CleanURL: A Privacy Aware Link Shortener

Daniel Kim, Kevin Su, Andrew G. West, and Adam J. Aviv
Department of Computer and Information Science
University of Pennsylvania – Philadelphia, PA, USA
{dki, kevinsu, westand, aviv}@seas.upenn.edu

ABSTRACT
When URLs containing application parameters are posted in public settings privacy can be compromised if the arguments contain personal or tracking data. To this end we describe a privacy aware link shortening service that attempts to strip sensitive and non-essential parameters based on difference algorithms and human feedback. Our implementation, CleanURL, allows users to validate our automated logic and provides them with intuition about how these otherwise opaque arguments function. Finally, we apply CleanURL over a large Twitter URL corpus to measure the prevalence of such privacy leaks and further motivate our tool.

1. INTRODUCTION
Uniform resource locators (URLs) often contain functionality that allows data to be passed to web applications. For example, consider the following URL:
$$\text{http://www.example.com?key1=val1&key2=val2}$$
There, the key-value pairs which proceed the question mark are collectively called the query string. Such parameters are common in practice for purposes of referrer tracking, user identification, and storage of session data.
Privacy concerns can arise when a user posts a URL containing a query string in a public forum (e.g., a social network or micro-blogging platform) as parameters might contain/encode sensitive information. Moreover, since many posting environments are profile driven, a history of contributions could reveal considerable private user data.
When a parameter is interpreted by the server-side application it may or may not affect how the browser visually renders the page. Thus, privacy leaks occurring as the result of non-rendered arguments are avoidable given their purpose is orthogonal to that of link sharing, which is to share content. Certain users may be cognizant of this fact, but manual removal and inspection is an arduous task. More likely, most users are unaware of the threat. Thus, we aim to provide an intuitive and efficient means by which to visualize parameter output, educate about the potential dangers, and produce more secure links. This is realized as a privacy-aware link shortening service (CleanURL).
We begin by using visual and textual comparisons (i.e., diff algorithms) to determine whether or not a parameter’s inclusion affects rendering. Dynamic webpage content (e.g., advertisements) can blur such distinctions. Thus, we create an interface that allows end-users to visually confirm/correct our generated rankings regarding URL reduction (Sec. 2).
Over time, this feedback can be aggregated to refine our approach and increase usability by reducing service overhead. Moreover, we use our technique to conduct an unsupervised measurement study over a corpus of 1.6 million Twitter URLs. We found that half of the URLs have arguments, and of those, 63% contain non-rendered arguments; a cause for privacy concern (Sec. 3).

2. LINK SHORTENING SERVICE
In describing our link shortener, we begin by using diff algorithms to estimate parameter necessity (Sec. 2.1) before integrating these into a user-facing interface (Sec. 2.2).

2.1 Estimating Argument Necessity
Given two URLs (i.e., one inclusive and another exclusive of some parameter) it is our goal to determine whether that parameter is a rendered one (i.e., induces non-random content change). We consider two approaches:

- **Visual diff**: The two URLs are rendered as down-scaled bitmaps with a standardized viewport and the Hamming distance between the images is calculated.
- **HTML tag diff**: The HTML source of the two URLs is parsed to remove visible text content. Over the remaining content (i.e., HTML tags) a standard textual diff is applied (based on longest common subsequence) and the size of the delta/patch is computed.

While visual difference is perhaps the most intuitive comparison method, one must also accommodate dynamic content. Even when an identical URL is fetched multiple times in succession the pages may render differently. Often, this comes in the form of different advertisement images whose prominent sizing can skew visual diff calculation.

The HTML tag diff provides an additional perspective. While visually different when rendered, advertisement placement code is generally quite static in nature (perhaps only changing the image path, if anything).

Both diff algorithms return real-valued measures that can be used jointly to determine the relevance of a URL argument. The values are aggregated and compared against a threshold to force a binary decision of whether the versions are sufficiently similar. This threshold should be selected such that it is tolerant of dynamic content (see Sec. 3.1).
3. UTILIZING FEEDBACK LOOPS

Feedback from the interface allows refinement of automated techniques to minimize users’ time investment and heighten trust (Sec. 3.1). Confident in our system’s abilities, the approach can be employed autonomously over a large corpus of URLs to estimate the prevalence and properties of privacy leaks “in the wild” (Sec. 3.2).