Oleksii Starov, Phillipa Gill, and Nick Nikiforakis

Are You Sure You Want to Contact Us? Quantifying the Leakage of PII via Website Contact Forms

Abstract: The majority of commercial websites provide users the ability to contact them via dedicated contact pages. In these pages, users are typically requested to provide their names, email addresses, and reason for contacting the website. This effectively makes contact pages a gateway from being anonymous or pseudonymous, i.e., identified via stateful and stateless identifiers, to being eponymous. As such, the environment where users provide their personally identifiable information (PII) has to be trusted and free from intentional and unintentional information leaks. In this paper, we report on the first large-scale study of PII leakage via contact pages of the 100,000 most popular sites of the web. We develop a reliable methodology for identifying and interacting with contact forms as well as techniques that allow us to discover the leakage of PII towards third-parties, even when that information is obfuscated. Using these methods, we witness the leakage of PII towards third-parties in a wide range of ways, including the leakage through third-party form submissions, third-party scripts that collect PII information from a first-party page, and unintended leakage through a browser's Referer header. To recover the lost control of users over their PII, we design and develop FORMLOCK, a browser extension that warns the user when contact forms are using PII-leaking practices, and provides the ability to comprehensively lock-down a form so that a user's details cannot be, neither accidentally, nor intentionally, leaked to third parties.

Keywords: privacy, tracking, Personally Identifiable Information, HTTP Referer

DOI 10.1515/popets-2015-0028
Received 2015-04-15; revised 2015-07-15; accepted 2015-07-15.

Oleksii Starov: Department of Computer Science, Stony Brook University, E-mail: ostarov@cs.stonybrook.edu
Phillipa Gill: Department of Computer Science, Stony Brook University, E-mail: phillipa@cs.stonybrook.edu
Nick Nikiforakis: Department of Computer Science, Stony Brook University, E-mail: nick@cs.stonybrook.edu

1 Introduction

The rise of extremely popular online services offered at no fiscal cost to users has given rise to a rich online ecosystem of third party trackers and online advertisers. Indeed, this phenomenon has been studied extensively on generic websites [16], online social networks [6, 17], and in mobile applications [8, 13]. It has also raised the ire of online privacy advocates [38] who highlight the potential of tracking to compromise user privacy. Online service providers counter that tracking is based on non-personally identifiable information [27], and in the absence of user willingness to pay for online services, it seems unlikely this practice will cease any time soon, even though users find tracking unwanted and intrusive [40].

In this paper, we focus on third party tracking that occurs on the contact page of popular web pages. Tracking on contact pages is particularly concerning, as these are pages where users may enter in personally identifiable information (*e.g.,* contact information) to initiate contact with the given website. Unfortunately for users, as we show in this paper, their PII that is intended for the website they would like to contact, may be shared with third parties, even before they click on the "Submit" button on the form. Our work sheds light on an underexplored area in online tracking, online tracking that occurs on contact forms. We expose a rich ecosystem of third parties that primarily exist in the context of contact forms (*e.g.,* web leads companies, form builders) and are in a good position to obtain a user's PII.

Part of our goal in this paper is to understand third party tracking on contact pages at scale. This required surmounting obstacles not present in prior studies of online tracking. Specifically, we needed to automate interaction with the web page to (1) located the contact form, (2) fill in the contact form with the appropriate inputs, and (3) detect PII leakage from the contact form/the page embedding it. To surmount these challenges we design a custom crawler that is able to locate and fill out contact forms automatically. This crawler allows us to scale our study to the top 100,000 websites, as listed by Alexa. We also design a fuzzing methodology that allows us to identify PII leakage, even when the leaked information is encoded or obfuscated. Using this

methodology we study PII leakage to third parties on contact form pages along three axes:

- **What is the potential for leakage?** We consider the potential for leakage by quantifying differences in third-party JavaScript scripts included on contact form pages. We show that 29% of the sites with contact forms have more third-party JavaScript inclusions on their contact pages than on their main pages. Moreover, we categorize the actual leakage into two categories, *accidental* and *intentional*.
- **Which third parties obtain PII that is accidentally leaked by contact forms?** We discover that 2.5% of contact forms accidentally leak a user's PII to a total of 3,573 unique remote domains and we discuss the privacy ramifications of this accidental leakage.
- **Which third parties obtain PII that is intentionally leaked by contact forms?** In addition to accidental PII leakage, we show that 6.1% of all submitted contact forms are intentionally leaking a user's PII to third parties, mostly through the use of third-party form builders and marketing tools.

Motivated by our findings, we design and implement FORMLOCK, a browser extension that warns users about leaky forms and protects a user's PII against client-side leakage, as it is entered into contact forms. Instead of relying on blacklists, FORMLOCK uses a combination of temporary third-party request blocking, browser cookies and cache rollback, and removal of sensitive parameters from a page's URL, going above and beyond generic anti-tracking extensions.

2 Data Collection

The web is a platform where the intended end-client is a human user. While browsers are used to render the incoming HTML content and relay the user's desires through HTTP requests, it is the user's responsibility to identify the rendered elements on a page and interact with them. This has allowed immense flexibility in the design of web pages where there can be near infinite different HTML and CSS arrangements that result in the same visual result.

For this reason, the web still resists reliable and generic automation. Consider, for instance, an HTML page rendered in an iframe and the same page rendered in a 5-level nested iframe. As far as the user is concerned, the two pages look identical and can be interacted with, in the exact same way. A crawler, however, needs to identify the iframe, switch contexts, and decide on a maximum number of nested iframes that it is willing to explore before moving on. In fact, there are many ways that pages can purposefully or accidentally trap naive crawlers in infinite loops, such as through the use of unbounded in-page calendars and an infinite number of dynamically generated links that always point to the same page.

For the purposes of this project, we need to be able to autonomously locate contact pages, successfully identify the expected type of input for each contact-form field, fill-in the appropriate details, and submit the forms. At the same time, we look for the accidental leakage of PII information via HTTP requests as well as the presence of active third-party content that is, or could be, exfiltrating a user's PII either in the clear or through the use of obfuscation. We opine that building a crawler that can reliably find one specific page (*i.e.,* the contact page) and successfully interact with it, is a harder task than finding any page and merely indexing it, as typical crawlers do.

2.1 Crawler Setup

The workhorse module of our crawler is PhantomJS [32], a JavaScript-enabled, headless browser based on Webkit. All traffic to and from PhantomJS is proxied through BrowserMob [5], a scriptable proxy with MitM capabilities, where it is analyzed for PII leakage. Through the use of the GhostDriver bindings [20], we implemented the logic of our contact-form-sensitive crawler in the Java language. In the following paragraphs, we outline the process with which we first identify and then interact with contact forms.

Finding the page containing a contact form

We argue that if websites want to encourage users to contact them, links to pages containing contact forms should be easy to find from the website's main page. Thus, for every website, we assume that there exists a separate page with a contact form, linked from the website's main page which the crawler can detect by the word "contact" appearing in the text associated with that link.

Locating the contact form within a page

Once the crawler has identified a page that could include a contact form it searches for an HTML form element that satisfies certain criteria. That is, precautionary steps must be taken to ensure that the crawler interacts with a contact form and not with other types of forms, such as, login forms and search forms. Thus, we exclude forms that are i) invisible or whose rendered dimensions are below a certain threshold, or ii) have less than two input fields (*i.e.,* one for a user's email address and one for the message itself). For these reasons, we do not attempt to identify forms implemented using different web technologies, such as, Flash or Java.

Filling and submitting forms

Even though one could attempt to fill-in all input fields with random text and press the submit button, we argue that this would not result in successful form submissions for a significant fraction of the web. Websites typically employ JavaScript functions that are triggered upon submission and inspect whether the text of each input field matches the expected format, e.g., ensuring that the value typed in a telephone input field only contains numbers and dashes. As such, our crawler looks for textual cues in order to identify what information to fill-in in any given textbox. When the type of an input field cannot be reliably determined, we opted to retype our control email address (*i.e.,* the email address created for our crawling experiments), since the lexical structure of email addresses have the capacity to satisfy many different input fields. We locate the form submission button by its input type, as well as its placement in relation to other input fields. Note that we chose not to attempt to break CAPTCHAs and thus any website requiring a CAPTCHA for a successful contact form submission is not represented in our results. Our decision was motivated by the difficulty of breaking modern CAPTCHAs using computer vision, and our unwillingness to support human-powered, CAPTCHA-solving sweatshops [25]. We argue that this decision does not change the take-away messages of our work, since all of our findings are already lower bounds. As such, if we would be able to support CAPTCHAs, the percentage of discovered PII-leaking forms could only increase.

We consider a form to be successfully submitted when a website redirects us to a new page that contains phrases indicative of a successful form submission, such as, "thank you" or "successfully submitted."

Detecting PII leakage

PII is the information about users that can later be used to trace users back to their real identity. In the context of contact forms, this information typically includes the users' first names and last names, their email addresses and their phone numbers. To track the dispersion of PII once it is submitted in a contact form, we always populated forms with the same first name and last name of a made-up persona, an email address created for the purposes of this experiment, and the universal contact message "I want to ask a couple of questions, how can I call you?". We consider PII to be leaked when it is sent to a third-party domain, that is, a domain other than the one present in a browser's address bar. In addition, we quantify the potential leakage through remote JavaScript providers, by tracking the dynamically-loaded, third-party JavaScript libraries located on a contact page which could, if they wanted to, obtain the user's information as that is being entered into a contact form.

2.2 Identifying PII in HTTP traffic

A detail that is missing from the above description is how exactly we identify a user's PII in HTTP requests. While one can straightforwardly search for PII being transmitted in the clear, this method will not be able to account for leaks that use various encoding or obfuscation techniques to hide the PII in the outgoing traffic. To account for these schemes, we devised the following methodology:

- We execute the crawl three times to have three clearly separate submissions per website contact form. In the first two submissions we use the same email address, while we choose a second control email address in the third submission.
- Third-party requests during each submission of the same contact form are matched by URL (including path) to obtain request triples in the form of:

$$(r_{i1}, r_{i2}, r_{i3}) \forall i \in R$$

  where R is the set of third-party requests occurring during and after the submission of a given contact form. Afterwards, for each request, we extract all sent parameters and their values, including parameters from the query string, cookies, headers, post data or multipart data.
- We compare the extracted parameters and values within each triple. If one of the request parameters

was the same during our first two form submissions (when the same email address was used), but was different in the third one (when we used a different email address), we mark this form as one that is potentially leaking user PII and requires manual analysis.

Even though this method cannot be used to automatically mark forms as suspicious when the requests resulting from a form submission vary significantly from the previous submissions of the same form, (such as in the case of modern encryption algorithms where the same plaintext is transformed into different ciphertexts), it can be used to automatically detect cases of encoding and obfuscation on otherwise stable forms of leakage. That is, our fuzzing method will be able to detect requests which carry parameters whose values change when the input data changes.

2.3 Crawler Results

To measure the leakage of PII on popular websites, we used our contact-form-sensitive crawler on the 100,000 most popular websites, according to Alexa [3]. The crawl was executed on a 16-core server located in the US, in January 2015.

The results of our crawl are as follows: Our crawler identified a contact page on 29.6% of the 100K websites. Due to the presence of contact pages with just email addresses or ones implemented using Flash or Java, our crawler successfully located contact forms on 17% of the original 100K websites. We were able to complete and successfully submit approximately 82.5% of the discovered forms. The distribution of the ranks of the websites where we were successful in identifying and submitting contact forms was uniform, i.e., the accuracy and reliability of our contact-page-sensitive crawler is not dependent on a site's popularity.

3 Analysis of crawler-collected data

In this section, we report on our analysis of the data collected from locating and submitting contact forms in the web's most popular 100K websites. The PII leakage is analyzed according to the road map presented in Figure 1. First, in Section 3.1, we discuss the cases of *potential* PII leakage, where websites give third parties access

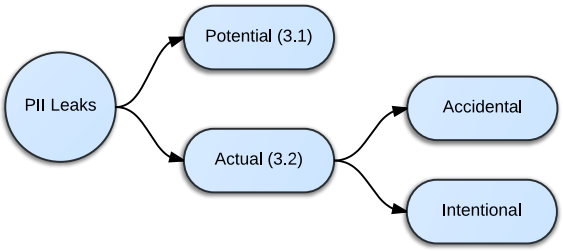

Fig. 1. Covered Variations of PII Leakage

to user PII by including remote JavaScript libraries on contact pages. Even though not all JavaScript libraries will try to maliciously exfiltrate a user's PII, any single one can, at any point in time, abuse its privileged position to do so. Second, in Section 3.2, we present the discovered cases of *actual* PII leakage, where a user's PII is visibly leaked to third parties via third-party scripts and content. We further categorize this actual PII leakage into *accidental* and *intentional*. For the former, we witness how poor coding practices are responsible for leaking user PII to thousands of third-party domains. For the latter, we discover that website owners intentionally leak a user's PII to third parties, typically through the outsourcing of contact forms and the use of various marketing tools.

We would like to emphasize that by "intentional leakage" we merely mean that a website, through the conscious use of third-party tools that receive user PII, is leaking a user's private information to third parties, who may or may not capitalize on that information in an abusive and opaque way. We *do not* mean that a website is, capriciously or maliciously, compromising the PII of its users.

3.1 Potential PII Leakage

JavaScript has emerged as the defacto client-side programming language of the web. The vast majority of modern websites use JavaScript to dynamically manipulate a page's DOM, enrich user experience through asynchronous requests and responses, and off-load non-critical, server functionality to the client.

As with any programming language, one can develop complex libraries using JavaScript and make diverse functionality available through well-defined APIs. In fact, programmers of web applications have come to rely on external third-party libraries to implement analytics (e.g. Google Analytics), interface enhancements (e.g. jQuery), and social-network integration (e.g. Face-

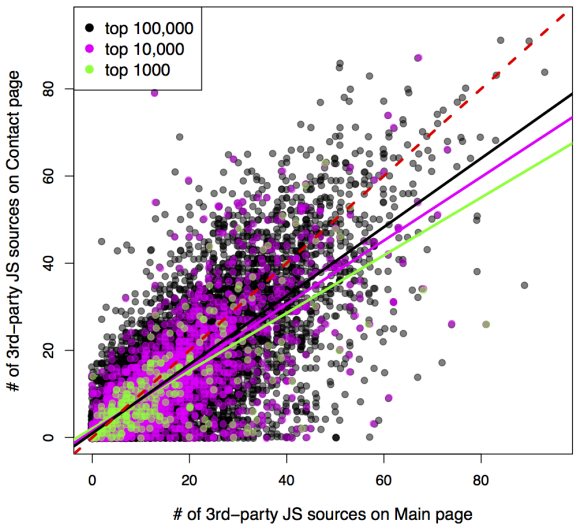

Fig. 2. Number of remote domains trusted for JavaScript on a website's main page versus its contact page.

book Like and Share). These libraries are loaded, on-demand, by a user's browser and incorporated into the origin of the including website. As such, in the absence of explicit isolating done through complicated, non-standard mechanisms, the providers of externally loaded JavaScript libraries have, by default, full control over the content of the webpage in which they are included. The remotely-included, third-party code can inspect and modify everything that a local JavaScript file can. For this reason, researchers have conducted measurements studies mapping the trust relationships between websites and the servers from where they include remote JavaScript code, showing the dangers of careless remote JavaScript inclusions [28, 41].

For our purposes, we are interested mostly in the JavaScript libraries included in contact pages and how these libraries differ in respect to the ones loaded on the main page of a website. Figure 2 shows a scatterplot of the number of unique third-party domains associated with the included JavaScript libraries on the main page of each website, versus the number of unique third-party domains associated with libraries on the contact page of the same site. The full lines are the regression lines for the top 1K, 10K and 100K domains, whereas the dotted line represents the $y = x$ which allows one to easily distinguish the websites including JavaScript from more unique domains on their contact page than on their main page (points above the dotted line). Based on Figure 2, we make the following observations:

– As prior studies have shown, popular websites include remote JavaScript sources from tens of remote unique domains. If any one of these remote domains gets compromised, the attacker can abuse the existing trust relationship to serve malicious code to the most popular websites of the Internet.
– Fortunately, most web administrators understand that a contact page is more sensitive that the main page of a website and thus choose to include less remote libraries on it. At the same time, 29% of popular websites do the exact opposite, with the worst offender including code from 20 times as many remote domains on their contact page, than on their main page.
– Inclusions of remote JavaScript code increase as we move towards less popular sites, both on the main page of a website as well as on its contact page.

Every single one of the remote JavaScript libraries present on a contact form can, with relative ease, read the user's PII as it is typed by the user, through the use of appropriate key-pressed event handlers. As such, we consider the excessive use of remote JavaScript libraries on contact forms to be *potential PII leakage*.

Due to the popularity of certain remote JavaScript providers (over all website pages), it is hard to observe the change of remote inclusions for smaller players in the contact form ecosystem. For this reason, in Table 1, we show the top remote code providers in terms of relative increase. For instance, zndsk.com was seen 54 times more on a contact page, than on a main page. One can see many entries that offer contact-form-handling services like zndsk.com (used by zendesk.com), desk.com and wufoo.com, as well as visitor tracking services like crazyegg.com. Finally, Oracle uses the rnengage.com domain for its RightNow software that also provides personalized communication with customers.

3.2 Actual PII Leakage

In this section, we turn our attention to actual PII leakage captured by our proxy, i.e., contact forms leaking a user's personally identifiable information to third parties. We categorize the leakage as either accidental, or intentional, depending on the way that a user's information is handled and our analysis of each case of leakage.

Accidental leakage

A common vector for actual PII leakage is through the well-understood practice of exposing sensitive informa-

Table 1. Remote JavaScript providers that appear much more on contact pages than on main pages.

Domain	Main	Contact	Times
More on Contacts			
zndsk.com	1	54	54
amikay.com	3	90	30
desk.com	4	89	22
wufoo.com	14	193	14
nr-data.net	27	372	14
rnengage.com	10	113	11
ooh.li	7	58	8
gwallet.com	19	117	6
ezakus.net	20	108	5
r1-cdn.net	28	122	4
Just on Contacts			
prfct.co	0	291	-
crazyegg.com	0	196	-
hrblock.com	0	175	-
zdassets.com	0	147	-

tion in URLs. For our study, we measure the number of contact forms that expose the user's PII by submitting them to the server using the GET method. For instance, if example.com has a contact form that uses the GET method to submit a user's contact data to www.example.com/submit, the user's browser will eventually create a request such as www.example.com/submit?fname=John&lname=Doe&msg=[...].
The web server will respond with a page containing some sort of acknowledgement and, in a typical web page, references to multiple remote resources. Thus, if, for instance, the acknowledgment page is using the Google analytics platform, the browser will request the appropriate JavaScript library from the google-analytics.com domain. That request, will also contain the Referer header (*i.e.,* the URL containing the form parameters) which will essentially be volunteering to Google the user's first name, last name, and any other contact details provided by the user on the contact form. Note that the Referer header is emitted with any request for a remote resource, thus a user's PII will also be leaked in requests for remote images, cascading style sheets, pages for iframes, SWF files, etc.

In our study, we discovered that 423 forms i.e. 2.5% of the total number of contact forms which our crawler submitted, leaked the user's PII through the URL query string. The data leaked to a total of 3,573 unique third-party domains with an average of 8 third-party domains receiving the user's PII per GET-submitting form. For our purposes, a third-party domain is either a domain name whose Public-Suffix+1 is different than the first-party domain (e.g. server.ads.com receiving PII via the Referer header of www.example.com), or a subdomain of the main domain whose IP address is registered to a different autonomous system (AS) than the IP address of the first-party website. We chose to broaden the notion of a remote domain since certain tracking and analytics companies bypass third-party restrictions imposed by the user by mapping a subdomain of the main first-party domain, e.g., tracking.example.com, directly to their remote servers [33]. Table 2 shows the 15 most popular third-party domains that ended up receiving our PII through the browser's Referer header. One can straightforwardly recognize popular companies offering libraries for analytics, social networks, advertising, and market research.

Even though one may be tempted to dismiss the seriousness of this issue due to the small overall percentage of PII-leakage through the URL (2.5%), it is important to recognize the *cascading effects* on a user's privacy, of even a single PII-including Referer header towards a popular third-party. Any single one of the 3,573 unique remote hosts where the browser voluntarily leaks the user's PII can now link the user's contact details, to whatever cross-domain tracking method they are already employing, such as, third-party cookies [34] or device fingerprinting [1, 2, 7, 26, 29]. Thus, as the user keeps browsing the web, the popular third-parties with a near-ubiquitous presence can now track that user, by name, across websites. While we do not know whether all 3,573 third parties make use of this accidentally leaked data, it is a safe assumption that some do. This assumption is backed up by historical examples where advertising companies do not hesitate to try any method that gives them an advantage over the rest of the competition [4, 12, 35, 36]. Finally, note that manually disabling the transmission of the Referer header does not fully solve the problem. While the providers of passive remote resources (such as images and Cascading Style Sheets) will no longer receive a user's PII, the active remote resources can still inspect the URL of the website in which they are included (e.g. inspecting the `window.location.href` property through JavaScript) which, in the cases of GET-submitted forms, will include a user's PII.

Intentional leakage

After accounting for the potential PII leakage through remote JavaScript inclusions and the actual, but accidental, PII leakage due to poorly-coded HTML forms,

Table 2. Third-party domains that most frequently receive user PII via the Referer header. The "Overall Presence" column shows the presence of each third party over the 100K websites of our study, thereby quantifying the cascading effects of accidental PII leakage

Domain	# of websites	Overall Presence
google-analytics.com	260	63.9%
google.com	174	41.8%
facebook.com	156	38.2%
doubleclick.net	150	45.0%
twitter.com	85	20.8%
googlesyndication.com	69	24.6%
scorecardresearch.com	58	16.3%
googleapis.com	58	40.3%
adnxs.com	50	20.8%
googleadservices.com	46	22.6%
rlcdn.com	40	7.3%
cloudfront.net	35	14.1%
yahoo.com	33	15.2%
...	...	...
gstatic.com	22	25.9%

there is one last category of PII leakage related to contact forms left to be discussed, namely, intentional PII leakage.

By analyzing the results of our crawling experiment and removing cases of accidental PII leakage we discovered that 6.1% (1,035) of all submitted contact forms leak a user's PII to third parties. Even though the exact nature of each third party is unique, one can, without too much loss of precision, categorize them in two broad categories, namely, *form builders* and *marketing solutions*.

Form Builders. Form builders include solutions that allow website administrators to deploy contact forms without implementing them or hosting them. Upon registering with a form builder, the form builder provides to a website administrator, either an automatically generated HTML form or a JavaScript snippet which is responsible for dynamically creating and styling a similar HTML contact form. In both cases, while the contact form is situated within the first-party website, the user's contact details will be submitted to the form builder. The form builder will then either forward each user inquiry to the form-embedding administrator via an email, or provide a more sophisticated control panel for managing and replying to user inquiries. Form builders are responsible for 3% of the total actual and intentional PII leakage. We list some of the most popular form builders in Table 3.

Even though form builders may initially appear harmless, it is important to realize that they are essentially invisible aggregators of user inquiries with the ability to store and correlate PII from a multitude of unrelated third-party domains. We classify them as invisible since there is no visual indication available to the user to communicate the fact that while her details are provided to what appears to be example.com they will, in fact, be submitted to formbuilder.com. To the extent that the submission of a contact form is analogous to the sending of a letter, we argue that users expect that only an intended recipient will be able to read their letters, without middle men that can opaquely open, inspect, and forward their letters. Note that, as shown in Table 3, some form builders do not even wait for a user to click "submit", before reading and sending out their PII.

Marketing solutions. Next to form builders, the vast majority of third parties receiving a user's PII from web contact forms belong to the broad category of marketing solutions whose aim is to capture, analyze and manage a website's target audience. These PII-handling products are responsible for 3.65% of intentional PII leakage over all submitted forms.[1] It is more likely that PII captured by those scripts will be shared with other parties and used for targeted advertising, email marketing, than other types of third parties, such as the aforementioned form builders. Table 3 shows the most popular marketing solutions that were the recipients of intentionally leaked PII.

Even though no two marketing solutions are identical, the marketing solutions that were the most unexpected, were *web leads*. Web leads are analytics platforms that go beyond the traditional analytics in that they try to identify users of interest, within the population of users browsing a website at any given time. As their name indicates, web leads platforms try to identify users of high value so that a sales team with constrained resources can focus their efforts on the users that are the most likely to be beneficial to their company. For instance, in the most straightforward, and least intrusive type of web leads, collected email addresses are combined with business information obtained by a combination reverse DNS-lookups on the visitors' IP addresses, and the augmenting of that information with data from

1 Note that the sum of these two categories is greater than 6.1% since some websites adopt either more than one marketing solutions, or a combination of a marketing solution and a form builder.

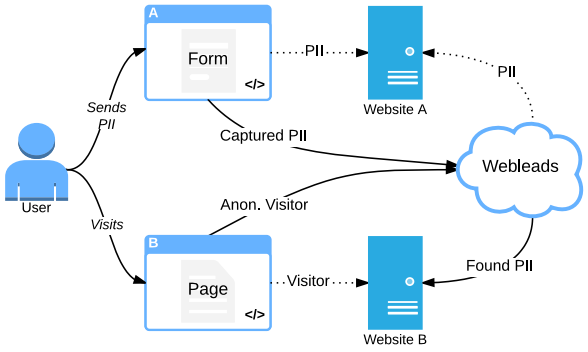

Fig. 3. High-level view of hidden, server-side, PII sharing between customers of specific web leads platforms

additional sources, such as LinkedIn. Thus, a sales agent can focus on users belonging to companies that are the most likely to become future customers. The captured PII becomes part of the user's tracked identity so that a user who has identified herself (via a contact form) to example.com, will be identified, by name, in all future visits of example.com. While the exact tracking details vary, we noticed that some companies, such as Active-Conversion and Sirius Digit'M, go beyond the use of stateful tracking, by using various fingerprinting techniques.

While researching the exact workings of the identified web leads companies, we encountered companies that use, what we consider to be, the most intrusive and least transparent technique for deanonymizing users. In this technique, as shown in Figure 3, all first party websites that are customers of one specific web leads platform share a common server-side PII database where contact details captured on one website, are available to all other websites. Specifically, the details that a user provides in the contact form of one website are captured and associated with a third-party cookie belonging to the domain of the web leads platform. This effectively means that thousands of sites that are customers of the same web leads platform and with which a user has never interacted, can suddenly deanonymize a fraction of their visiting users. As more and more websites participate in this "ring", more and more users can be effortlessly deanonymized. These companies, such as LeadLander [19] and the recently defunct Relead, openly claim to provide such deanonymizing functionality to their clients [18, 31] while users who have happened to be contacted by a website that they were browsing but never provided their details to, have openly expressed their discomfort [37].

Table 3. Third-party domains belonging to marketing solutions and form builders that capture user PII that is purposefully leaked via a contact form. Shaded entries denote domains belonging to companies offering marketing solutions. Note that some third-party JavaScript libraries leak a user's PII even before the user submits the contact form.

Domain	# of websites
On Submit	
wufoo.com	159
marketo.com	77
salesforce.com	71
hubspot.com	71
list-manage.com	67
aweber.com	46
formstack.com	37
eloqua.com	34
zopim.com	33
infusionsoft.com	30
blogger.com	25
addthis.com	23
zendesk.com	21
omtrdc.net	21
bizible.com	21
foxyform.com	20
2o7.net	20
emailmeform.com	18
caspio.com	15
pardot.com	15
...	...
trackalyzer.com	9
bluekai.com	8
...	...
leadformix.com	4
Filling Out	
sessioncam.com	5
veinteractive.com	3
hawksearch.com	1
kayako.com	1
onlinechatcenters.com	1
tendinc.com	1
hawksearch.info	1

Even though LeadLander has not achieved a high market penetration in the Alexa top 100K websites (discovered by our crawler on nine different websites as trackalyzer.com), the same cannot be said for its adoption by less popular websites. According to statistics kindly provided to us by Ghostery [11], LeadLander has been identified on more than 1,108 different domains.

Listing 4 shows a JavaScript snippet of LeadLander's PII capturing script, namely the part that is responsible for extracting a user's email address from a contact form. One can see that the script is trying to automatically identify the element containing a user's

Table 4. Actual PII leakage by website category

Category	% Forms	% Accident	% Purpose
Computers/Internet	18.5	2.0	8.5
Business/Economy	10.2	3.6	8.4
Shopping	7.4	2.3	5.3
Entertainment	4.9	2.5	3.7
News/Media	4.7	4.1	9.8
Blogs/Web Comm.	4.6	1.8	10.8
Education	3.1	2.7	5.6
Travel	3.0	3.5	2.5
Financial Services	2.8	1.9	3.0
Search/Portals	2.5	2.8	3.3
Games	2.4	2.2	3.1
Health	2.1	1.9	5.8
Sports	2.0	1.4	3.7
Pornography	1.9	0.9	2.7

email address by iteratively working through a long list of commonly used element names. The code is triggered on the form's `onSubmit` JavaScript event and transmits the captured input values to LeadLander via a tracking "pixel tag". This *plug-and-play* nature of the script means that a website administrator can just include the remote JavaScript library from the LeadLander platform and become part of their "deanonymizing ring" with little or no effort.

It is worthwhile pointing out that there may be many more web leads platforms that behave in similar ways. The unobservable, server-side sharing of PII information makes it, unfortunately, impossible to detect this PII sharing from the client-side, barring, of course, side-effects such as a user being contacted by a website to which she did not provide her email address. To obtain a rough approximation of this phenomenon, we tracked the email responses that we received from the contact forms submitted by our contact-form-sensitive crawler. Our inquiry to all recipients was the, rather universal, message "I want to ask a couple of questions, how can I call you?". Over the duration of our experiment we received 309 emails from third parties, that is, domains which our crawler never contacted.

3.3 Categories of first-party websites

One may wonder whether certain categories of websites are more prone to PII leakage than others. To investigate this category-based leakage, we used TrendMicro's public website categorization engine [39] to categorize the 16,981 websites on which our crawler was able to discover a contact form. Due to the large number of discovered categories (78), Table 4 shows the actual leakage of the categories of websites which comprised at least 2% of 16,981 websites.

```
var llelementemail = (document.forms[llformlooper2].
elements[llelementlooper].name)

if ((llemailfound==false &&
   (llelementemail=='email123')
   || (llelementemail=='signup_t1')
   || (llelementemail=='Sender')
   || (llelementemail=='jform[contact_email]')
   || (llelementemail=='web_form_1[field_2]value'
       && llfrmid==23954)
   || (llelementemail=='input_5' && llfrmid==25451)
   || (llelementemail=='your-email'
       && llfrmid==19415)
   || (llelementemail=='email')
   || (llelementemail=='name' && llfrmid==22344)
   || (llelementemail=='Email')
   || (llelementemail=='lldEmail')
   || (llelementemail=='si_contact_email')
   || (llelementemail=='Contact_Email')
   || (llelementemail=='mail')
   || (llelementemail=='e-mail')
   || (llelementemail=='E-mail')
   || (llelementemail=='E-Mail')
   || (llelementemail=='e_mail')
   || (llelementemail=='EMAIL')
    ...
```

Fig. 4. Snippet of LeadLander email-extracting JavaScript code. Notice how the script tries to guess the name of the DOM element contain a user's email address.

While the leakage of each category varies significantly, one can see certain patterns emerging. For instance, websites categorized as News/Media are consistently among the top sites that leak PII, in both accidental as well as intentional leakage. Moreover, the range of leakage in the accidental-leakage category is significantly smaller than the one in the intentional-leakage category. This shows that while programming mistakes and carelessness can appear in all websites, some categories of websites are much more likely to intentionally leak a user's PII. Lastly, we were surprised to see that sites related to pornography leak the least amount of PII. One reasonable explanation is that users, if they ever feel the need to contact an adult website, are unlikely to reveal their true identities in the contact forms. Thus any PII captured may be of little use to marketers.

4 Countermeasures

As we showed in the previous section, a user's PII, as that is entered into the contact forms of first-party websites, can land in the hands of third parties, either by omission or by commission. While one could, in princi-

ple, inspect the HTML and JavaScript code of a contact form before filling in their details, the technical skills required to do so are well outside the reach of everyday users of the web.

To remedy this situation, in this section, we describe the design and implementation of FORMLOCK, a browser extension that not only alerts users of "leaky" forms but also protects the user's PII from leakage, even if a user decides, despite our warning, to utilize a leaky contact form. In the following paragraphs, we describe the detection of leaky forms and the steps that FORMLOCK takes in order to safeguard a user's PII from client-side leakage. Section 7 provides a non-tracked download link for our FORMLOCK extension for the Chrome browser.

Form Warning

Given the current specification of HTML, the method of form submission (GET versus POST), the target of a form (first party versus third-party website) and the protocol of the submission (HTTP versus HTTPS), play no role in the visualization of a form. For instance, Moxie Marlinspike's SSL stripping techniques, included a MitM rewriting login forms converting them from submitting the user's credentials over HTTPS to submitting them over HTTP [21]. Marlinspike's attack abused, among others, the lack of any visual cues indicating the secure or insecure submission of credentials, as that is dictated by the protocol of the submitting form.

In the context of this paper, we argue that users need to know that a contact form can endanger their PII. To that extent, FORMLOCK highlights contact forms that either submit their inputs to third-party domains, or submit their inputs using the GET method. These form attributes are straightforwardly and reliably extractable from the contact form's HTML code (see Figure 5), ensuring that users are not flooded with false positives.

While not strictly necessary from a technical point of view, in addition to the highlighting of forms, FORMLOCK provides a context menu which the user can consult in order to find out the explanation why FORMLOCK marked any given contact form as leaky.

Form Locking

Even though interacting with a leaky form is likely to result in PII leakage, we expect that some users will want to contact a remote website, despite our warnings. Moreover, PII leakage may occur, not because of a leaky form, but because of dynamic scripts that will exfiltrate user information as soon as that is typed into a contact form. To accommodate these cases, FORMLOCK has a "Lock-Form" mode which the user can enable and which attempts to lock-down the user's browsing environment so that the chance of leakage is reduced, if not altogether eliminated. This lock down consists of the following steps:

1. Upon the enabling of the "Lock-Form" mode, FORMLOCK temporarily blocks all requests towards servers other than the first-party server and the server that is denoted in the action parameter of the contact form. This will stop the leakage of PII by invasive scripts which extract a user's details while those are being typed or upon form submission, without hindering the originally intended submission of the contact form. Note that we cannot stop the submission of a form towards a third-party domain, as in the case of form builders, since if we do, the user's contact request will never reach its intended destination.
2. FORMLOCK records the timestamp of the lock-down so that it can, upon releasing the lock (Step 4), remove all new browsing data, including cookies and values in HTML5 storage. We included this feature in order to defend against scripts which, upon discovering that they cannot contact the outside world, would "stash" the user's PII in cookies and HTML5 storage, so that they can leak it *after* the submission of the contact form, when their connectivity will be restored.
3. If a form is expected to be submitted using the GET method, FORMLOCK cannot just change the protocol of the submission to POST since the server-side code that is meant to receive the user's details will likely fail. Instead, FORMLOCK allows the submis-

```
for (var f = 0; f < document.forms.length; ++f) {
    var curForm = document.forms[f];

    var actionDomain = curForm.getAttribute('action');
    // Supporting functions
    actionDomain = getHostname(actionDomain);
    actionDomain = getRootDomain(actionDomain);

    // tabDomain is provided upon injection
    if (tabDomain !== actionDomain ||
        curForm.method === "get") {
        curForm.style.border = "medium solid red";
    }
}
```

Fig. 5. Snippet of FORMLOCK, responsible for identifying and highlighting "leaky" contact forms

sion of the form using a GET request while all third-party requests are blocked, and removes parameters from the resulting URL before refreshing the page and releasing the lock (Step 4). As such, the user's details will not leak via the Referer header or the `document.location.href` property.

4. Once a form is successfully submitted, the user can request from FORMLOCK to go out of the "Lock-form" mode. Upon that request, FORMLOCK reloads the page (essentially stopping all scripts that may be constantly trying to leak the captured PII), restores the cookies and HTML5 storage to their previously stored state (Step 2), stops the temporary blocking of third-party requests, and reloads the contact page, thereby fully restoring the functionality of the original page.

Limitations

Since FORMLOCK is a browser extension, it is, by definition, only able to stop client-side leakage, occuring through third-party requests which leak PII using headers and parameters. This leakage is stopped by Formlock through the temporary blocking of all third-party requests and the rollback of client-side state. If, however, a first party colludes with a third party to leak information from the server-side of the first party to the third party, then FORMLOCK will not be able to stop this kind of PII leakage.

Future work

We intend to extend FORMLOCK in order to add crowdsourcing capabilities, where an extension, after receiving a user's explicit opt-in, will report to our servers the websites where leaky forms were discovered, without, of course, reporting the user's PII. This will allow us to expand our understanding past Alexa's top 100K websites, and particularly to smaller and more regional websites which tend to evade investigation due to their absence from the lists of popular websites provided by Alexa and Quantcast. Similar crowdsourcing techniques have been used both by researchers [24] as well as commercial extensions [11] and have allowed them to quantify phenomena of interest at a scale which would never be realizable via active crawling. We will further investigate the exact architecture of Formlock's crowdsourcing features, to provide as much privacy as possible to the participating users.

5 Related Work

To the best of our knowledge, there has been no study of intended and unindented PII leakage through HTML contact forms that is of comparable breadth to our work.

Referer-based studies

Krishnamurthy et al. manually investigate the PII leakage of 120 popular websites that occurs during sign-up and normal interaction due to GET-submitting forms [15]. The authors discovered that 48% of these sites leaked a user identifier to a third-party website. Using a similar methodology, Mayer signed-up and interacted with 250 websites while recording all requests with the FourthParty measurement platform [10], finding that 61% of the tested websites leaked a user's PII to third parties [22].

Krishnamurthy and Wills performed a separate study for online social networks (OSNs) where they manually interacted with 12 OSNs finding cases where the PII that a user entrusted to an OSN would unintentionally find its way to third parties [17]. Chaabane et al. recently investigated the leakage of user PII via OSN applications by automatically interacting with more than 1,200 OSN applications on two different OSNs [6]. The authors discovered that 22% of Facebook applications and 69% of RenRen applications leak, intentionally (via an HTTP request) or unintentionally (via the Referer header), a user's PII to forth-parties, that is, parties other than the third-parties who serve the OSN application. In our study, we opted for a more fine-grained analysis of intentional and accidental leakage since we observed that many generic trackers send to third-parties the URL of their hosting page. As described in Section 3.2, when the current URL is a result of a GET-submitted form, it will include PII which will be accidentally sent towards third-parties and thus, in the methodology of Chaabane et al., will be counted as intentional leakage.

Nikiforakis et al. investigated the referer-anonymizing services that some websites use to hide their presence from linked websites [30]. The authors found that the use of such services sacrifices the privacy of users in order to protect the privacy of the original website, since advertisers who are present on these services can link a user across more websites than they could, had the user transited directly from the linking site to the linked one.

JavaScript-based studies

More than one study has established that the ease of dynamic inclusions of remote JavaScript code does not come without a cost. From a security perspective, including remote code that can change without notice opens up websites to attacks where an attacker, instead of attacking a website directly, attacks the servers that are trusted for JavaScript code and alters the served JavaScript scripts [28, 41]. In fact, there have been many real-world examples of such attacks, with a recent one being the Syrian Electronic Army redirecting visitors of Reuters by actually compromising one of the advertising networks that Reuters was trusting for ads (via the use of remote JavaScript inclusions) [14]. We suspect that, due to the lack of visual changes, a similar attack where a remotely-included JavaScript library would be maliciously modified to exfiltrate a user's PII, would remain undetected for a substantial amount of time.

From a privacy perspective, the ease of including remote JavaScript code has enabled the increased adoption of tracking technologies [16, 23], where websites can now receive detailed tracking information about their users by simply adding a small snippet of JavaScript in their pages. Recent studies have shown that next to traditional stateful, cookie-based tracking [9, 34, 35], many websites are now tracking their users using stateless tracking techniques, which typically involve the fingerprinting of a user's browsing environment [1, 2, 7, 26, 29]. As described in Section 3.2, some of the marketing solutions that we discovered on the contact pages that intentionally leaked user PII are also making use of fingerprinting techniques.

6 Conclusion

In this paper, we conducted the first large-scale study quantifying the leakage of personally identifiable information from the contact forms of websites. Contact forms are where users are asked to give up their anonymity and communicate, by name, with a website. As such, the pages that host these forms must respect their users and not leak, either by accident or intentionally, a user's PII to third parties. By designing a custom crawler that can autonomously identify and fill-in forms while monitoring browser requests, we were able to characterize the PII leakage on the 100,000 most popular websites of the web. By analyzing the crawler-collected data, we discovered that, apart from having their details intentionally leaked to third parties, users are requested to enter their details in pages laden with third-party content and poorly-coded HTML forms, resulting in thousands of third-party domains receiving user PII, without the user's knowledge, or consent.

Instead of merely measuring the problem and relying on web programmers and browser vendors to take steps to decrease the discovered PII leakage, we designed and implemented Formlock, a browser extension that warns the user of leaky forms and has the ability to temporarily lock-down a page so that a user's PII can be protected, both against accidental as well as intentional client-side leakage. We hope that this study can educate developers about the privacy consequences of sloppy programming practices on the web, raise awareness of the presence and practices of certain intrusive online marketing platforms, and equip users with the technology to reclaim the lost control over the dissemination of their PII.

Acknowledgments: We thank the anonymous reviewers and our shepherd Erman Ayday for their valuable input. We also thank Ghostery and Evidon for sharing with us data about the presence of specific web leads companies outside of the Alexa top 100K webpages. Some of the described experiments were conducted on virtual servers kindly provided to us by Linode. This work was supported by NSF grant CNS-1527086.

7 Availability

The prototype of the Formlock Chrome extension is available via the following code repository:

`https://github.com/ostarov/Formlock`

References

[1] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The Web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*, 2014.

[2] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel. FPDetective: Dusting the Web for fingerprinters. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2013.

[3] Alexa. Top 1 million websites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[4] J. Angwin. Meet the Online Tracking Device That is Virtually Impossible to Block. http://www.propublica.org/article/meet-the-online- tracking-device-that-is-virtually-impossible- to-block, 2014.

[5] BrowserMob Proxy. http://bmp.lightbody.net/.

[6] A. Chaabane, Y. Ding, R. Dey, M. Ali Kaafar, and K. Ross. A Closer Look at Third-Party OSN Applications: Are They Leaking Your Personal Information? In *Passive and Active Measurement conference (2014)*, Los Angeles, United States, Mar. 2014. Springer.

[7] P. Eckersley. How Unique Is Your Browser? In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, pages 1–17, 2010.

[8] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[9] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 289–299, 2014.

[10] FourthParty: Web Measurement Platform. http://www.fourthparty.info.

[11] Ghostery. https://www.ghostery.com.

[12] J. HOFFMAN-ANDREWS. Verizon Injecting Perma-Cookies to Track Mobile Customers, Bypassing Privacy Controls. https://www.eff.org/deeplinks/2014/11/verizon-x-uidh, 2014.

[13] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 639–652. ACM, 2011.

[14] F. Jacobs. How Reuters got compromised by the Syrian Electronic Army. https://medium.com/@FredericJacobs/the-reuters-compromise-by-the-syrian- electronic-army-6bf570e1a85b, 2014.

[15] B. Krishnamurthy, K. Naryshkin, and C. E. Wills. Privacy leakage vs. protection measures: the growing disconnect. In *Web 2.0 Security and Privacy Workshop*, 2011.

[16] B. Krishnamurthy and C. Wills. Privacy diffusion on the web: A longitudinal perspective. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 541–550, New York, NY, USA, 2009. ACM.

[17] B. Krishnamurthy and C. E. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 7–12. ACM, 2009.

[18] Who are these tracking companies? Meet LeadLander. http://www.abine.com/blog/2012/leadlander/, 2012.

[19] LeadLander. *Website Visitor Analytics*. http://www.leadlander.com.

[20] I. D. Marino. *Ghost Driver / PhantomJSDriver*. https://github.com/detro/ghostdriver.

[21] M. Marlinspike. New Tricks for Defeating SSL in Practice. In *Proceedings of BlackHat 2009*, DC, 2009.

[22] J. Mayer. Tracking the trackers: Where everybody knows your username. http://cyberlaw.stanford.edu/node/6740, 2011.

[23] J. R. Mayer and J. C. Mitchell. Third-party web tracking: Policy and technology. In *IEEE Symposium on Security and Privacy*, pages 413–427. IEEE Computer Society, 2012.

[24] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Crowd-assisted Search for Price Discrimination in E-Commerce: First results. In *Proceedings of the 9th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2013.

[25] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage. Re: Captchas-understanding captcha-solving services in an economic context. In *USENIX Security Symposium*, volume 10, page 3, 2010.

[26] K. Mowery and H. Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In *Proceedings of the Workshop on Web 2.0 Security and Privacy (W2SP)*. IEEE Computer Society, May 2012.

[27] Network Advertising Initiative. Understanding Online Advertising. https://www.networkadvertising.org/faq.

[28] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. You are what you include: Large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 736–747, New York, NY, USA, 2012. ACM.

[29] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 541–555, Washington, DC, USA, 2013. IEEE Computer Society.

[30] N. Nikiforakis, S. Van Acker, F. Piessens, and W. Joosen. Exploring the Ecosystem of Referrer-Anonymizing Services. In *Proceedings of the 12th Privacy Enhancing Technology Symposium (PETS)*, pages 259–278, 2012.

[31] D. Nix. You're not anonymous. I know your name, email, and company. https://web.archive.org/web/20140103065932/http://blog.42floors.com/youre-not-anonymous-i-know-your-name-email- and-company/, 2012.

[32] PhantomJS. *Headless WebKit*. http://phantomjs.org.

[33] E. Picard. We Don't Need No Stinkin' Third-Party Cookies. http://adexchanger.com/data-driven-thinking/we-dont-need-no-stinkin-third-party-cookies/, 2013.

[34] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 12–12, Berkeley, CA, USA, 2012. USENIX Association.

[35] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle. Flash cookies and privacy. In *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.

[36] E. Steel. A Web Pioneer Profiles Users by Name. http://www.wsj.com/articles/SB10001424052702304410504575560243259416072, 2010.

[37] S. Sunam. *Google Plus post on December 8, 2012*, (accessed February 23, 2015). https://plus.google.com/+SumitSuman01/posts/2jLJ5B4yPYF.

[38] The Wall Street Journal. What They Know. http://www.wsj.com/public/page/what-they-know-digital-privacy.html.

[39] Trend Micro Site Safety Center. http://global.sitesafety.trendmicro.com/.

[40] B. Ur, P. G. Leon, L. F. Cranor, R. Shay, and Y. Wang. Smart, useful, scary, creepy: Perceptions of online behavioral advertising. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 4:1–4:15, New York, NY, USA, 2012. ACM.

[41] C. Yue and H. Wang. Characterizing insecure javascript practices on the web. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 961–970, New York, NY, USA, 2009. ACM.