November, 2014

Measuring Privacy Disclosures in URL Query Strings

Andrew G. West
Adam J. Aviv

Measuring Privacy Disclosures in URL Query Strings

Andrew G. West
Verisign Labs
Reston, VA, USA
awest@verisign.com

Adam J. Aviv
U.S. Naval Academy
Annapolis, MD, USA
aviv@usna.edu

***Abstract*—Publicly posted URLs may contain a wealth of information about the identities and activities of the users who share them. URLs often utilize query strings (*i.e.,* key-value pairs appended to the URL path) as a means to pass session parameters and form data. While often benign and necessary to render the web page, query strings sometimes contain tracking mechanisms, user names, email addresses, and other information that users may not wish to publicly reveal. In isolation this is not particularly problematic, but the growth of Web 2.0 platforms such as social networks and micro-blogging means URLs (often copy-pasted from web browsers) are increasingly being publicly broadcast.**

To study the privacy ramifications of URL sharing this paper presents a measurement study of 892 million user-submitted URLs, many disseminated in (semi-) public forums. Within the corpus we find a trove of personal information, including 1.7 million email addresses. In the most egregious examples, query strings contain plaintext usernames and passwords for administrative and sensitive accounts. Data leakage is identified via both key-driven and value-driven analysis using manual inspections and automatic detection logic. Additionally, we analyze the "click-through" rates of sensitive URLs, examine geographical and mobile behavior patterns, and measure the broader statistical properties of key/value pairs. Finally, we argue that this study motivates a "CleanURL" service that can "scrub" URLs of privacy violating content.

I. Introduction

Uniform resource locators (URLs) specify functionality that allow data to be passed to server-side web applications. For example, the following URL:

```
http://www.ex.com/path/to/content.php?key1=val1&key2=val2
```

contains a set of key-value pairs (or application parameters) which collectively are called the *query string*. Query strings are common. In our dataset, 55% of URLs have at least one key-value pair. Query strings are used for a variety of purposes when retrieving web content, but not all parameters may be necessary or desirable for the faithful retrieval of that content. Such strings may contain tracking metrics or more sensitive personal information about the client/end-user.

In isolation URL privacy is of minimal concern, at most susceptible to attacks involving physical over-the shoulder observation (so called "shoulder surfing" attacks). The problem is massively exacerbated when URLs are shared and published online. The URL and the sensitive data contained within becomes available to marketers, spammers harvesting contact information, and cyber-criminals with nefarious intentions. It comes as no surprise that a tremendous quantity of URLs end up on the public web, in no small part due to a Web 2.0 culture increasingly characterized by social networking and information sharing [1]. Moreover, since many posting environments are profile driven, a history of contributions could reveal considerable private user data [2]. Additionally, query strings may also be exposed in man-in-the-middle attacks unless HTTPS is used to encrypt server requests.

In this paper we argue that these impacts on user privacy from published URLs are significant and prevalent. To the best of our knowledge the privacy ramifications of URLs have not been previously analyzed in depth (Sec. II). Instead, we support our argument through a measurement study of over 892 million user-submitted URLs. Additionally, *we feel social platforms have been insufficient in curbing these leaks*, despite being intuitive locales for privacy preserving logic. To address this deficiency we propose a system that can automatically identify unnecessary key-value pairs in submitted URLs, producing sanitized URLs that still faithfully render the web document.

Our URL measurement study (Sec. III) yields 1.3 billion key-value pairs for analysis. We find over a quarter-billion plaintext key/value pairs involved in referral tracking, with more than 10 million pairs possibly revealing some form of demographic, identity-based, or geographical information. In extreme examples, user and password authentication credentials were found in plaintext. Given non-standardized naming conventions these quantities represent the lower bound on an issue of significant scale. For example, 2000+ unique keys have email address values, and our manual analysis only considers the most popular such keys. Methods for automatic key- and value-driven detection of disclosures are also discussed, leveraging entropy, expected formats, known distributions, and self-verifiability. Though these values were sparse in our corpus, we believe this approach is a viable means to detect some of the most acute leaks (*e.g.,* credit card numbers).

Metadata regarding the URLs and their contributors also proves valuable (Sec. IV). Using "click-through" data we show sensitive URLs are being shared and visited, although they tend to incur below-average traffic. While we observe minor differences in the prevalence of problematic URLs originating from different geographical locales, we find mobile devices form an unusually large portion of the problem space.

Finally, the social platforms via which URLs are often published offer an opportunity to analyze and "scrub" URLs of potentially privacy violating content. Towards this, we propose a prototype service called "CleanURL" (Sec. V). The service will analyze URLs for query string parameters that have no affect on web content, and warn users when privacy revealing elements are not superfluous.

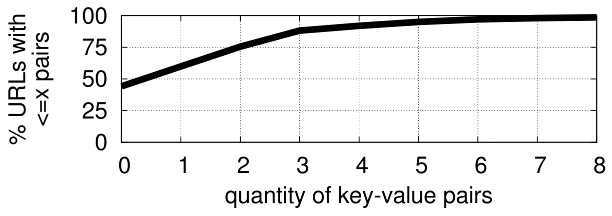

Fig. 1: CDF for quantity of key-value pairs in URLs

II. Related Work

The privacy concerns surrounding URLs and query strings have not been extensively studied in the literature. However, the broader security considerations of URL transformation services and argument passing have been examined. Link shortening services have been an area of focus, as their obfuscation of URLs has enabled phishing and other abuses [3], [4], [5]. Others have produced specifications for secure cross-organizational parameter passing [6] and described the intentional manipulation of key-value pairs [7].

Link shortening services resemble our CleanURL proposal in that they also transform input URLs. Such shorteners place a single redirection alias between a short link and its full representation. While convenient for presentation purposes and length constrained settings [1], shorteners do not sanitize links. Instead, these services provide one-hop of obfuscation for plaintext URLs. While this aids privacy by superficially keeping query strings from public view, it also prevents the raw URL from being interpreted by human users. This combined with the ease of link generation make shorteners a catalyst in a variety of attacks [8], including spam, phishing, and DNS fast-fluxing [3], [4], [5], [9], [10]. Our proposed CleanURL service aims to strip sensitive keys/values from URLs in an irreversible fashion; it is at the user's discretion whether these "clean" URLs are subsequently shortened.

The proposed CleanURL service needs to determine the affect of individual query parameters on page rendering, even in the presence of dynamic content. Recent work in the Internet censorship domain describes the use of Merkle hash trees over page structure to compare and detect differences between web content obtained via different network routing paths [11]. Vi-DIFF [12] takes a more visual approach to understanding the content and structural evolution of webpages. Our prototype currently prefers simpler heuristics, but either of these proposals could be integrated as the system matures.

III. Finding and Measuring Sensitive URLs

To demonstrate the privacy concerns of query strings a large URL corpus is analyzed. That corpus and its basic properties are first described (Sec. III-A). Attention then turns to key-value pairs with privacy implications, found by manual inspection of keys (Sec. III-B) and programmatic analysis of values (Sec. III-C). Entropy is identified as a measure helpful in identifying disclosures and their scope (Sec. III-D), and finally, REST-ful URLs are discussed (Sec. III-E).

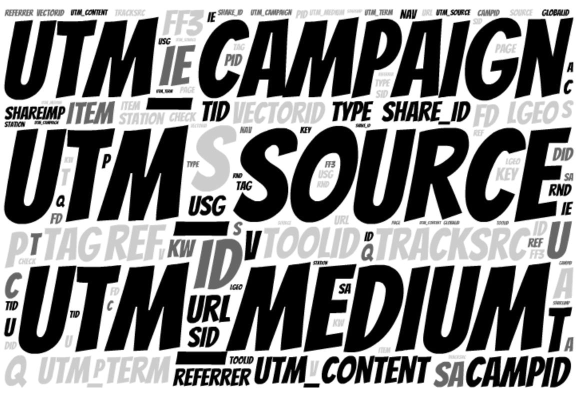

Fig. 2: Word cloud for common keys. Size indicates prevalence, with $\log_2$ applied to size weights for presentation

A. Data Source & Summary

URLs for analysis were obtained from an industry partner which has access to a large quantity of URLs submitted directly by end-users. The nature of this partner service is such that it eases link tracking and handling, meaning the vast majority of submitted links are later found on Web 2.0 social and collaborative services (see Sec. IV-A). Thus, sensitive query string information is likely to find itself in the (semi-) public domain where it may be harvested by peers, marketers, or cyber-criminals. Our URL set consists of 892 million URLs, 490 million (54.9%) of which have 1+ key-value pairs (Tab. I). Some 5% of URLs have greater than 5 pairs and 23.4k URLs had more than 100 (Fig. 1). There are roughly 909k unique key labels producing 1.3 billion total key-value pairs.

The word cloud of Fig. 2 visualizes the most common keys in the data. Leading the way is key `utm_source` with 128.5 million instances; in 14% of all addresses. That key – like 7 of the 10 most popular – and all those prefixed by "`utm`" are used to monitor referrers and traffic campaigns. The "Urchin tracking model" (UTM) is a structured means of link tracking that has become widespread due to its integration into the Google Analytics platform. In contrast, many keys are ambiguous in meaning and/or used by specific web platforms without an obvious naming convention. Single-letter keys are very common as are those that build around "`id`".

B. Key-driven Manual Analysis

The bulk of query strings are uninteresting, serving as opaque identifiers or benign session parameters. More interesting are those that reveal personal information about the identity, location, *etc.* of whomever visited and subsequently shared a URL. At this point we do not concern ourselves with whether these sensitive key-value pairs are intended or crucial to page rendering, only that they are present within the URL.

To find sensitive pairs we first manually inspect the 861 keys with 100k+ occurrences; interesting findings are grouped thematically in Tab. I. While key names are often indicative of use-cases, their values were also surveyed to confirm that sensitive data is present.[1] For example, we want to confirm that `zip` keys usually have 5-digit numerical values. With the exception of the "authentication" category, plaintext and

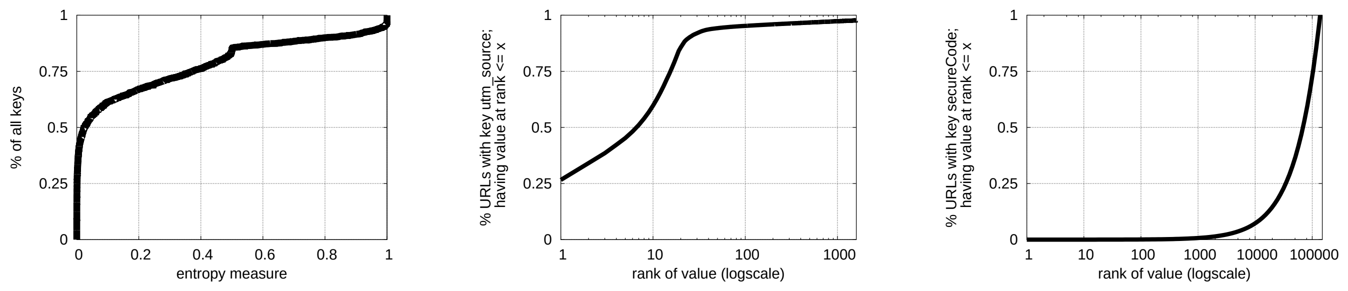

Fig. 3: CDFs describing key entropy and related measures: (a) Entropy distribution over all keys with 15,000+ instances; (b) Value prevalence for key `utm_source`; (c) Value prevalence for key `secureCode`.

THEME	KEYS	SUM-#
ALL URLS	—	892,934,790
URLS w/keys	*	490,227,789
referrer data	`utm_source, ref, tracksrc, referrer, source, src, sentFrom, referralSource, referral_source`	259,490,318
geo. location	`my_lat, my_lon, zip, country, coordinate, hours_offset, address`	5,961,565
network props.	`ul_speed, dl_speed, network_name, mobile`	3,824,398
online identity	`uname, user_email, email, user_id, user, login_account_id`	2,142,654
authentication	`login_password, pwd`	672,948
personal identity	`name1, name2, gender`	533,222
phone	`phone`	56,267

TABLE I: Keys w/100k+ occurrences having likely privacy ramifications, based on manual inspection[1]

human-readable values are the norm. Thus, Tab. I shows that a tremendous amount of personal information is potentially leaked via published URLs. In some ways this table *under reports* the risks. For example, the key "`email`" appears 103k times, but there are 637k pairs where the key matches the pattern `*email*`, and 1.7 *million* email addresses in the corpus based on a pattern match over values.

However, one must also be careful about such claims. There is no way of knowing to what extent values are "personal". Geographic coordinates in a URL could be referencing a user's exact location or, quite differently, be centering a map application over a landmark. We do not attempt to validate any of the mined personal information because of ethical considerations. This is particularly relevant when handling authentication credentials. Fortunately, when `password` or its analogues are present the values are *almost* always encrypted or hashed following best practices (*e.g.,* adding random "salting" values to limit hash-to-hash comparisons). Accordingly, the MD5 and SHA hashes of 100 common passwords matched no corpus values. Unfortunately, there are isolated examples (our shallow search found several dozen) of full credentials being passed in plain text via a query string. In the most egregious examples: (1) the credentials to an "admin" account were revealed, and (2) the user/password were for a site serving extremely personal information. These situations are "smoking gun" examples that would prove interesting to readers. Regrettably, the sensitivities surrounding these URLs are such that they cannot be published without significant redaction (*e.g.,* `https://www.⊙.com/index.aspx?accountname=⊙&username=⊙&password=⊙`).

C. Value-driven Autonomous Searching

A key-driven search for sensitive data has shortcomings. Manual efforts limit the depth to which labeling can be performed, and the process relies on hosts/applications adhering to non-standardized naming conventions. An alternate means of study is to analyze the values themselves for privacy leaks.

For example, credit card numbers are *self-verifiable* in that they have an expected length, established prefixes, and checksum via Luhn's algorithm [13]. Our search found 93,420 values that matched these criteria, but these were distributed across many key labels. We are confident these are opaque numerical identifiers; when aggregated by key (and subsequently by key *and* domain), no key's full value set had more criteria-matching values than probable over a set of random values. Other values have a *constrained and expected format*, such as dates of birth. Nearly 2 million such dates were identified, but it was ambiguous whether the dates were referencing personal information or an alternative data point (nearly all reside under the broad `date` key). Lastly, other values have an *expected distribution*. Value-first searching for 4-digit numerical values yielded keys `pinid`, `pno`, and `customid`, whose names are evocative of banking or confirmation PIN numbers. However, when the distributions of these numbers were plotted, they were entirely inconsistent with prior research into user-selected values of this type [14]. In this manner, value-first analysis allowed us to eliminate a potential privacy disclosure vector.

While not particularly fruitful over our corpus, we believe the value-first methodology is the preferred means to uncover these types of data points, *if they are present.*

[1]We do not contend that *every* value associated with these keys is privacy revealing. Instead we use Monte-Carlo sampling to confidently determine that at least a *majority* of values match an expected format.

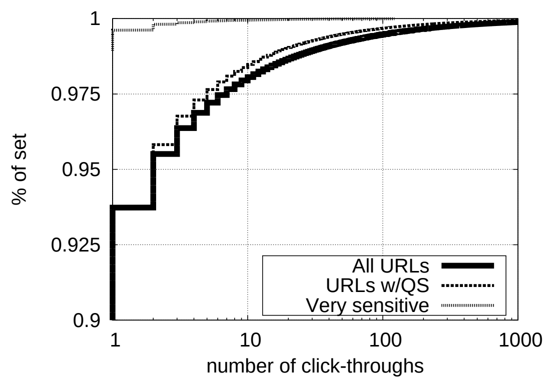

Fig. 4: CDF of click-throughs for all URLs, those with query strings (QS), and very "sensitive" ones (URLs containing keys in Tab. I, excluding those in the "referrer data" row)

COUNTRY	URLs	QUERY STR	SENSITIVE
US	63.88%	58.72%	0.31%
GB	14.40%	57.76%	0.22%
JP	9.44%	65.99%	0.11%
BR	6.23%	55.18%	0.11%
CA	6.19%	64.27%	0.36%
DE	5.96%	54.80%	0.34%
ES	5.28%	61.24%	0.19%
FR	4.77%	71.02%	0.29%
AU	3.54%	56.97%	0.24%
IN	3.50%	61.84%	0.25%
NL	3.27%	64.94%	0.31%
IT	2.78%	65.19%	0.25%
KR	2.77%	33.81%	4.74%
MX	2.49%	62.77%	0.24%
TR	2.35%	50.53%	0.14%

TABLE II: Geographic data on URL contributors. "Sensitive" URLs are those containing keys in Tab. I, excluding those in the "referrer data" row.

D. Value Entropy

Whereas the previous section data-mined values to find private data, our work has also discovered the higher-level *diversity* or *entropy* metric of a key's value set to be helpful. A key that is used in a binary fashion will have few unique values and low entropy. Even if this information were personal, *e.g.,* via the `gender` key, it does not reveal a terrible amount about the user in question (a random guess would often be correct). In contrast, high entropy keys have so many unique values that they can describe very specific properties towards identifying an individual. We compute a diversity (d) measure that lies on $(0, 1]$ by dividing the number of *unique* values in a keys's value list by the magnitude of that list, with Fig. 3a showing the distribution of d over popular keys. A majority of keys have very low entropy, including the most popular key, `utm_source`, whose distribution is plotted in Fig. 3b. Less than 10 unique values constitute a majority of that key's occurrences, led by values `twitterfeed` (self-explanatory; 26% of all values) and `share_petition` (from `change.org`; 7.5% of values).

Contrast that distribution (noting the differing log scales) with that of Fig. 3c, showing key `secureCode`, a key used for confirmation of account creations and mailing list subscriptions. Like `secureCode`, we find that most of the (interesting and privacy relevant) keys we identify in Tab. I lie on $0.33 < d < 0.66$. Examples of these keys include `user` (0.53), `email` (0.49), and `my_lat` + `my_lon` (both 0.38). That being said, many keys in this space do not appear to have privacy implications. Instead, the range contains a significantly reduced number of keys (per Fig. 3a) over which human analysts or complex computational methods can operate.

E. REST-ful URLs

Proponents of REST-ful URLs [15] believe that query strings decrease URL usability and accessibility. Instead, they advocate embedding parameters directly onto the URL path, thereby making the path a potential location for private data. We found that REST-ful URLs were not uncommon when dealing with host or application specific data (*e.g.,* product identifiers). However, as it pertains to client data, whose privacy we are concerned with, REST-ful URLs appear to be a very small portion of the problem space. Query string key `zip` has 270k appearances in our corpus, but searching for `*/zip/[5-digits]/*` in URL paths yielded just 252 results (0.093% as frequent). Similar results were found with other keys from Tab. I, justifying our choice to give REST-ful URLs no further attention moving forward.

IV. URL & Contributor Metadata

Our data source provides minimal information about those who contribute links, but does monitor user-agent and geolocation data for users who click those links. Measuring traffic to shared URLs is straightforward (Sec. IV-A). Moreover, in 5.3 million cases we are able to use an encrypted client identifier to join these data sets; precisely those cases where a user "tests" their contribution almost immediately after its creation. This enables discussion on the geographical patterns of contributors (Sec. IV-B) and the use of mobile devices (Sec. IV-C).

A. Traffic to Sensitive URLs

In the interest of efficiency, a single day's sample of contributed URLs was monitored for click traffic in the subsequent month. Around 12% of those 19 million URLs saw at least one click, and there were 103 million aggregate clicks in this interval. Many of our data partner's URLs are contributed in bulk by automated agents, not all of which are utilized in a timely manner, if at all. The lack of traffic to many links should not be a point of emphasis, it is a relative interpretation of Fig. 4 that is more significant. One could discard automated submissions using metadata supplied by our data provider, but we contend such contributions are an interesting portion of the URL sharing ecosystem.

Fig. 4 shows the traffic at links with a query string tracks closely with that of all URLs ($\approx$12% having 1+ views). Far less viewed are links that exhibit the most acute of privacy concerns (just 1.1% having 1+ views). This is not entirely surprising: these hosts/applications are ignoring best practices and this likely speaks to the quality and popularity of their entire operation. While this is seemingly a triumph for user privacy, realize that harvesters and criminals do not need to actually visit a link to obtain private data, they simply need to know of its existence.

B. Geographical Considerations

Tab. II provides a breakdown of contribution quantity and query string statistics by country. Query string presence shows statistically significant variance between nations of ±20% off the 60% mean. One anomaly is Korea's (KR) 4.7% rate of sensitive disclosures. Further investigation revealed this was due to a popular mobile messaging client that included a `user=` key; fortunately this parameter did not map to individual user names, but seemed to fulfill a more administrative function.

C. Role of Mobile Devices

Leveraging user-agent strings allows one to determine whether a URL contributor is using a mobile device. Recalling that our data joining methodology presumably excludes many automated clients, we found 22% of contributions were mobile in nature. In this set, 63% of mobile contributions have query strings compared to just 38% from non-mobile machines. This would seem to indicate either: (a) There is a fundamental difference in the content viewed and shared on mobile devices, or (b) non-mobile users are manually performing URL sanitization, a task that may be difficult for mobile users given small screen sizes, awkward keyboards, *etc.*. In fact, 40% of all "sensitive" URLs come from mobile devices, even though they compose just 22% of the broader set.

V. PRIVACY-ENHANCED URL SHARING

Discussion now shifts to "CleanURL", our proposal to reduce privacy disclosures via URL query strings. CleanURL consists of back-end logic to determine the (1) necessity and (2) sensitivity of key-value pairs (Sec. V-A). The output of the system is a sanitized URL that is graphically presented to end-users for confirmation or modification (Sec. V-B). See our technical report for more complete details.[2]

A. Argument Removal Logic

When attempting to sanitize a URL there are two properties of each key-value pair to assess: its *sensitivity*, whether the value contains private data – and its *necessity*, whether the content of the page renders correctly if the pair is removed. Programmatic methods for *necessity* sanitization logic are complex; we consider:

- VISUAL DIFF: The two URLs (pair inclusive and exclusive) 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 HTML tags a standard textual diff is applied and the delta size computed.

Both have proven moderately effective in preliminary testing. The primary complication is the presence of dynamic content (*e.g.,* advertisement images changing on every reload). In practice one needs to select or learn diff thresholds which can tolerate small amounts of dynamic noise and well represent the degree of change.

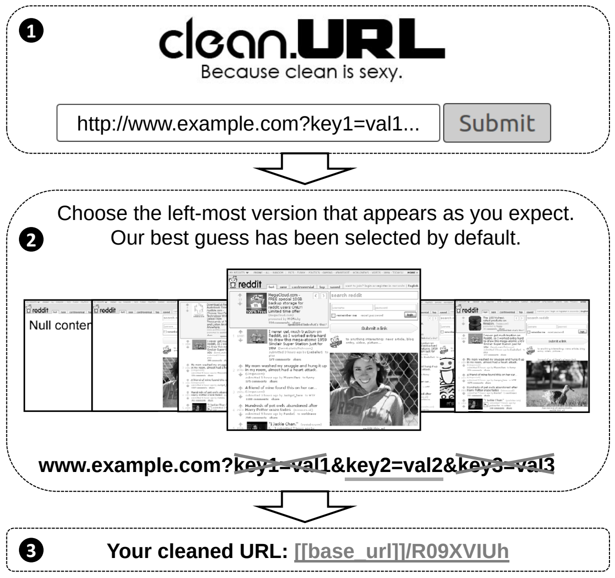
1
clean.URL
Because clean is sexy.
http://www.example.com?key1=val1...
Submit
2
Choose the left-most version that appears as you expect.
Our best guess has been selected by default.
Null conter
www.example.com?~~key1=val1~~&key2=val2&~~key3=val3~~
3
Your cleaned URL: [[base_url]]/R09XVIUh

Fig. 5: Simplified screenshots detailing CleanURL interface

With respect to determining pair *sensitivity* we rely on:

- Regular expressions gleaned from the naming patterns of known sensitive keys per Sec. III-B.
- Value-driven analysis based on self-verifiability, expected formats, and known distributions per Sec. III-C.
- Mining URL corpora with metrics like key entropy, which can indicate sensitive pairs per Sec. III-D.
- Human feedback loops about output correctness per the CleanURL GUI (see next subsection).

B. CleanURL User Interface

For prototyping purposes we wrap our sanitization logic as a stand-alone link shortening service. A typical session with the shortener is depicted in Fig. 5. A user begins by entering a URL in a simple form field. This sets off the computational removal logic. For each parameter combination the webpage source is downloaded, visually rendered, and input into our diff functions. Sensitivity properties are also investigated. Ultimately, combinations are sorted from most-to-least privacy preserving.

This ordering is the basis by which screenshots are presented in a "shuffle" selector to the end-user (see Fig. 5). The suggested version is selected by default: the combination that faithfully renders the page while removing the most sensitive parameters. If a sensitive parameter cannot be removed the end-user will be notified. Our design goal was to visualize the impact of URL manipulation and achieve end-user awareness while still maintaining a simple and usable interface. If our logic was too aggressive in removing parameters this interface allows the user to correct that error. It also provides an opportunity to better understand human factors, collect ground-truth, and analyze the sensitivities surrounding certain data.

[2]University of Pennsylvania Technical Report MS-CIS-12-12

VI. Conclusion

In conclusion, we have analyzed 892 million user-submitted URLs, many of which are destined for public broadcast. We found that many of these links have query strings with key-value pairs containing sensitive data ranging from tracking identifiers (`utm_*`) to contact details (`email`), location information (`my_lat`, `my_lon`), and even passwords. Entropy and value-driven searching were identified as programmatic ways to complement manual discovery of privacy disclosures. URL and contributor metadata such as link traffic, geographical origin, and mobile device participation provided further insight into the query string ecosystem. These results motivate our position that the privacy impacts of URL query strings should be analyzed further, especially given our prevalent Web 2.0 information sharing culture.

Motivated by these measurements, we proposed a system ("CleanURL") that addresses the privacy ramifications of URL query strings. Input URLs are programmatically analyzed to identify which key-value pairs lack *necessity* or are *sensitive*, towards the output of sanitized and privacy-safe URLs.

Acknowledgements: This journal version of the paper represents a significant extension to one appearing in *W2SP '14: Proceedings of the 8th Workshop on Web 2.0 Security and Privacy*. Daniel Kim and Kevin Su are thanked for their contributions on a preliminary version, *University of Pennsylvania Tech Report MS-CIS-12-12*. This research was partially supported by ONR grant N001614WX30023 and this material is based upon work supported by the Maryland Procurement Office under contract H98230-14-C-0127.

References

[1] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, and T. Karagiannis, "We.B: The web of short URLs," in *WWW '11: Proceedings of the 20th International Conference on World Wide Web*, Hyderabad, India, 2011, pp. 715–724.

[2] B. Krishnamurthy and C. Wills, "Privacy diffusion on the web: A longitudinal perspective," in *WWW '09: Proceedings of the 18th International Conference on World Wide Web*, Madrid, Spain, 2009, pp. 541–550.

[3] F. Maggi, A. Frossi, S. Zanero, G. Stringhini, B. Stone-Gross, C. Kruegel, and G. Vigna, "Two years of short URLs Internet measurement: Security threats and countermeasures," in *WWW '13: Proceedings of the 22nd International Conference on World Wide Web*, Rio de Janeiro, Brazil, 2013, pp. 861–872.

[4] S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru, "Phi.sh/$ocial: The phishing landscape through short URLs," in *CEAS '11: Proc. of the 8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, Redmond, WA, USA, 2011, pp. 92–101.

[5] S. Lee and J. Kim, "Fluxing botnet command and control channels with URL shortening services," *Computer Communications*, vol. 36, no. 3, pp. 320–332, February 2013.

[6] B. Pfitzmann and M. Waidner, "Privacy in browser-based attribute exchange," in *WPES '02: Proceedings of the ACM Workshop on Privacy in the Electronic Society*, Washington DC, USA, 2002, pp. 52–62.

[7] M. Balduzzi, C. Gimenez, D. Balzarotti, and E. Kirda, "Automated discovery of parameter pollution vulnerabilities in web applications," in *NDSS '11: Proceedings of the Network and Distributed Systems Security Symposium*, San Diego, CA, USA, 2011.

[8] D. Weiss, "The security implications of URL shortening services," April 2009, http://unweary.com/2009/04/the-security-implications-of-url-shortening-services.html.

[9] F. Klien and M. Strohmaier, "Short links under attack: Geographical analysis of spam in a URL shortener network," in *HT '12: Proceedings of the 23rd ACM Conference on Hypertext and Social Media*, Milwaukee, WI, USA, 2012, pp. 83–88.

[10] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: The underground on 140 characters or less," in *CCS '10: Proceedings of the 17th ACM Conference on Computer and Communications Security*, Chicago, IL, USA, 2010, pp. 27–37.

[11] J. Wilberding, A. Yates, M. Sherr, and W. Zhou, "Validating web content with Senser," in *ACSAC '13: Proceedings of the 29th Annual Computer Security Applications Conference*, New Orleans, LA, USA, December 2013, pp. 339–348.

[12] Z. Pehlivan, M. Ben-Saad, and S. Gançarski, "Vi-DIFF: Understanding web pages changes," in *DEXA '10: Proceedings of the 21st International Conference on Database and Expert Systems Applications*, Bilbao, Spain, 2010, pp. 1–15.

[13] H. Luhn, "Computer for verifying numbers," US Patent 2,950,048.

[14] J. Bonneau, S. Preibusch, and R. Anderson, "A birthday present every eleven wallets? The security of customer-chosen banking pins," in *FC 12: The 16th International Confernece on Financial Cryptography and Data Security*, Kralendijk, Bonaire, 2012, pp. 25–40.

[15] T. Berners-Lee, "Cool URIs don't change," Style Guide for Online Hypertext. W3C, 1998, http://www.w3.org/Provider/Style/URI.html.