we will implement an experimental setup to analyze shortened URLs. This setup will likely be a value addition for the research community.

## 2 RELATED WORK

There is ample research [7] [6] [5] on new ways to secure URL shorteners. Some past research and experiments about security and privacy implications of USS are also available. However there is no consolidated review of current USS which will guide an uninitiated reader on how to go about implementing a USS. We present details on the basics of USS and how to build one yourself with optimal security features.

We do not see any work on quantifying the secureness of such services. We will attempt to extract metrics using an experimental setup and assign a secureness score to the services.

## 3 APPROACH

We will perform a thorough review of available literature, tracing the history of USS and noting the events which led to changes. Understanding the triggers for change will make it easy to understand design decisions. This bottom-up approach will also serve as a good guide for any new reader, exposing the bare basics of USS.

Realizing the potential dangers arising from linkrot, the URL-Team project was initiated by archive.org [8]. This will be our primary source of shortened URLs over a time range.

We will use Apache Spark to analyze URLs and extract data points. We may run the risk of encountering computation and storage limitations and decide to limit our analysis within some limited time bounds or limit the domain of other features. However our experimental setup will be generic enough to work, resources permitting, at full scale. We will, for the purpose of this work, perform the experiments on our local hardware (think typical Macbook Pro) and, time permitting, transition to AWS compute nodes.

## 4 EXPERIMENTS

We will use Apache Spark to parse through large volume (big data) of shortened URLs from the archives. At the outset, we expect to analyze the following:

(1) what kind of links are shortened the most (zips, forms, etc) ?
(2) are the documents shared using shortened URLs password-protected ?

## 5 EVALUATION METRICS

Our preliminary analysis suggests the following metrics can be instrumental in quantifying the secureness of a USS:

(1) length of shortened URL (domain of values);
(2) density of characters from domain of characters;
(3) obfuscation of shortened URL (qualitative feature);

In the past, USS have been proven to be susceptible to brute force attacks [1], including those generated by Google and Microsoft . One of the ways to solve this problem is by making the domain space of short URLs large enough to avoid brute-forcing but still short enough to remain viable. So we can measure the probability of susceptibility to being brute-forced and use that as a metric.

There are some qualitative features that can be considered as well. Some USS use a timeout on the short URLs, similar to OTP in SMS. Some USS tradeoff length for clarity. This reduces the chances of masking underlying URL and luring a user to a black-listed/unexpected website.

## REFERENCES

[1] Martin Georgiev, Vitaly Shmatikov. *Gone in Six Characters: Short URLs Considered Harmful for Cloud Services.* http://arxiv.org/abs/1604.02734
[2] Alexander Neumann, Johannes Barnickel, Ulrike Meyer. *Security and Privacy Implications of URL Shortening Services.* Journal of Advances in Computer Networks, Vol. 3, No. 3, September 2015.
[3] Sophie Le Page, Guy-Vincent Jourdan, Gregor V. Bochmann, Jason Flood, Iosif-Viorel Onut. *Using URL shorteners to compare phishing and malware attacks.* 2018 APWG Symposium on Electronic Crime Research (eCrime)
[4] Chris Dale. *The Secrets in URL Shortening Services.* https://www.sans.org/blog/the-secrets-in-url-shortening-services/
[5] Dr. Reem J. Ismail. *Proposing a Secure URL Shortening Service by using Blackboard Architecture* .
[6] Hyung-Jin Mun, Yongzhen Li. *Secure Short URL Generation Method that Recognizes Risk of Target URL.* Wireless Pers Commun (2017) 93:269–283.
[7] Simon Bell, Peter Komisarczuk. *Measuring the Effectiveness of Twitter's URL Shortener (t.co) at Protecting Users from Phishing and Malware Attacks.* ACSW '20: Proceedings of the Australasian Computer Science Week Multiconference.
[8] https://www.archiveteam.org/index.php/URLTeam