Project Proposal
Security of Systems and Networks
Exhaustive search on URL shorteners

Alexandros Stavroulakis, Xavier Torrent Gorjón, and Nikolaos Petros Triantafyllidis*

University of Amsterdam, System and Network Engineering (MSc)

November 14, 2014

1 Introduction

As web resources keep growing in complexity, so does the size of the URLs used to access specific web content. These URLs can contain certain parameters regarding user preferences, search queries or the structure of a website, which if exposed might lead to potential security issues. A new type of service has appeared on the Internet in the last few years, the URL shorteners (goo.gl, bitly.com, tinyurl.com, among others). These services provide a means to share these addresses on platforms with limited size (SMS,Twitter, etc.).
We believe that these services can potentially expose unaware users to security vulnerabilities. The problem posed is that these services not only build a very centralised and targeted database of shared URLs, but because of the short URL lengths an exhaustive search is very feasible. That means that even if these shortened URLs are shared through private channels, they can still be easily guessed through brute force attacks trying sequential combinations of characters to synthesise those shortened URLs and retrieve the actual URLs behind them. Although we believe this cannot be a targeted attack, a malicious user could still retrieve vast amounts data from many Internet users and detect possible attacks to be performed at a later stage.

*e-mail: Alexandros.Stavroulakis@os3.nl, Xavier.TorrentGorjon@os3.nl, Nikolaos.Triantafyllidis@os3.nl

2 Research Questions

The questions we aim to answer in this project are the following:

- What kind of data can be retrieved from such an attack and how much of it can be considered sensitive?
- How can we efficiently design targeted queries in order to spot security holes, which can be later exploited for targeted attacks?
- Is there some URL generation pattern that can lead to certain targeted attacks?

This project focuses on the most vulnerable side of computer security: the users themselves. Even though we will try to determine if there are vulnerabilities on the URL shorteners themselves, most of the information leakage will come from users unaware of this kind of attacks.
We believe that this misuse of the URL shorteners can lead to security issues of different magnitudes and repercussions. Starting from that assumption, we will build a system to iteratively run over the full set of shortened URLs to retrieve the original URLs. From this point we will study the mined data and determine if our original assumption is correct. Depending on the data we find, this could be presented as a study to request the websites providing those services to warn their users about these risks.

3 Approach

Bit.ly and goo.gl both expose public Web APIs that allow to programmatically shorten long URLs but also expand their short versions and retrieve the long form urls.[1][2] We will develop an API consumer that will try to exhaustively expand all possible short urls by calling the public APIs, limited by the rate limits set by these services and the project deadline.
Once enough data has been collected and we have an insight on the information that will be gathered we can start designing the patterns that we will try to mine from the dataset. Such patterns, that could probably lead us to sensitive data, would be for example full names of people, username and passwords communicated in the URL, important dates (such as birthdates), google map links that could lead to a person's location and other details that can reveal the identity of a user. Further more there could be even more details to be mined that could reveal the weak points of systems such as internal IP addresses, URL parameters that give access to confidential data or even server configurations that can be exploited to perform attacks.
We should clarify at this point that the analysis is to be performed solely on the URL strings without needing to look at the actual web resource.
The last part of our research will be to extract certain statistics from the data mining results and draw conclusions regarding the amount and type of sensitive data and probable security threats within the dataset. Software-wise, we will use

an appropriate web programming language (probably Python or Go based on how they perform) to build the data aggregating software. The services will be deployed on our assigned SNE servers.

4 Planning

The project officially starts on November 17th and its final presentation is on December 15th, resulting in four working weeks. The planning for these weeks is as follows:

Week	Task
Week 1	Background research, familiarisation with the shortening services
Week 1	Literature review
Week 2	Software development
Week 2	Bulk data aggregation
Week 3	Start designing the data mining techniques
Week 3	Data aggregation finishes
Week 4	Application of pattern search on full dataset
Week 4	Statistical analysis of results
Week 4	Report composition
Week 5 (Dec 15)	Presentation

5 Ethical implications

All the information we will try to retrieve is publicly available on the Internet. However, we might encounter sensitive information (such as user preferences, birth dates, network configurations, etc.) in the process. That is the reason we will have to develop our crawlers and storage system in a way that the information we gather will not be leaked.
Another aspect to be considered is the fact that certain services try to limit the number of hits they get from a certain IP address and may consider a big amount of traffic originating from our servers as an attack. Fortunately the two biggest url shortening services (bit.ly, goo.gl) expose web APIs through which one can retrieve the full sized versions of the URLs. The limits set by those services are 5000 concurrent [1] connections for bit.ly and 1000000 hits per day for goo.gl [2]. We believe that these rates will provide us with a big enough data set to conduct our research.

References

[1] Bit.ly Web API documentation, http://dev.bitly.com/api.html

[2] Goo.gl Web API documentation, https://developers.google.com/url-shortener/v1/getting_started