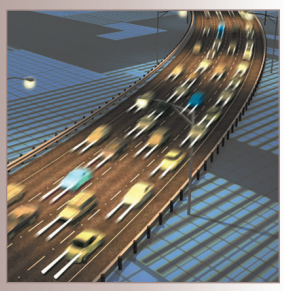

Measuring Privacy Disclosures in URL Query Strings

URLs sometimes contain sensitive information about the users who share them. To examine the associated privacy ramifications, a study looked at 892 million user-submitted URLs and found a trove of personal information, including 1.7 million email addresses and — in the most egregious examples — plaintext account credentials. The study examined data leakage, click-through rates, and various behavior patterns. The findings indicate the need for a mechanism to sanitize URLs of sensitive content.

Andrew G. West
Verisign Labs

Adam J. Aviv
US Naval Academy

URLs often utilize query strings – that is, key-value pairs appended to the URL path – to pass session parameters and form data. The URL http://www.ex.com/path/to/content.php?key1=val1&key2=val2, for example, has a query string with two key-value pairs. Although often benign and necessary to render the Web page, query strings sometimes contain tracking mechanisms, usernames, email addresses, and other information that users might not wish to publicly reveal.

In isolation, URL privacy is of minimal concern; at most, it leaves users susceptible to physical over-the-shoulder observation, or *shoulder surfing* attacks. However, the problem is massively exacerbated when URLs are shared and published online. The URL and the sensitive data contained within it then become available to marketers, spammers harvesting contact information, and cybercriminals with nefarious intentions. It comes as no surprise that many 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, because many posting environments are profile driven, a history of contributions could reveal considerable private user data.[2] Additionally, query strings can be exposed in man-in-the-middle attacks unless HTTPS is used to encrypt server requests.

We argue that published URLs have a significant and prevalent impact on user privacy, an issue that has yet to be studied in depth (see the "Related Work in URL Security" sidebar). As we describe here, our argument is supported by a measurement study of more than 892 million user-submitted URLs. We further contend that social platforms have been insufficient in curbing these leaks, despite being intuitive locales for privacy-preserving logic. To address this deficiency, we propose a

 Published by the IEEE Computer Society 1089-7801/14/$31.00

Related Work in URL Security

The privacy concerns surrounding URLs and query strings haven't been extensively reported on in the literature. However, researchers have examined the broader security considerations of URL transformation services and argument passing.

Link-shortening services have been one area of focus, given that their obfuscation of URLs has enabled phishing and other abuses.[1–3] Other researchers have produced specifications for secure cross-organizational parameter passing[4] and described the intentional manipulation of key-value pairs.[5]

Link-shortening services resemble our CleanURL proposal in that they also transform input URLs by placing a single redirection alias between a short link and its full representation. Although convenient for presentation purposes and length-constrained settings,[6] shorteners don't sanitize links. Instead, these services provide one hop of obfuscation for plaintext URLs. Although this aids privacy by superficially keeping query strings from public view, it also prevents human users from interpreting the raw URL. This, combined with the ease of link generation, make shorteners a catalyst in a variety of attacks,[7] including spam, phishing, and DNS fast-fluxing.[1–3,8,9]

Our proposed CleanURL service aims to strip sensitive keys and values from URLs in an irreversible fashion; it's at the user's discretion whether these "clean" URLs are subsequently shortened. The proposed CleanURL service must determine the effect 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.[10] Vi-DIFF[11] takes a more visual approach to understanding webpage content and structural evolution. Our prototype currently uses simpler heuristics, but either of these proposals could be integrated as the system matures.

References

1. F. Maggi et al., "Two Years of Short URLs Internet Measurement: Security Threats and Countermeasures," *Proc. 22th Int'l Conf. World Wide Web*, 2013, pp. 861–872.
2. S. Chhabra et al., "Phi.sh/$ocial: The Phishing Landscape through Short Urls," *Proc. 8th Ann. Collaboration, Electronic Messaging, Anti-Abuse, and Spam Conf.*, 2011, pp. 92–101.
3. S. Lee and J. Kim, "Fluxing Botnet Command and Control Channels with URL Shortening Services," *Computer Comm.*, vol. 36, no. 3, 2013, pp. 320–332.
4. B. Pfitzmann and M. Waidner, "Privacy in Browser-Based Attribute Exchange," *Proc. ACM Workshop Privacy in the Electronic Society*, 2002, pp. 52–62.
5. M. Balduzzi et al., "Automated Discovery of Parameter Pollution Vulnerabilities in Web Applications," *Proc. Network and Distributed Systems Security Symp.*, 2011; www.isoc.org/isoc/conferences/ndss/11/proceedings.shtml.
6. D. Antoniades et al., "We.B: the Web of Short Urls," *Proc. 20th Int'l Conf. World Wide Web*, 2011, pp. 715–724.
7. D. Weiss, "The Security Implications of URL Shortening Services," unweary.com, Apr. 2009; http://web.archive.org/web/20090405102930/http://unweary.com/2009/04/the-security-implications-of-url-shortening-services.html.
8. F. Klien and M. Strohmaier, "Short Links Under Attack: Geographical Analysis of Spam in a URL Shortener Network," *Proc. 23rd ACM Conf. Hypertext and Social Media*, 2012, pp. 83–88.
9. C. Grier et al., "@Spam: The Underground on 140 Characters or Less," *Proc. 17th ACM Conf. Computer and Comm. Security*, 2010, pp. 27–37.
10. J. Wilberding et al., "Validating Web Content with Senser," *Proc. 29th Ann. Computer Security Applications Conf.*, 2013, pp. 339–348.
11. Z. Pehlivan, M. Ben-Saad, and S. Gancarski, "Vi-DIFF: Understanding Web Pages Changes," *Proc. 21st Int'l Conf. Database and Expert Systems Applications*, 2010, pp. 1–15.

system that can automatically identify unnecessary key-value pairs in submitted URLs, producing sanitized URLs that still faithfully render the Web document.

Finding and Measuring Sensitive URLs

To demonstrate the privacy concerns of query strings, we obtained a large URL corpus from an industry partner with access to a large quantity of URLs submitted directly by users. Because this partner service eases link tracking and handling, many submitted links are later found on Web 2.0 social and collaborative services. Thus, sensitive query string information is likely to find itself in the semi-public domain where it can be harvested by peers, marketers, or cybercriminals.

Our URL set consists of 892 million URLs, 490 million (54.9 percent) of which have one or more key-value pairs (see Table 1). Approximately 5 percent of the URLs have more than five pairs, and more than 23,000 URLs have more than 100 (see Figure 1). In all, the set has roughly 909,000 unique key labels producing 1.3 billion total key-value pairs.

Figure 2's word cloud visualizes the most common keys in the data. Leading the way is the key `utm_source`, with 128.5 million instances in 14 percent of all addresses. The `utm_source` key is used to monitor referrers and traffic campaigns, as are seven of the 10 most popular keys and all of those prefixed by "`utm`" (for the *Urchin Tracking Model*, which offers a structured way to track links and has

*Table 1. Keys with privacy ramifications that have more than 100,000 occurrences**

Theme	Keys	Total occurrences
Total URLS URLS w/keys	—	892,934,790 490,227,789
Referrer data	`utm_source`, `ref`, `tracksrc`, `referrer`, `source`, `src`, `sentFrom`, `referralSource`, `referral_source`	259,490,318
Geographic location	`my_lat`, `my_lon`, `zip`, `country`, `coordinate`, `hours_offset`, `address`	5,961,565
Network properties	`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

*Not every value associated with these keys is privacy revealing; we use Monte Carlo sampling to confidently determine that at least a majority of values match an expected format.

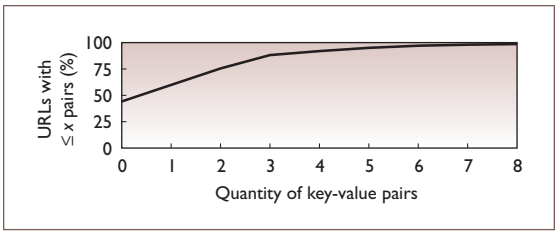

Figure 1. The cumulative distribution function for the quantity of key-value pairs in URLs. More than 54 percent of URLs have at least one such pair, with the long tail on this distribution indicating that some URLs handle significant quantities of data.

become widespread due to its integration into the Google Analytics platform). In contrast, many keys are ambiguous in meaning or used by specific Web platforms without an obvious naming convention. Single-letter keys are common, for example, as are those that build around the `id` key.

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, such as the identity and location of whomever visited and subsequently shared a URL. At this point, we don't concern ourselves with whether these sensitive key-value pairs are intended or crucial to page rendering, only that they're present within the URL.

To find sensitive pairs, we first manually inspect the 861 keys with more than 100,000 occurrences; Table 1 shows the interesting findings grouped thematically. Although key names often indicate use cases, we also surveyed their values to confirm that sensitive data is present. For example, we want to confirm that `zip` keys usually have five-digit numerical values. With the exception of the "authentication" category, plaintext and human-readable values are the norm. Thus, as Table 1 shows, considerable personal information is potentially leaked via published URLs. In some ways, this table underreports the risks. For example, the key `email` appears 103,000 times, but there are 637,000 pairs in which the key matches the pattern `*email*` and 1.7 million email addresses in the corpus based on a pattern match over values.

In any case, we must be careful about such claims because we can't know to what extent values are "personal." Geographic coordinates in a URL could be referencing a user's exact location or be centering a map application over a landmark. Given the ethical considerations, we don't attempt to validate any of the mined personal information. 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 (such as 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, isolated examples arise of full credentials being passed in plaintext via a query string (our shallow search found several dozen). In the two most egregious examples, the credentials to an `admin` account were

revealed, and the user/password were shown for a site serving extremely personal information. Such situations are interesting smoking-gun examples, but the sensitivities surrounding these URLs are such that they can't be published without significant redaction (such as https://www.▪.com/index.aspx?accountname=▪&username=▪&tpassword=▪).

Figure 2. Word cloud for common keys. Size indicates prevalence; we applied log2 to size weights for presentation.

Value-Driven Autonomous Searching

A key-driven search for sensitive data has shortcomings. Manual efforts limit the depth to which labeling can occur, and the process relies on hosts and applications adhering to nonstandardized 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.[3] Our search found 93,420 values that matched these criteria, but these were distributed across many key labels. We're 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. We identified nearly two million such dates, 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 four-digit numerical values yielded the keys `pinid`, `pno`, and `customid`, whose names are evocative of banking or confirmation PINs. However, when the distributions of these numbers were plotted, they were entirely inconsistent with prior research into user-selected values of this type.[4] In this manner, value-first analysis let us eliminate a potential privacy disclosure vector.

Although 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're present.

Value Entropy

In addition to data mining values to find private data, we found that the higher-level diversity or entropy metric of a key's value set was also helpful. A key that is used in a binary fashion will have few unique values and low entropy. Even if this information were personal, such as via the `gender` key, it doesn't 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 toward identifying an individual.

We compute a diversity (d) measure that lies on (0,1] by dividing the number of unique values in a key's value list by the magnitude of that list; Figure 3a shows d's distribution over popular keys. Most keys have low entropy, including the most popular key, `utm_source` (Figure 3b plots its distribution). Fewer than 10 unique values constitute a majority of that key's occurrences, led by values `twitterfeed` (self-explanatory; 26 percent of all values) and `share_petition` (from change.org; 7.5 percent of values).

Contrast Figure 3b's distribution (noting the differing log scales) with that of Figure 3c showing the key `secureCode`, which is used for confirming account creations and mailing list subscriptions. As with `secureCode`, we find that most of the (interesting and privacy-relevant) keys we identify in Table 1 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 don't appear to have privacy implications. Instead, the range contains a significantly reduced number of keys (per Figure 3a) over which human analysts or complex computational methods can operate.

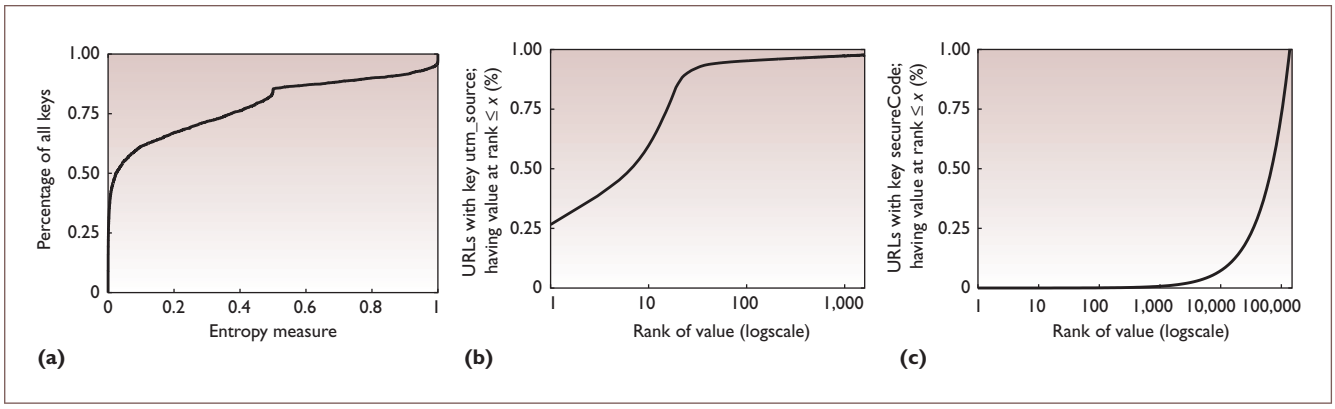

Figure 3. The cumulative distribution function describing key entropy and related measures. (a) Entropy distribution across all keys with 15,000 instances or more. (b) Value prevalence for the key `utm_source`*. (c) The value prevalence for the key* `secureCode`*.*

RESTful URLs

Proponents of RESTful URLs[5] believe that query strings decrease URL usability and accessibility. Instead, they advocate for embedding parameters directly onto the URL path, thereby making the path a potential location for private data. In our study, RESTful URLs weren't uncommon when dealing with host- or application-specific data (such as product identifiers). However, as it pertains to client data — whose privacy we're concerned with — RESTful URLs appear to be a minute portion of the problem space. The query string key `zip` has 270,000 appearances in our corpus, but searching for `*/zip/[5-digits]/*` in URL paths yielded just 252 results (0.093 percent as frequent). We found similar results with other keys from Table 1, justifying our choice to give RESTful URLs no further attention moving forward.

URL and 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. Although traffic quantification is still straightforward, our ability to learn about a contributor's location and device type is limited. However, we can gain insight from 5.3 million cases in which we use an encrypted client identifier to join the "contributor" and "click-through" datasets — which are precisely those cases in which users test their contribution soon after its creation.

Traffic to Sensitive URLs

In the interest of efficiency, we monitored a single day's sample of contributed URLs for click traffic in the subsequent month. Approximately 12 percent 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, and not all URLs are used in a timely manner, if at all. The lack of traffic to many links shouldn't be a point of emphasis; a relative interpretation of Figure 4 is more significant. We could have discarded automated submissions using metadata supplied by our data provider, but we contend such contributions are an interesting portion of the URL-sharing ecosystem.

Figure 4 shows that the traffic at links with a query string tracks closely with that of all URLs (approximately 12 percent having one or more views). Far less viewed are links that exhibit the most acute privacy concerns (with just 1.1 percent having one or more views). This isn't entirely surprising: these hosts/applications are ignoring best practices, which likely speaks to the quality and popularity of their entire operation. Although this is seemingly a triumph for user privacy, realize that harvesters and criminals don't need to actually visit a link to obtain private data — they simply need to know of its existence.

Geographical Considerations

Table 2 provides a breakdown of contribution quantity and query string statistics by country. Query string presence shows statistically significant variance between nations of plus or minus 20 percent off the 60 percent mean. One anomaly is Korea's 4.7 percent rate of sensitive disclosures.

Further investigation revealed this was due to a popular mobile messaging client that included a `user=` key; fortunately, this parameter didn't map to individual usernames, but seemed to fulfill a more administrative function.

Role of Mobile Devices

Leveraging user-agent strings lets us determine whether a URL contributor is using a mobile device. Recalling that our data-joining methodology presumably excludes many automated clients, we found 22 percent of contributions to be mobile in nature. In this set, 63 percent of mobile contributions have query strings, compared to just 38 percent from non-mobile machines. This seems to indicate that either

- mobile users view and share fundamentally different content, or
- non-mobile users are manually performing URL sanitization, a task that might be difficult for mobile users given small screen sizes, awkward keyboards, and so on.

In fact, 40 percent of all "sensitive" URLs come from mobile devices, even though they compose just 22 percent of the broader set.

Privacy-Enhanced URL Sharing

To reduce privacy disclosures via URL query strings, we propose CleanURL, a system that uses back-end logic to determine both the necessity and sensitivity of key-value pairs. The system's output is a sanitized URL that is graphically presented to users for confirmation or modification. For complete details, see our technical report.[6]

Argument Removal Logic

When attempting to sanitize a URL, we must first assess each key-value pair's sensitivity (whether the value contains private data) and its necessity (whether the page content renders correctly if the pair is removed). Programmatic methods for necessity sanitization logic are complex. We consider two:

- *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.

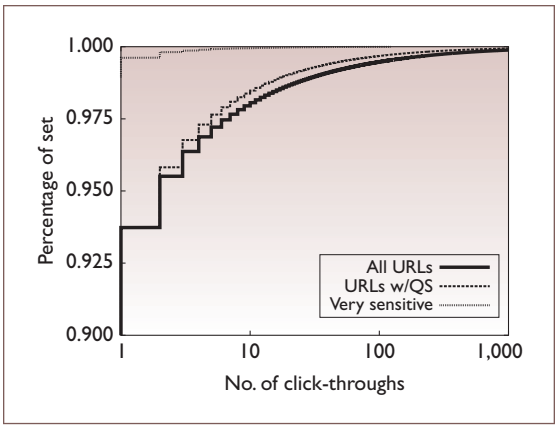

Figure 4. URL click-through rates. The graph shows the cumulative distribution function of click-throughs for all URLs, those with query strings (QS), and highly sensitive keys from Table 1 (excluding those in the "referrer data" row).

Table 2. Geographic data on URL contributors.

Country	URLs (%)	Query string (%)	Sensitive URLs* (%)
US	63.88	58.72	0.31
UK	14.40	57.76	0.22
Japan	9.44	65.99	0.11
Brazil	6.23	55.18	0.11
Canada	6.19	64.27	0.36
Germany	5.96	54.80	0.34
Spain	5.28	61.24	0.19
France	4.77	71.02	0.29
Australia	3.54	56.97	0.24
India	3.50	61.84	0.25
Netherlands	3.27	64.94	0.31
Italy	2.78	65.19	0.25
Korea	2.77	33.81	4.74
Mexico	2.49	62.77	0.24
Turkey	2.35	50.53	0.14

*"Sensitive" URLs are those containing keys in Table 1, excluding those in the "referrer data" row.

- *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 is computed.

Both methods have proven moderately effective in preliminary testing. The primary complication is the presence of dynamic content

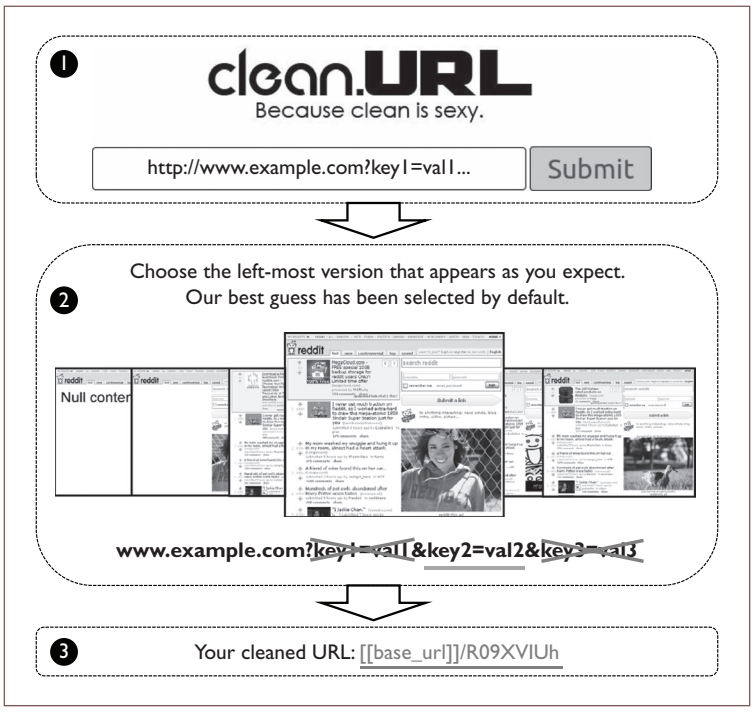

Figure 5. Simplified screenshots detailing the CleanURL interface. Although this figure shows a user-facing version of the software, the logic could be installed to transparently perform URL sanitization.

(such as when advertisement images change on every reload). In practice, it's necessary to select or learn diff thresholds that can tolerate small amounts of dynamic noise and effectively represent the degree of change.

To determine pair sensitivity, we rely on techniques that build on our earlier efforts to find such keys, including

- regular expressions gleaned from the naming patterns of known sensitive keys;
- value-driven analysis based on self-verifiability, expected formats, and known distributions; and
- mining URL corpora with metrics such as key entropy, which can indicate sensitive pairs.

We can also use human feedback loops as the basis for evaluating output correctness, as we now describe.

CleanURL User Interface

For prototyping purposes, we wrap our sanitization logic as a stand-alone link-shortening service. Figure 5 shows a typical session with the shortener. A user begins by entering a URL in a simple form field, which sets off the computational removal logic. For each parameter combination, the webpage source is downloaded, visually rendered, and input into our diff functions. We also investigate sensitivity properties. 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 user (see Figure 5). The suggested version is selected by default as the combination that faithfully renders the page while also removing the most sensitive parameters. If a sensitive parameter can't be removed, the user is notified.

Our design goal was to visualize the impact of URL manipulation and achieve user awareness while still maintaining a simple and usable interface. If our logic was too aggressive in removing parameters, this interface lets users correct that error. It also provides an opportunity to better understand human factors, collect ground truth, and analyze the sensitivities surrounding certain data.

Moving forward with this research, our primary goal is to complete the implementation of our CleanURL proposal. Recruiting a user base will enable usability studies and ground-truth for evaluating our "necessity" and "sensitivity" logic, which will facilitate development of a tool that can operate autonomously and transparently at high accuracy. With this, we can then survey industry partners in the URL-handling and user-generated content domains (including our data partner) about bringing this technology to bear on their platforms.

Acknowledgments

This article is a significant extension to the one that appeared in the *Proceedings of the 8th Workshop on Web 2.0 Security and Privacy* (W2SP 14). We thank Daniel Kim and Kevin Su for their contributions on a preliminary version.[6] ONR grant N001614WX30023 partially supported our research; this article is based on work supported by the Maryland Procurement Office under contract H98230-14-C-0127.

References

1. D. Antoniades et al., "We.B: the Web of Short Urls," *Proc. 20th Int'l Conf. World Wide Web*, 2011, pp. 715–724.

2. B. Krishnamurthy and C. Wills, "Privacy Diffusion on the Web: A Longitudinal Perspective," *Proc. 18th Int'l Conf. World Wide Web*, 2009, pp. 541–550.
3. H. Luhn, *Computer For Verifying Numbers*, US Patent 2,950,048, 1960.
4. J. Bonneau, S. Preibusch, and R. Anderson, "A Birthday Present Every Eleven Wallets? The Security of Customer-Chosen Banking Pins," *Proc. 16th Int'l Conf. Financial Cryptography and Data Security*, 2012, pp. 25–40.
5. T. Berners-Lee, "Cool URIs Don't Change," style guide for online hypertext, W3C, 1998, www.w3.org/provider/style/uri.html.
6. A.G. West and A.J. Aviv, *CleanURL: A Privacy Aware Link Shortener*, tech. report MS-CIS-12-12, Computer and Information Science, Univ. of Pennsylvania, 2012.

Andrew G. West is a research scientist at Verisign Labs. His research interests include Web 2.0 and network security, electronic anti-abuse, applied machine learning, trust/reputation management, and Internet measurement; his dissertation work investigated content security for collaborative information systems. West has a PhD in computer and information science from the University of Pennsylvania. Contact him at awest@verisign.com; www.andrew-g-west.com.

Adam J. Aviv is an assistant professor of computer science at the US Naval Academy. His research interests in computer security include system and network security, applied cryptography, smartphone security, and usable security with a focus on mobile devices. Aviv has a PhD in computer and information science from the University of Pennsylvania. Contact him at aviv@usna.edu; www.usna.edu/Users/cs/aviv.