Identifying Personal Information in Internet Traffic

Yabing Liu
Northeastern University
ybliu@ccs.neu.edu

Han Hee Song
Cisco Systems
hanhsong@cisco.com

Ignacio Bermudez
Symantec Corporation
ignacio_bermudezcorr@symantec.com

Alan Mislove
Northeastern University
amislove@ccs.neu.edu

Mario Baldi
Cisco Systems & Politecnico di Torino
mbaldi@polito.it

Alok Tongaonkar
Symantec Corporation
alok_tongaonkar@symantec.com

ABSTRACT

Users today access a multitude of online services—among the most popular of which are online social networks (OSNs)—via both web sites and dedicated mobile applications (apps), using a range of devices (traditional PCs, tablets, and smartphones) that are connected via a variety of networks. The resulting infrastructure makes these services conveniently available anytime and anywhere, enabling them to become an integral part of daily life. As a consequence, users explicitly and implicitly provide a wealth of Personal Information (PI) that reflects several aspects of their life. Service providers monetize this information by selling to third parties (e.g., advertisers). Unfortunately, today, it remains difficult for end users to fully understand the amount and nature of the collected data.

Our goal in this paper is to bring visibility into PI collected when accessing online services such as online social networks. This is a major challenge because PI is transferred in a proprietary way by each service. We develop a novel method that can automatically discover various types of PI carried within protocol fields of network traffic; the method includes techniques to filter out potential "containers" that do not actually carry PI and extend the set of containers initially found with additional ones. We evaluate the false positive/negative rates of our proposed method and show examples of interesting findings, including what kind of web sites or apps are more likely to transmit PI and which types of PI are most commonly collected.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: Security and protection; K.4.1 [**Computers and Society**]: Privacy

Keywords

Privacy; Leaks; Web sites; Mobile applications

COSN'15, November 2–3, 2015, Palo Alto, California, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3951-3/15/11 ...$15.00.
DOI: http://dx.doi.org/10.1145/2817946.2817947.

1. INTRODUCTION

People heavily and constantly rely on services accessible through the Internet for professional, personal, and entertainment needs. For example, online social networks (OSNs) are a popular way for individuals to keep in touch, communicate, and share content. These services are accessed not only via web sites, but also via dedicated applications (apps) on mobile devices, thus being accessible through a range of devices (PCs, tablets, and smartphones) and a variety of wired, wireless, and mobile networks. Given that such services are an integral part of users' lives, service providers have a privileged observation point into the habits and interests of their users. Hence, many operate using a similar business model: services are made available for free in exchange for users (allegedly with an informed decision) accepting that service providers monetize on such personal data by selling it to third parties (e.g., advertisers).

Even though users are often provided with privacy controls, these generally only affect flow of information to other users or third-party applications; users today have no option of making their data private *from* the service provider. Even worse, the limited visibility into app behavior coupled with the significant amount of data stored on smartphones makes it even harder for users to understand the extent to which these services are automatically collecting personal data.

In this paper, we address this situation by developing techniques that automatically detect *personal information* (PI) traveling through the network as it is collected by services accessed via web browsers or mobile apps. This phenomenon is hereafter referred to as PI *leaks*. In contrast with related approaches that rely on "rooting" a user's device [4, 6] or instrumenting applications or browsers [1, 17], we instead aim at a solution requiring access only to the network itself, because such an approach significantly lowers barriers to deployment. Moreover, it has the potential to achieve higher coverage since the system can leverage visibility on traffic from multiple users in learning how PI is transmitted (i.e., how it is encapsulated in proprietary ways). Although the approach is general, we choose to focus on HTTP, as this is the protocol on which both traditional web services and a large fraction of mobile apps base their communications.

In designing and evaluating our approach, we make three high-level contributions. First, we underscore the difficulty of the problem of locating PI in network traffic by demonstrating that only a very small fraction of protocol fields actually convey PI, making our endeavor akin to "finding a needle in the haystack" (in § 4.1). We also show that an approach based on simple statistical analysis (e.g., selecting fields that are unique to a user, or common across different services) is not practical as it results in unacceptably high levels of false positives/negatives (in § 4.2).

Second, we develop a novel method based on (*i*) grouping data according to the domain name of the servers it is sent to and the key associated to it for the transmission, called a *domain-key*, and (*ii*) concluding that all of the domain-key combinations in a group are PI "containers" if a threshold subset of them are found to contain PI (§ 5.1). This subset of PI containers are identified through a list of *seed rules* manually crafted to locate PI of different types, including, but not limited to, users' names, genders, email addresses, ages, geo-locations, cities, postal codes, and phone numbers (§ 5.2 and § 5.3). Then the coverage is extended by inferring additional containers by analogy with the seeded ones (§ 5.4).

Third, we evaluate our approach on a network dataset collected from a point-of-presence of a European ISP, covering 13,000 real users. As we do not know the ground truth PI for these users, we establish ground truth by relying on multiple human raters to label domain-keys that contain PI. In § 6.1, we find that our approach is able to identify these rare domain-keys automatically, with a low false negative rate (2.7%) and an acceptable false positive rate (13.6%). We then apply our approach to the entire dataset in § 6.2, exploring the frequency with which web sites and applications transmit PI in practice. There, we discover that different types of Internet service focus on different PI (*e.g.*, CDNs tend to leak physical locations of users while adult services leak age and gender information). We also find that an invasive user-tracking service leaks higher amount of PI than others.

2. BACKGROUND

In this section, we provide a more formal definition of personal information (PI) and describe our assumptions and intended operating environment.

2.1 Personal Information (PI)

There are many different kinds of PI, including a user's name and social security number, their current location when performing a purchase over the Internet, or even rich media information in the form of photos and videos. For simplicity, we focus on a text-based personal information (e.g., names, user identifiers, and locations) collected by web sites and smartphone applications. Further, to discern different high-level characteristics of PI, we classify them along the following three dimensions.

Static vs. Dynamic Static PI does not typically change over time (at least, over short- to medium-length time intervals). Examples include the user's name, gender, phone number, and email address. In contrast, dynamic PI may change over such intervals; examples include the user's geo-location, a user's session ID, or the user's set of personal interests.

Unique vs. Non-unique Unique PI distinctly identifies a single (human) user from others. For example, a user's email address or phone number uniquely distinguishes a user from the rest. On the other hand, non-unique PI may be shared by multiple users; examples include a user's name, gender, or date of birth.

Personal information	**Static?**	**Unique?**	**Shared?**
Name	☑	☐	☑
Email address	☑	☑	☑
Date of Birth	☑	☐	☑
Geo-location	☐	☐	☐
Username	☑	☑	☑
Tracking cookie	☐	☑	☐

Table 1: Examples of different types of textual PI, and the breakdown of PI along different dimensions.

Shared vs. Distinct The third dimension we consider is PI that, for a given user, is likely to be shared *across services* or distinct. An example of shared PI is mailing address of a user (presuming that the user provides factual information to each site). In contrast, distinct PI is potentially different for each website (or domain name); examples include time the user last logged in or the session identifier in a tracking cookie.

We provide breakdowns of how different examples of PI are classified along these dimensions in Table 1.

2.2 Assumptions, environment, threat model

In contrast to approaches that assume access to user devices (e.g., via browser plugins, "rooting", or operating system modifications), we instead assume that network administrators wish to understand when their users' personal information is being transmitted over the network. Thus, we assume that we have access to traces of network activity from a large group of users, as would be the case at a large corporation or university.

While web sites are constrained by the browser to only using HTTP to communicate with remote servers, smartphone applications are free to use any UDP/TCP protocol. However, as much as 40% of application traffic actually is HTTP [5,21], presumably to re-use many of the same APIs as web-based services and to avoid certain firewalls. Hence, we only consider PI leaks that occur over HTTP, but our approach could easily be extended to other protocols if given appropriate parsers.

We therefore develop techniques to look for instances of textual PI in certain HTTP fields of the observed network traffic. We assume that applications and web sites are not actively obfuscating transmitted information by hiding PI or obscuring data by using steganography-like techniques. Handling PI transmitted by actively adversarial applications (e.g., malware) introduces significant additional challenges, and we leave it to future work.

Finally, because we do not assume any privileged access to devices, we are unable to gain visibility into HTTPS traffic.[1] While HTTPS and, more generally, any TLS/SSL encrypted traffic, represent an ever increasing fraction of Internet data, we find that a significant fraction of traffic remains in plain

[1]We assume that mechanisms to access encrypted content can be put in place independently of our techniques for PI identification. For organizations that provision devices with additional trusted root certificates (e.g., certificates owned by the local administrators), one could leverage techniques for interposing on HTTPS transactions to gain such visibility [14,20], thereby extending our approach to HTTPS traffic as well. However, such an approach would have significant implications on privacy and security, so we do not consider it to be the common deployment scenario.

Observed HTTP transaction

```
GET /foo.html? user_firstname=Alice & id=17  HTTP/1.1
Host:  imagevenue.com
Cookie:  a=293 & g=00s9229daa & age=39 & id=27
ETag:  2039-2dc90ea2-12
Referer:  http://www.facebook.com/? user_id=89
Accept-Encoding:  deflate,gzip

HTTP/1.1 200 OK
Date:  Mon, 23 May 2013 22:38:34 GMT
...
```

Derived domain-keys and values

Domain	Key	Field	Value
imagevenue.com	user_firstname	GET	Alice
imagevenue.com	id	GET	17
imagevenue.com	a	Cookie	293
imagevenue.com	g	Cookie	00s9229daa
imagevenue.com	age	Cookie	39
imagevenue.com	id	Cookie	27
imagevenue.com	user_id	Referer	89

Figure 1: Example query from one HTTP connection, and the derived domain-keys with the associated values.

HTTP. For example, in our *Lab traffic* dataset described in Section 3.2, we find that 62% of the flows are HTTP, and that 44% of the ground-truth user PI is located in these HTTP flows. As a result, even without access to HTTPS traffic, we still can observe a large fraction (44%) of PI transmitted in the traffic.

3. DATASET DESCRIPTION

In this section, we introduce the datasets used for characterization of PI transmission over the network and for evaluating extraction of PI from traffic. While it is ideal to collect traffic traces with ground truth PI on every user, it is unrealistic to be able to collect such data at large scale. Instead, we use two complementary datasets: (i) small scale traffic traces with reliable ground truth collected in a controlled lab environment (*Lab traffic*) and (ii) large-scale traces collected from an ADSL point-of-presence (*ISP traffic*). The *Lab traffic* dataset helps us to obtain preliminary understanding on the mechanisms of PI transmission on the Internet; the *ISP traffic* dataset is then used to build models for PI extraction and test them at scale.

We begin this section by overviewing how we parse out PI from raw traffic, followed by detailed explanation of the *Lab traffic* and *ISP traffic* datasets, respectively.

3.1 HTTP parsing

From Layer-7 flows of user traffic, we extract PI of users from HTTP requests assuming that it is transferred in the form of *key-value* pairs. In order to properly handle these keys, we use the concept of *domain-key* in which we combine a key name with the name of the domain associated to the request. The intuition behind this is that key names will likely be used coherently (i.e., for carrying the same type of information) within the same domain (e.g., google.com may use the keyword "ggender" to collect user's gender regardless the specific Google service). On the other hand, a same key might be used in the context of different domains with a different meanings (e.g., the key `id` may be used different by Google and Facebook). Specifically, we extract the domain from `Host` HTTP header, and derive keys (and values) from three locations: (a) the query string of HTTP GET requests, (b) the query string in the `Referer` HTTP header, and (c) the `Cookie` HTTP header. For each location of potential keys, we divide the contents into key/value pairs using standard formatting rules (e.g., for GET query string parameters, we use the `&` character; for cookies, we use commas, semicolons, and ampersands). Figure 1 shows an example of a query and domain-keys and values extracted from it.

The use of domain-keys (as opposed to just keys alone) allows us to capture how different domains use keys with the same name. Consider two HTTP GET messages `http://loginradius.com/login?name=alice` and `http://ymail.com/getservice?name=send_mail`. While the query string `name` is the same for both domains, the former domain uses it as a login ID, while the latter uses it the name of the service that the user is requesting. Thus, keeping these separate allows us to identify the former as potentially carrying PI, while the latter is unlikely to.

Dataset	HTTP flows	Tuples	Domain-keys
Lab traffic	9,227	20,810	8,372
ISP traffic	40,775,119	51,368,712	3,113,696

Table 2: High-level statistics of our two datasets.

3.2 Controlled lab environment dataset

In the *Lab traffic* dataset, we collect traffic of a single user who intentionally transmitted a variety of *known* PI (*i.e.*, ground truth) to a number of popular online services. For this dataset, we wanted to be able to examine the TLS/SSL encrypted traffic as well. Thus, we collect data by tapping in to the connections with a middlebox that passes traffic through VPN/Man-In-The-Middle (MITM) transparent setup [15].

Using an Apple iPhone 4 with iOS 7, we ran the top 35 free iTunes apps for 15 minutes each, conducting a variety of activities including service registration, login/logout, message posting, chat message transmission, browsing, etc. Detailed information about the size of this dataset is provided in Table 2. Given that this dataset is highly biased toward one specific user and one device, we use this dataset as an exploratory sandbox, in which we manually inspect how the known PI is transmitted in HTTP/S traffic.

3.3 Real ISP dataset

In order to generalize our PI extraction models with larger scale data, we conducted a large-scale traffic collection from a point-of-presence of an ISP providing ADSL service in a European city. We use a Layer-7 traffic trace, *ISP traffic*, of 13,000 users for 24-hour in August 2011.[2]

We identify over 3 million unique domain-keys and 51 million unique domain-key/value tuples (see Table 2 for more details). While this real-user dataset lacks ground truth information on the user PI, we evaluate correctness of our proposed method by having humans manually inspect sampled domain-keys in Section 6.1.3.

[2]Per the privacy policy of the ISP, we anonymized true identities of the users prior to our study.

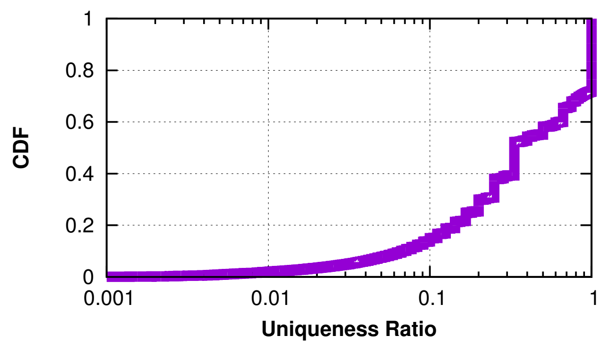

Figure 2: The cumulative distribution of the *uniqueness ratio* of all static domain-keys in *ISP traffic* dataset.

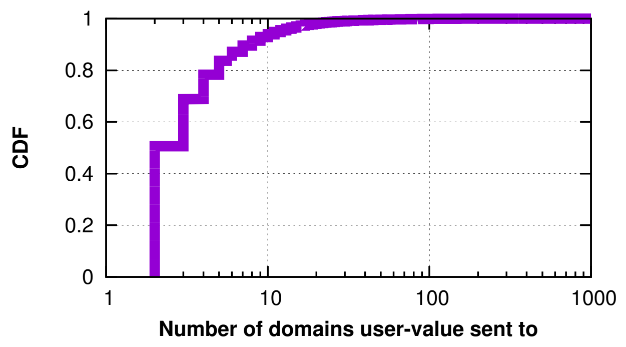

Figure 3: The cumulative distribution of number of domains that each user-value has been sent to in *ISP traffic* dataset.

4. STATISTICAL APPROACH

We begin by measuring the overall level of PI present in traffic, and then explore whether simple statistical techniques (inspired by Section 2.1) might be able to identify PI leaked in large ISP traffic.

4.1 Small scale study on controlled lab traffic

Using the *Lab traffic* dataset, we locate all of the domain-keys present in the traffic; this results in 8,372 domain-keys in total. Because we know the ground truth entered by the user, we search through the values of the domain-keys, looking for information that was provided by the user including the email address, name, city, postal code, gender, age, and geo-coordinates. We find that the fraction of domain-keys with different PI varies between 0.01% (for phone number) to 0.31% (for postal code), but overall, only 1.25% of *all* domain-keys ever contained any PI. Thus, we observe that locating PI in raw traffic is akin to finding needle in a haystack.

4.2 Statistical metrics in discovering PI

Next, using *ISP traffic* dataset, we explore whether we may be able to identify PI in traffic by looking for simple statistical properties of the domain-keys. For example, perhaps values that users upload to different domains may be more likely to be PI than other values. Below, we explore each of the three properties of the domain-keys in the *ISP traffic* dataset based on the taxonomy presented in Section 2.1. Then, we analyze the effectiveness of combining the statistical metrics in discovering the PI leakage.

Static vs. Dynamic Overall, we find that there are 341,179 (11.0%) static domain-keys (*i.e.*, every user has only one value for the domain-key) and 111,664 (3.6%) dynamic domain-keys (*i.e.*, every user has two or more values); the remaining are mixture of both (*i.e.*, some users have single value, some have multiples). This is unsurprising as the vast majority of domain-keys have very few values. Manually inspecting the static and dynamic domain-keys reveals a few candidates for PI, but the majority of domain-keys have no obvious semantic meaning for PI. Thus, while this approach does identify domain-keys that are static or dynamic for users, it is still not precise enough to be useful for pinpointing PI.

Unique vs. Non-unique The second feature we explore is whether each user is mapped to a unique value. We focus on the static domain-keys discovered above in order to use domain-keys that are likely to map a single value to each user (i.e., examining uniqueness of dynamic values requires careful consideration of the number of online users, etc). We define a new metric *uniqueness ratio* for a domain-key, which is simply the number of unique values in the domain-key divided by the number of users for the domain-key. We show the cumulative distribution of the *uniqueness ratio* in Figure 2. Among all the static domain-keys, 96,375 (28.24%) of them have a *uniqueness ratio* of 1, meaning every user has a unique value.

While the static domain-keys with *uniqueness ratio* of 1 are more likely to contain the static type of PI, such as user's username, and email address, the majority of them are comprised of session identifiers, GUIDs, and the like. Thus, relying on uniqueness alone is also likely to produce too many false positives to be useful.

Shared vs. Distinct Data a user is sharing across domains suggests that the value may correspond to the user's PI (e.g., the same email address used as login account for different websites). Out of 26,453,858 unique user-values, we find 5,923,084 (22.4%) of them have been sent to multiple domains; we show the cumulative distribution of the number of domains for each user-value in Figure 3. There, we find over 7% of user-values have been sent to at least 10 domains.

Among the values sent to multiple domains (*i.e.*, distinct data), we find some meaningful PI (e.g., we find tracking user identifiers that are used across domains). However, the majority of them are common values with no implications on PI such as 0, 1, true, false, etc (*e.g.*, a user sent the value of 1 to more than 100 different domains). Thus, as before, looking only at the values that are shared across domains is simply not precise enough to effectively locate PI.

Combining features We have observed so far that looking for individual features of domain-keys is not precise enough to locate PI. We now briefly explore combining statistical features together, looking for domain-keys that are static, unique to a user, *and* shared across domains. Our selection of these features is to capture PI such as email addresses, which may be used as login information for multiple web sites. This combined criteria leaves with a small set of 262 *pairs* of domain-keys that are unique, static, and have at least 20 users that share the same value in both domain-keys.

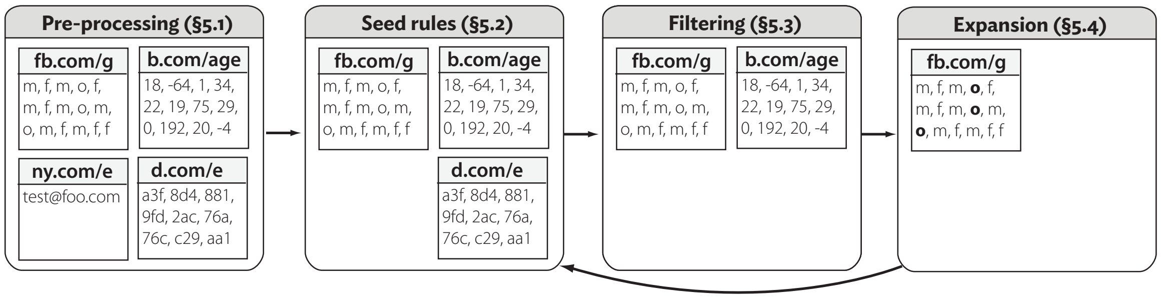

Figure 4: Overview of our proposed approach, exemplifying the four steps in identifying domain-keys that may contain PI. In the Pre-processing step, domain-keys with too few values are filtered. In the Seed rules step, different rules are applied to the values in each domain-key; domain-keys with too few matching values are filtered out in the Filtering stage. Finally, in the Expansion step, the newly discovered values are shown in **bold**, and these may be used to help refine the seed rules.

To evaluate whether these domain-keys contain real user PI, several human raters[3] manually inspected each of the domain-keys. Overall, we find that only 33 (12.6%) of the domain-keys were labeled by humans as PI; examples include email addresses, ids, user interest, locations, etc. Taking a closer look at the false positives, we find that they contain values that are related to the users' activities, however, not users' sensitive PI, including dates, timestamps, click tags, referrers, etc.

Summary Applying individual statistical tests in the real *ISP traffic* results in too many false positives; applying a combination of the tests results in too few true positives. In the following section, we propose a more sophisticated technique based on the properties of values of different domain-keys we learned so far.

5. SEEDED APPROACH

This section describes the method we propose to identify PI in network traffic traces, parsing traffic data and distinguishing user PI in an automated fashion. We begin by extracting fields from the various HTTP headers in the manner described in Section 3.1.

As observed in the previous section, in the vast majority of cases, reliance on the statistics of domain-keys fails to reveal values with PI. Hence we shift our focus to the domain-key *values* and propose a novel semantically-based method which we refer to as the *seeded method*. We briefly describe our high-level approach here, and provide more details on each step. First, we have an initial pre-processing step, where we examine all domain-keys of a dataset and filter out those that do not have enough values to produce statistically meaningful results. Second, we apply a number of *seed rules* crafted to find clues of PI *directly from the values* contained in each domain-key. Using these rules, we select candidate domain-keys to be those that have a sufficient level of matches. Fourth, we extend the set of possible values to include those in the candidate domain-keys by adding the missing values into our value pool. A diagram of these four steps is presented in Figure 4.

These four steps are described in more detail in the following subsections. Many of the steps require choices of constants and parameters; when describing each of the steps, we describe our process for selecting the parameters based on observations from the *Lab traffic* and *ISP traffic* datasets.

[3]To further protect privacy of users from de-anonimization, the only human participants allowed to review the data was limited to the six authors of this paper.

5.1 Pre-processing

Our approach relies on the format of the values of different domain-keys to select domain-keys that are likely to be carrying PI. Thus, we need a large enough sample of values to be able to produce statistically significant results. To do so, we simply select a threshold n, and only consider domain-keys for which we have observed n tuples (user/value pairs). For example, $n = 5$ can either mean one unique value from each of 5 different users, or 5 different values from a single user.

When applying this pre-processing step, we naturally face a tradeoff between the potential false positives and the coverage of domain-keys where we have few data points. Thus, we briefly explore the *coverage* of domain-keys that different choices of n provide. Using the *ISP traffic* dataset, we plot the cumulative distribution of the number of distinct values each domain-key has (we also plot the number of users and total number of tuples for comparison) in Figure 5. We observe that out of the 3.1M total domain-keys, only the top 270,756 (8.7%) "heavy hitter" domain-keys have at least 10 distinct tuples. However, these heavy hitter domain-keys in aggregate cover 90.8% of all observed tuples; thus, when applying pre-processing, we filter out a significant fraction

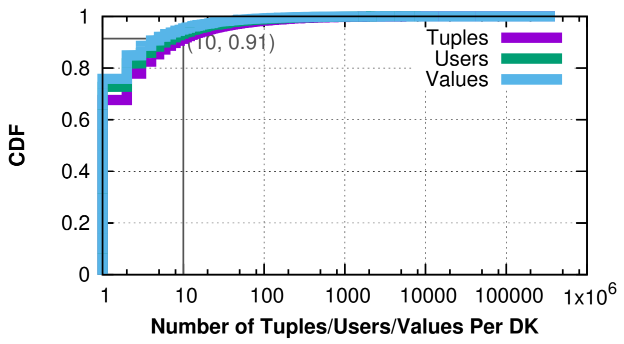

Figure 5: The cumulative distribution of the number of unique tuples, users, and distinct values for each domain-key discovered in *ISP traffic* dataset.

PI type	**Regular expression**
Age Range	`/^[0-9]{1,3}-[0-9]{1,3}$/` (where the second number is larger than the first)
Email	`/^(\w|\-|\_|\.)+\@((\w|\-|\_)+\.)+[a-zA-Z]{2,}$/`
Geo	`/^[\+\-]{0,1}\d+\.\d{4}\d+$/` (where the value is within the range of the country)
Gender	`/^[mf]$/ or /^(fe)?male$/` (or the corresponding words for male/female in local language)
Phone	`/^([+]`*code*`)?((38[{8,9}|0])|(34[{7-9}|0])|(36[6|8|0])|(33[{3-9}|0])|(32[{8,9}]))([\d]{7})$/`
Postal code	`/^\d{5}$/`

Table 3: Examples of regular expressions used for a subset of the seed rules. Some of the regular expressions require minor post-processing, such as a the "age range" PI category, where the second element of the range must be greater than the first.

of the domain-keys, but still retain the vast majority of the observed tuples in the trace.

5.2 Seed rules

We develop a list of constraints, or *seed rules*, based on the format of expected PI. For many of the different types of PI, seed rules can be expressed as simple *regular expressions*, and are sufficient to express the possible data formats. For example, in Table 3, regular expressions are sufficient to capture email addresses, genders, age ranges, geo-coordinates, postal codes, and phone numbers.[4] For some of these, some simple post-filtering is required (e.g., to express that an age range is from a lower number to a higher one).

However, other types of PI may not be as easily expressible as a regular expression. Examples of such PI include user's names, cities, and regions. To capture these, we also allow seed rules to be expressed as *dictionaries* containing lists of possible values. For example, for first names of users, we create a comprehensive list of names[5] by downloading a set of corresponding web pages with boys or girls' names from the given country. Similarly, we create a dictionary of different cities and regions in the country of interest in order to create a seed rule for the user's location.

Our dictionary-based rules do not need to be exhaustive to be effective. As we show in the next section, as long as our seed rules are sufficient to cover a significant fraction of the actual vales (in practice, we have found good performance with as low as 20% coverage), our Expansion step is able to discover the additional values as potential PI.

Lastly, we note that we limit our seed rules to the above eight PI instances simply for brevity, not because the expressiveness of the rules is limited to just these types of PI. We believe these exemplary rules are sufficient for demonstrating both the utility of our proposed method and the applicability to various other types of PI.

Of course, the seed rules that we have selected are unlikely to cover all the cases, formats, and languages; they can easily be improved and expanded, based on the input and results. Though the seed rule solution is not universally applicable, for example in the Table 3 "where the value is within the range of the country", we need to apply different things in the seed rules based on the input of dataset. However, once the seed rules are generated, they can help us in discovering domain-keys with different types of user PI in an efficient and automatic way.

5.3 Filtering domain-keys

Not all the domain-keys matching seed rules represent PI. To confirm that the domain-keys are indeed likely used as a container for transmitting PI, for each domain-key, we look at all its values, and we compute a metric *ratio matched values* in relation to each seed rule. This metric is simply number of values that match the seed rule, divided by the total number of values. When the *ratio matched values* is above a given threshold as described below, we consider that this domain-key is likely to carry the type of PI with a confidence represented by the matching value ratio.

Again, choosing an appropriate *ratio matched values* is a tradeoff, where a small threshold *ratio matched values* increases the coverage, but results in higher false positive rates. To illustrate how we choose the threshold in practice, Figure 6 plots the distribution of *ratio matched values* across domain-keys in the *ISP traffic* dataset for each of the eight seed rules. For each rule, only those domain-keys that have *at least one* matching value are considered.

We make a number of interesting observations. First, we observe that while the distribution is different for each of the seed rules, all of the rules show a "knee" at some point in the curve. In the case of email addresses, for example, 21% of domain-keys have all their values matching the rule (i.e., the ratio is 1) and over half (58%) of the domain-keys have at least 20% of their values matching the seed rule. In contrast, the name domain-key shows that the vast majority (over 90%) of domain-keys that have at least one matching value have less than 10% of all their values match. This difference in performance is due to the nature of the seed rules; for email, the regular expression is unlikely to select values that are not actually email addresses, whereas the name seed rule is only a subset of the possible user names.

To choose a good tradeoff between false positives and coverage, for each rule, we choose a *ratio matched values* threshold to the knee points of the corresponding distribution. For example, we select a threshold to be 1 for postal codes, 0.9 for geo locations, and 0.2 for the rest.

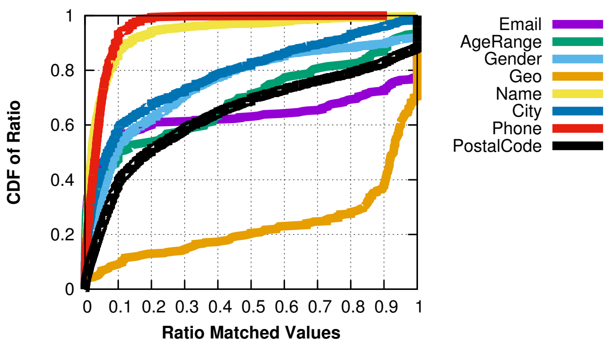

Figure 6: The cumulative distribution of *ratio matched values* for domain-keys in *ISP traffic* dataset for each of the eight different seed rules. For each rule, only those domain-keys that have *at least one* matching value are considered. Different rules show different properties, but most show a "knee" in the curve at some threshold.

[4]We note that our examples of phone numbers and postal codes use the local formats of the country where our dataset is from.

[5]`http://www.babynamespedia.com/search/m/countryname`

Domain	Key	Value
`google-analytics.com`	email	johnDoe@gmail.com
`google-analytics.com`	email	janeDoe@hotmail.com
`google-analytics.com`	email	johnDoe
`google-analytics.com`	email	janeDoe
`facebook.com`	gender	female
`facebook.com`	gender	m
`facebook.com`	gender	f
`facebook.com`	gender	1
`facebook.com`	gender	f-f
`facebook.com`	gender	f-m

Table 4: Examples of values that both match and do not match the seed rules. We observe that values that do not match the seed rules may still identify potential leaks, and may help to refine the seed rules.

5.4 Expanding candidate domain-keys

We understand our seed rules are not exhaustive, and it is challenging to develop perfect seed rules that can match all the possible formats of PI. To address this limitation, we expand the candidate values associated with selected domain-keys into our value pool, to compliment our findings. For example, if we pick 0.2 for the email case shown in Figure 5, we get a list of domain-keys with email, and each of them have at least 20% of their values matching the seed rule. When we consider the set of PI, we add the values that do not match the rule into our value pool as well. Though they do not match the exact regular expression, they may also contain user PI (e.g., the username without a domain, an email address with extra whitespace, etc).

A few examples of matching and non-matching values are presented in Table 4. The first four rows show values associated with domain-key (`google-analytics.com`, `email`). While the first two examples match the regular expression, next two rows with values "johnDoe" and "janeDoe" are truncated form of the first two email addresses used as usernames. The last six rows show values associated with the domain-key (`facebook.com`, `gender`). We observe that the first three match the seed rule, but the final three do not (although they likely contain some form of PI being conveyed by Facebook or a third-party application). In both cases, considering the values that do not match the seed rules can help to refine the seed rules, as well as present additional potential leaks to the administrator.

We note that there are a few reasons why a service would use different formats for the same key. First, the service may have different formats of values within different parts of HTTP header, such as url, referrer, and cookie. Second, some services provide APIs for the external developers, who may use formats that differ from the main service. For example, we notice that Facebook advertising API allows us to specify the key "gender" with value of 0 to target male users and value of 1 for female users. Instead, the main Facebook service uses a list of other values, such as male, female, m, and f to specify gender. Third, the values may be based on end user input, which is not always well-formatted.

6. EVALUATION

In this section, we apply our proposed approach to *ISP traffic* and evaluate its performance in comparison to a baseline approach. We then present interesting findings on user PI through an in-depth analysis on the discovered domain-keys.

PI Type	Dom.-keys selected	Ground truth	Coverage	Accuracy
Age Range	0	0	—	—
City	19	19	100%	100%
Email	22	22	100%	100%
Geo	34	7	0%	0%
Gender	38	14	100%	36.8%
Name	16	16	100%	100%
Phone	1	1	100%	100%
Post code	144	26	100%	18.1%
Total	274	105	100%	38.3%

Table 6: Comparison of domain-keys selected by the seeded method and ground truth in *Lab traffic* dataset.

6.1 Evaluation of seeded method

Having applied the seeded method to our datasets, we now analyze the performance of the method in terms of coverage and accuracy. To evaluate our proposed method in controlled environment with ground truth, in Section 6.1.1, we begin by analyzing results from small-scale *Lab traffic* dataset (§ 3.2). Then in § 6.1.2 through § 6.1.4, we evaluate the method on full-scale *ISP traffic* dataset (§ 3.3). Here, we first quantify *coverage* of our seeded method by comparing it to a baseline naïve method. We then further analyze the *accuracy* of our results by manually inspecting correctness on a sample of the discovered values.

6.1.1 Verification on Lab traffic *dataset*

Using the ground truth on PI we have in *Lab traffic* dataset, we measure the validity of domain-keys and their values obtained using the seeded method. Out of 20,810 tuples from 8,372 domain-keys available in the HTTP/S data, seeded rules extract 274 domain-keys as containing PI. A breakdown of domain-keys discovered by the rules, and those containing the ground truth PI leaks, is detailed in Table 6.

We make a number of interesting observations from our inspection of the results: overall, only 3.27% of all domain-keys are discovered as containing PI. Given the "needle in a haystack" observation we made in Section 4.1, it is reasonable that the selected domain-keys are only a small fraction of all domain-keys.

In a few particular cases, we find a single rule matching multiple instances (different values) of PI. For example, a domain-key (`cm.g.doubleclick.net`, `ct`) matching the City rule has a list of different values: Boston, Beijing, Seoul, Shanghai. While we consider Boston as the most "correct" answer (as it is the current residential city of the user), the rule found other cities that she visited in the past. At the same time, the existence of multiple PI instances supports the utility of the candidate value expansion in Section 5.4 as it accommodates broader range of the candidate values into user PI pool.

Occasionally, we found the expansion stage of the approach to introduce a few false negative values. For example, we find an extracted domain-key (`graph.facebook.com`, `name`) containing two application names, "Pinterest" and "iHeartRadio", along with the target user name "Yabing Liu" (one of the authors). The application names were found in the domain-key because the user was logging on to the applications through her Facebook account.

Overall, our evaluation on the *Lab traffic* dataset suggests that our approach is able to identify domain-keys that carry PI with high coverage. Moreover, we observed a significant

Type	Key semantics	True positive ex.	False positive ex.
Age Range	substring of age	age, age_range,	message, language, pagesize
City	substr. of city, citta, area, state, provincia, loc, region, where	city, location	client_state, locale
Email	substr. of email, user, account, login, logon, or equal to "e"	email, login_email	user_segment, login_password
Gender	substr. of gen, gnd, gdr, ycg, sex, or equal to "g"	gender, user_sex	pagename, useragent
Geo(Lat/Lon)	substr. of lat, lon, lng, geo	latitude, logitute	platform, relation
Name	substr. of name, nome, pers, author	name, person	app_name, slotname, listname
Phone	substr. of phone, pid, or equal to "p"	phone, pid	appid, sdkapid
Postal Code	substr. of zip, geo	zipcode, geo	gzip, gzipbyteencoding

Table 5: Examples of both true and false positives when using the key semantic method.

fraction of these (44%) were observed in unencrypted HTTP traffic, including the user's name, gender, city, and postal code. We now explore running our approach on the much larger *ISP traffic* dataset.

6.1.2 Improvement over a baseline approach

From this section on, we evaluate our method on a larger, realistic dataset of *ISP traffic*. To understand the improvement of our seeded method over baseline results, we begin this section by running a comparative study of our seeded method against a naïve, key-semantic based approach that analyzes *key names* (as opposed to *values* as in our seeded approach). Then we provide our reasoning on why seeded method constantly outperforms the baseline method without even considering the semantics of the keys.

Baseline key-semantics approach We create a strawman approach based on *key semantics* in which we leverage common intuition that keys that are suggestive of PI (e.g., keys named "email", etc) would carry the PI as their values. In other words, if the majority of keys containing PI have dictionary words such as "email", "gender", or "name", the baseline approach should be able to collect all such domain-keys with PI. We later compare the results of our seeded approach against that of the simple baseline approach, and quantify their gap in terms of coverage.

To create the baseline approach, we select a list of lexicons of each PI category. Table 5 presents the selected key terms for each of the eight PI categories we consider, along with some examples of true and false positives. Overall, we find 20,565 of our domain-keys match at least one of the rules (a breakdown is shown in Table 7, column six).

Performance comparison Table 7 presents a detailed comparison of the coverage of our seeded method to the coverage of the baseline method. In particular, comparing columns four and six shows the total number of domain-keys selected by the two methods in each category, respectively. We immediately observe that the baseline method finds many more potential domain-keys containing PI (in some cases, up to three orders-of-magnitude more). However, this result may be somewhat misleading, as these are only *potential* domain-keys that may or may not carry PI (e.g., the domain-key (`facebook.com`, `function-name`) would be selected by the baseline method, as it has `name` in the key name). Thus, to fairly compare the two methods, we need to estimate their false positive rate.

6.1.3 Accuracy analysis on samples of results

As we do not have the ground truth PI of the *ISP traffic* dataset users (*i.e.*, customers of the ISP), we instead rely on multiple human raters to identify potential PI. Due to the size of the dataset, we use sampling to make the evaluation tractable.

We first describe how we evaluate the accuracy of our seeded method. To evaluate the false positives, we began by choosing up to 170 random domain-keys from each of the eight PI category from the final output of the seeded method, hence choosing 873 flagged domain-keys in total (31.3% sampling rate out of the final 2,789 domain-keys). Similarly, to measure false negatives, we randomly sampled 1,000 domain-keys that the seeded method did not chose (0.032% sampling rate out of 3,110,907 non-flagged domain-keys). To measure the accuracy of the baseline key-semantic method, we take a similar approach. We select up to 25 random domain-keys from the domain-keys identified by the baseline method in each category of PI.

For each of the domain-keys tested, the six human raters either labeled positive (*i.e.*, the type of the PI), negative, and neutral (*i.e.*, don't know) for the question of whether the domain-keys contain PI or not. In all tests, for each domain-key, we then chose 10 values randomly to present to the human rater, alongside the domain and key name. Each domain-key was reviewed by three raters, allowing us to run majority voting when labels disagreed.

Seeded method Overall, we find that 221 out of the 873 domain-keys flagged by the seeded method to be false positives (*i.e.*, the human raters indicated not containing PI), resulting in a false positive rate of 25.3% (with the corresponding confidence interval from 22.4% to 28.2%). However, we notice that the false positive rate for the Postal Code rule is as high as 91.6%, which means our seeded rule does not work well in identifying only domain-keys that contain postal codes (instead, it captures many additional domain-keys as well). As detailed in Section 5.2, a seed rule generalizes particular patterns embedded in PI. In the case of postal codes, the seed rule of `/^\d{5}$/` is not specific enough to separate the PI from random five digit numbers. For this reason, we filter out postal code from our ruleset, and obtain a resulting false positive rate of 13.6% (with the corresponding confidence interval from 11.3% to 15.9%).

We also find that 27 out of the 1,000 non-flagged domain-keys to be identified by the human raters as containing PI, thereby representing a false negative rate of 2.7% (with the corresponding confidence interval from 1.7% to 3.7%).

Baseline method For the baseline key-semantic method, we observe that the human raters found 179 of the 200 domain-keys flagged by the baseline to be false positives, resulting in an extremely high false positive rate of 89.5%.

Overall, the survey finds the false positive rate to be high for the seeded method, and unacceptably high for the baseline method. However, we believe that the 13.6% false positive rate of our seeded method is acceptable due to three reasons: First, PI is rare, and it is difficult to find the cor-

	Seeded				Baseline		Comparison		
PI Type	# DKs	Threshold	# DKs above threshold	False positives	# DKs	False positives	Common DKs	Unique to Seeded	Unique to Baseline
Age Range	199	0.2	17	0.0%	3,729	88.0%	0	17	3,729
City	1,402	0.2	465	8.8%	3,191	76.0%	241	224	2,948
Email	382	0.2	154	3.9%	3,253	76.0%	82	72	3,171
Gender	2,041	0.2	147	0.0%	1,358	100.0%	140	7	1,218
Geo (lat/lon)	341	0.9	214	10.0%	1,986	88.0%	110	104	1,876
Name	1,549	0.2	100	52.5%	2,142	92.0%	22	78	2,120
Phone	993	0.2	11	90.9%	3,864	100.0%	0	11	3,864
Postal Code	13,449	1	1,681	91.6%	1,044	92.3%	22	1,659	1,022
Total	20,356	—	2,789	13.6% (25.3%)	20,565	89.5%	617	2,172	19,948

Table 7: Comparison between our proposed seeded method and baseline key-semantic method. The seeded method has a dramatically lower false positive rate (13.6%, when disregarding postal code) than the baseline method.

rect PI from a huge dataset without any ground truth. Second, the false positive rate is tunable by selecting a different threshold; we opted for increased coverage in these experiments, and could easily lower our false positive rate at a cost of increased false negatives (currently 2.7%). Third, we observe that it is difficult even for humans to agree what is PI and what is not. For example, among the 873 labeled domain-keys from seeded method, only on 81% of them did the human raters agree: on 18% one rater disagreed, and 1%, all disagreed. The upshot is that our method is able to focus quickly on the small subset of domain-keys that potentially leak PI.

6.1.4 *Exploring higher accuracy of seeded method*

While the seeded method only focuses on the *syntax of values* (via regular expressions and dictionaries), it captures many more DKs with PI than the baseline approach focusing on the *semantics of keys*. To better understand the reason for the large difference, we take an in-depth look at the key semantics of the domain-keys found by the seeded method.

We first analyze domain-keys the baseline method selected but our seeded method did not. Column 8 of the Table 7 shows the number of domain-keys overlapping between the two methods. Compared to column 10 (*i.e.*, total number of domain-keys selected by the baseline method), we observe that only a small fraction (3.1%) of the domain-keys selected by the baseline method are indeed included in the final results of the seeded method, suggesting that the vast majority of services do not name their keys semantically accurately. For instance, a key term "name" does not always draw terms relevant to user names we target. Instead, as exemplified in Table 5, it erroneously includes mobile app names, names of data slot, and a binary representation of existence of a name.

We then analyze domain-keys the seeded method selected but the baseline method did not. As the small difference between column 4 and column 8 of Table 7 suggests, the majority of DKs seeded method selects are included in the selection of baseline method as well.

From the total of 20,356 domain-keys that match seed rules, the "ratio" thresholds we impose in Section 5.3 selects only 2,789 of them (13%) as the rest do not contain enough number of valid values. Figure 7 further explains this using the email category as an example. The curve shows the cumulative distribution of email domain-keys ordered by the fraction (ratio) of values matching our seed rule. Out of the total of 382 domain-keys that match seed rules, the seeded method selects 154 of them after imposing the preset threshold ratio of 0.2. Upon our manual inspection on the 228 domain-keys that were left out, many of them contained values irrelevant to emails. For example, in domain-key (`static.ak.connect.facebok.com`, `email`), binary tags of 0 and 1 are used for its value, possibly encoding the existence of emails. In domain-key (`adserver.adtech.de`, `city`), indexes of cities are used, which we are unable to decipher without knowing its indexing mechanism.

In summary, we observed important shortcomings of the baseline key-semantic method to be applicable for automatic discovery of PI: sensitivity to selection of input domain-key categories and key terms, inability to discern domain-keys containing irrelevant values, and overly high false positives due to limited expressiveness of key terms. Our proposed seeded method, on the other hand, is deemed to be much more robust to the above issues.

6.2 Analysis on services leaking PI

Using the results of our seeded method, we now analyze the 2,789 domain-keys that contain user PI. In particular we aim to answer the following questions: (i) are there any specific types of user PI collected by particular kind of services and (ii) verify the existence of abusive domains that collect a broad range of user PI.

To this end, we focus on six types of PI: age range, city, email, gender, geo location, and name. For each root domain labeled as positive by our seeded method, we assign one of the following eleven service categories by manual inspection: Advertisement (*e.g.*, `ads.bluelithium.com`), Ad-

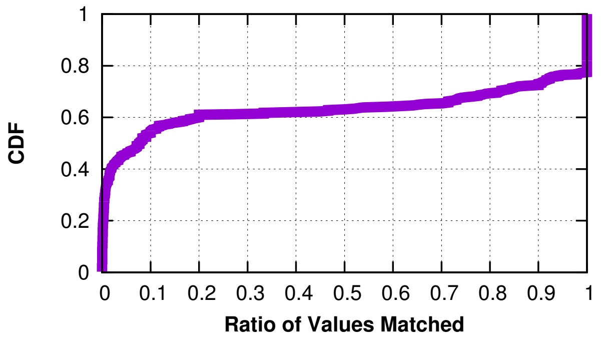

Figure 7: CDF of the ratio of values matched for email domain-keys discovered via seed rule.

ware[6] (`citibank.0009.ws`), Content Distribution Network (CDN) (`img-cdn.mediaplex.com`), User-tracking (`pixel.quantserve.com`), along with more familiar service categories of Game, OSN, Search-engines, Web-portals, and Adult. As an example of a domain with multiple services, we add Google in the category.

Figure 8 shows a heat map based on the probability density distribution of PI types by category. Comparing popularity of PI used across services, we notice that residential city turns out to be the most prevalently leaked PI (30.7%) followed by fine-grained geo-coordinates (15.2%) and gender information (10%).

PI leakage per service category With respect to question (i), we analyze the service categories that leak PI. CDNs highly benefit from spatial locality of cached data to users. Therefore, knowing user location is one of their primary interests. As shown in Figure 8, there is a high correlation between CDN and city. Similarly, for search engines, portals, and ad services, to which providing local information to users is also important, we observe high correlation to city and geo-location.

In contrary to the majority of the Internet service categories that exhibit high correlation to location information, OSNs show very low correlation to city and geo-location; they have comparatively high correlation with emails, gender, and age range. We speculate that this is due to the online-nature of the OSNs which weighs more on the knowledge of age and gender groups rather than physical locations.

In the case of adult services, knowledge on the age range and gender of users is deemed to be important as they may provide age-restricted, gender focused contents. From tracking category, we notice that some user-tracking web bugs, which are supposedly used for aggregated web analytics, can track and identify users by their email addresses.

PI leakage per service domain With respect to question (ii), we analyze the types of user PI leaked by different domains and identify domains more prone to leak user PI. Out of 588 different domains, we find 489 (83.1%) of them only have one type of PI leaked. 79 (13.4%) of them have two types of PI leaked, and overall, 20 (3.4%) of the domains have more than two types of PI. Interestingly, one domain in user-tracking category collects five types of PI. A cursory investigation reveals that this domain is identified by users as an invasive service and sometimes associated to spyware. From our traffic trace, we confirm similar suspicious behavior, as we observe the domain collecting email, name, gender, city, and location-related information, such as geographical coordinates. Among other examples of domains collecting above average amount of PI, we find large Internet service companies with ad services such as Google and Yahoo: `google.com` contains an average of 5 PI types per user, `google-analytics.com` with 4, `doubleclick.net` 6, and `yahoo.com` 5.

7. RELATED WORK

We now discuss the existing client-side privacy preserving tools, as well as related studies on measuring the personal information and privacy leakage.

[6]Different from advertisement services, we categorize adware as domains known to distribute undesired ads by means of phishing or malware.

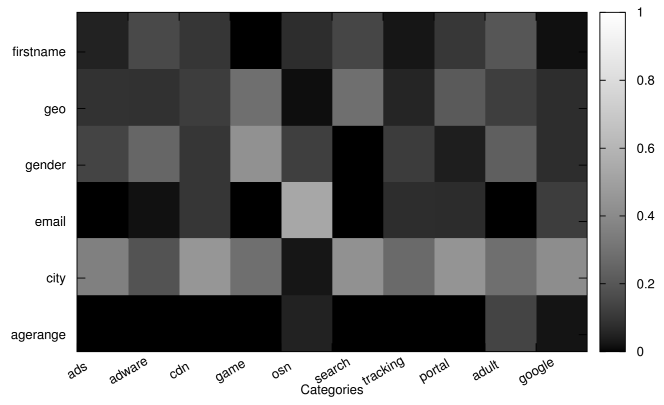

Figure 8: Heat map of PI leaked by categories of domains.

Measuring information leakage A number of projects [8–12] have examined privacy leaks by web sites, typically focusing on OSNs. By creating fake accounts on OSNs [8, 9, 11, 12] and on mobile services [10] and tracing all requests the browser makes, researchers were able to further examine how different information flows to third parties. In brief, these papers find that all OSNs exhibit some leakage of private information to third parties, typically via Request URIs and HTTP headers (e.g., `Host`, `User-Agent`, `Referrer`, and `Cookie`). These approaches are largely complimentary to the work presented in this paper, as we aim to develop a system that is able to detect leaks of (unknown) user data in a live network.

Client-side tools Researchers have developed client-side techniques such as browser extensions or add-ons, such as Adblock Plus [1, 2], RequestPolicy [19], NoScript [16], and NoTrace [17]. They have been widely used by many users, and also provide a way for researchers to measure the PI transmitted in the traffic. All the analyzed tools rely on standard blacklist/whitelist mechanisms of privacy protection. For example, AdBlock Plus is structured as a blacklist, blocking requests to a pre-defined list of advertisements and malware domains. Compared to our approach, these client-side tools offer users greater control over the process of blocking PI leaks, but face additional challenges obtaining large-scale deployment.

Recently, Ren et al. [18] have developed Recon, a client-side tool designed to capture mobile device traffic using a VPN and middlebox. Their system shares many of the same goals as ours, and the approach is largely complementary (the domain-keys that we identify as carrying PI can be leveraged by Recon as ones to flag, and their approach may offer visibility into HTTPS traffic if the user so chooses).

Static data flow analysis Static analysis aims to measure privacy leaks via the network between user applications and different web sites or mobile services. For example, PiOS [3] uses program slicing to detect privacy leaks in iOS apps. The use of static analysis enables exploring broad execution paths including infeasible ones. Similarly, Unsafe [7] focused on the advertising libraries the mobile apps contain. While static analysis has the potential to detect leaks before they occur, it often requires access to the application source code and can only feasibly be run on a small set of applications or web sites. In contrast, our approach only requires access to the network traffic, which can often be accomplished via a router tap.

Dynamic data flow analysis Different from the static analysis, dynamic analysis runs in real-time as a user executes applications. The major advantage of dynamic analysis is that, as users can provide relevance feedbacks to its false alerts, it can reduce false positives at runtime. For example, TaintDroid [4] used lightweight dynamic taint analysis built into modified Android middleware; the system alerts the user to the presence and nature of the leak in the whole apps. Similarly, Vision [6] directly instrumented the smartphone platform and tracked information flow at runtime. Leveraging unique Android execution model to reduce the search space, AppIntent [22] automatically presents a human analyst the UI manipulations that leads to the sensitive data transmission.

While these dynamic analyses can precisely pinpoint leaks from the devices they are installed, the intrinsic cost of their deployment (*e.g.*, having to install browser add-ons or custom Android builds) makes them difficult to deploy to a large userbase. Instead, integrating these data flow monitoring techniques to our approach which unobtrusively considers the network traffic at large, could lead to a more comprehensive solution to PI leakage detection (i.e., using static/dynamic analysis to identify additional potential keys of interest).

8. IMPACT AND LIMITATIONS

The method presented in this paper automatically locates personal information (PI) embedded in network traffic to Internet services. By detecting a limited number of instances of eight types of seed personal information the approach initially identifies a limited number of "containers" of personal information in terms of keys used by a specific service (or domain), hence domain-keys. Then, coverage is extended by inferring additional containers by analogy with the seeded ones. Our evaluation on a large-scale traffic trace collected on the network of a residential service provider shows that our proposed approach is able to locate the rare domain-keys that serve as containers for PI with low false negatives (2.7%) and acceptable false positives (13.6%).

Selecting thresholds The intervention of an analyst performing manual inspection on the PI identified by the methodology might be required in order to optimally set the thresholds at the basis of the operations of the proposed method. One of our future work directions will focus on automating threshold setting. For example, a "state space exploration" of the potential settings could be conducted to then pick the ones that perform best on a number of tests. This is not trivial as it might require a large amount of computation resources and carries the risk of overfitting. In the meantime, the need for having a human in the loop does not undermine the high value of the approach since it brings to the analyst attention only a small fraction of the large quantity of information flowing through the Internet. Without the support of this method, manually inspecting the full traffic has widely proven unfeasible.

Encrypted traffic Since the methodology here presented relies on inspection of data exchanged by Internet services its applicability can be limited when such data is encrypted or obfuscated. While the latter is a complexity that most service providers currently do not want to incur, in the last years the fraction of web traffic being encrypted (i.e., using HTTPS) has increased significantly. This does not undermine the relevance of the solution as it has several application areas of significant impact where the effectiveness of the methodology is not affected by the deployment of HTTPS.

PI leakage protection A service provider (being it an Internet, cloud, or cellular service provider) could offer to its customers a service to audit their traffic for PI that is potentially collected by third parties because it is being sent through the network in clear text form. This application does not require visibility into PI sent over encrypted connections as it does not represent a leakage in this context.

Enterprise protection against information leakage The proposed approach can be deployed by a company wanting to detect intentional and unintentional leakage of information critical to their business, which includes employees' PI. In this scenario the company will enforce all HTTP(S) traffic to go through a corporate (man-in-the-middle) proxy that terminates SSL sessions, thus acquiring visibility into encrypted traffic (in fact, there exist companies that use such proxies today). Such an approach requires applications (e.g., web browsers) to accept as legitimate the certificates that the proxy generates, signs with its own certificate, and presents in the initial SSL negotiation phase. This can be achieved by pre-loading the proxy certificate into corporate PCs as the certificate of a trusted certification authority. Employees that want to use their own devices to access the Internet through the corporate network are required to install the proxy certificate or manually accept the certificates offered when SSL sessions to new servers are negotiated.

PI disclosure assessment and control A provider could offer a service that identifies PI being embedded in the traffic of a user, both protected (i.e., through HTTPS) and unprotected. As it is common in many other contexts, such as online social networks, free e-mail services, etc., customers interested in the service are willing to grant the provider with visibility into their encrypted traffic. This can be achieved by the user either loading the certificate of a man-in-the-middle proxy operated by the service provider in the trusted certification authority repository, or installing a module (e.g., a browser plugin) that analyses the traffic before being encrypted [13]. Our technique can then be applied within the proxy or the plugin.

Future work As part of our future work we plan to extend the methodology to differentiate between PI the user has intentionally shared and other that was not, which is particularly valuable in the last application scenario listed above. One possible way to do this is by occasionally surveying users about observed leaks, learn of a few (in)voluntary ones, and extend the knowledge across users and web services. Eventually, we aim to build a system capable of informing users when personal information is being leaked without an explicit act on their side, so that they can decide whether the leak should be allowed or blocked (e.g., by substituting information with placeholders [18]).

Acknowledgements

We thank the anonymous reviewers and our shepherd, Ben Zhao, for their helpful comments. This research was supported in part by NSF grants CNS-1054233, CNS-1409191, CNS-1319019, and CNS-1421444. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

9. REFERENCES

[1] AdBlock Plus. `https://adblockplus.org/`.

[2] Adblock Plus : Statistics for Adblock Plus. `http://bit.ly/1OWEytx`.

[3] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. Pios: Detecting privacy leaks in ios applications. In *NDSS*, 2011.

[4] W. Enck, P. Gilbert, 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. In *OSDI*, 2010.

[5] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *SIGCOMM IMC*, 2010.

[6] P. Gilbert, B.-G. Chun, L. P. Cox, and J. Jung. Vision: Automated security validation of mobile apps at app markets. In *MCS*, 2011.

[7] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *WISEC*, 2012.

[8] B. Krishnamurthy. Privacy and online social networks: Can colorless green ideas sleep furiously? In *IEEE Security and Privacy*, 2013.

[9] B. Krishnamurthy and C. Wills. Privacy diffusion on the web: a longitudinal perspective. In *WWW*, 2009.

[10] B. Krishnamurthy and C. E. Wills. On the Leakage of Personally Identifiable Information Via Online Social Networks. In *WOSN*, 2009.

[11] B. Krishnamurthy and C. E. Wills. Privacy leakage in mobile online social networks. In *WOSN*, 2010.

[12] D. Malandrino, A. Petta, V. Scarano, L. Serra, R. Spinelli, and B. Krishnamurthy. Privacy awareness about information leakage: Who knows what about me? In *WPES*, 2013.

[13] H. Metwalley, S. Traverso, M. Mellia, S. Miskovic, and M. Baldi. CrowdSurf: Empowering Informed Choices in the Web. *CCR*, 45(4), 2015.

[14] mitmproxy: a man-in-the-middle proxy. `http://mitmproxy.org/`.

[15] mitmproxy Installation. `http://mitmproxy.org/doc/install.html`.

[16] NoScript. `http://noscript.net/`.

[17] NoTrace. `http://www.isislab.it/projects/NoTrace/`.

[18] J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. Choffnes. Recon: Revealing and controlling privacy leaks in mobile network traffic. `http://arxiv.org/abs/1507.00255`.

[19] RequestPolicy. `https://www.requestpolicy.com/`.

[20] A. Sapio, Y. Liao, M. Baldi, G. Ranjan, F. Risso, A. Tongaonkar, R. Torres, and A. Nucci. Per-user policy enforcement on mobile apps through network functions virtualization. In *MobiArch*, 2014.

[21] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *IMC*, 2011.

[22] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning, and X. S. Wang. Appintent: analyzing sensitive data transmission in android for privacy leakage detection. In *CCS*, 2013.