2018 IEEE Symposium on Security and Privacy

Privacy Risks with Facebook's PII-based Targeting: Auditing a Data Broker's Advertising Interface

Giridhari Venkatadri[†], Athanasios Andreou[§], Yabing Liu[†],
Alan Mislove[†], Krishna P. Gummadi[‡], Patrick Loiseau[*‡], Oana Goga[*]
[†]Northeastern University [§]EURECOM [‡]MPI-SWS [*]Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG

***Abstract*—Sites like Facebook and Google now serve as de facto data brokers, aggregating data on users for the purpose of implementing powerful advertising platforms. Historically, these services allowed advertisers to select which users see their ads via *targeting attributes*. Recently, most advertising platforms have begun allowing advertisers to target users *directly* by uploading the personal information of the users who they wish to advertise to (e.g., their names, email addresses, phone numbers, etc.); these services are often known as *custom audiences*. Custom audiences effectively represent powerful *linking* mechanisms, allowing advertisers to leverage any PII (e.g., from customer data, public records, etc.) to target users.**

In this paper, we focus on Facebook's custom audience implementation and demonstrate attacks that allow an adversary to exploit the interface to infer users' PII as well as to infer their activity. Specifically, we show how the adversary can infer users' full phone numbers knowing just their email address, determine whether a particular user visited a website, and de-anonymize all the visitors to a website by inferring their phone numbers en masse. These attacks can be conducted without any interaction with the victim(s), cannot be detected by the victim(s), and do not require the adversary to spend money or actually place an ad. We propose a simple and effective fix to the attacks based on reworking the way Facebook de-duplicates uploaded information. Facebook's security team acknowledged the vulnerability and has put into place a fix that is a variant of the fix we propose. Overall, our results indicate that advertising platforms need to carefully consider the privacy implications of their interfaces.

I. Introduction

Data brokers are businesses whose revenue model revolves around aggregating information about individuals from a variety of public and private sources. Traditional data brokers include Acxiom, Datalogix, and Equifax, who sell access to the collected data to third parties, including advertisers, marketers, and political campaigns. Recently, online services like Google and Facebook have become *de facto* data brokers: while they do not typically sell access to the collected data directly, they instead use the collected data to build powerful advertising services that have data on billions of users worldwide. Consequently, any breaches or hacks potentially threaten the privacy of large numbers of users, making them vulnerable to potential fraud, harassment, and identity theft attacks [6].

These new data brokers provide interfaces to their advertising services that enable advertisers to target their ads to users with specific attributes (e.g., all 35-year-old males living in Detroit); these groups are referred to as audiences. The interfaces provide advertisers with basic statistics about their selected audience, including an estimate of its size. In ground-breaking work in 2011, Korolova [18] demonstrated that malicious Facebook advertisers could select attributes that are "microtargeted", or chosen so that they are satisfied only by a single user. As a result, Korolova was able to use the audience size statistics to infer users' demographic information that was set to be private. In response, Facebook disallowed microtargeting, placing a minimum size on audiences.

Recently, data brokers such as Facebook [42] and Google [1] have introduced a new feature on their advertising interfaces: *custom audiences*. Instead of creating audiences based on user attributes, advertisers can now upload personally identifying information (PII) about specific users; the platform then locates matching accounts and creates an audience consisting of only these users. The advertiser can then use this audience when placing ads, thereby showing their ads only to the specific users whose information they uploaded. For example, a small business may know the names and addresses of its customers; using custom audiences, the business can upload this information to Facebook, and then target these users with advertising directly. The custom audience feature has proven popular with advertisers: it allows them to directly select *the users* to whom their ad is shown, as opposed to only selecting *the attributes of the users* [42], [28].

At its core, the custom audience feature is a *linking* mechanism, enabling advertisers to link various forms of user PII that they have collected to the information collected by the advertising platform. Advertisers come with various pieces of user PII—they may have the email addresses of some people, the names and addresses of others, and the phone numbers of a few more—and the platform links all of these disparate pieces of PII to the users' accounts. Unfortunately, if not implemented carefully, the custom audience feature can inadvertently leak bits of information about users to advertisers.

In this paper, we focus on auditing Facebook's advertising interface, as Facebook's advertising service is one of the most mature and well-used advertising platforms. Indeed, we show that Facebook's advertising interface leaks sensitive personal information about users by reporting coarse-grained information about the size of a custom audience. For example, malicious Facebook advertisers can infer the phone number of a user given only their email address, or they can infer whether a given user has visited a webpage the advertiser controls. None of these attacks require any interaction with the victim, none can be detected by the victim, and all can be performed without actually placing any ads or paying any

DOI 10.1109/SP.2018.00014

money. These attacks can be particularly devastating for user privacy: for example, they enable malicious users to infer the phone numbers of celebrities or politicians [31], [25], allow oppressive governments to identify and intimidate citizens who dissent online [30], enable adversaries to easily identify users' mobile numbers for purposes of "phone porting" attacks [27], and allow website operators to de-anonymize users who visit websites that may contain embarrassing or censored content.

To develop our attacks, we carefully study the features of the Facebook advertising interface. The attacks we present are enabled by two characteristics of this interface that we discover: *First*, we show that the size statistics reported by the interface are obfuscated using *rounding*; this enables us to create audiences that negate the effect of rounding by having a size that falls exactly on the rounding threshold. *Second*, we demonstrate that the interface *de-duplicates* multiple PII records that refer to the same user when reporting the size statistics; combined with the previous observation, we are able to determine whether two pieces of PII refer to the same user.

Overall, our paper makes four contributions:

- We characterize the Facebook custom advertising interface, revealing the two characteristics that enable our attacks.
- We show how an adversary can abuse the custom audience linking mechanism to infer other PII of a victim, knowing only basic information about the victim. This attack, for example, allows an adversary to infer the phone number or name/location of a victim, given their email address.
- We demonstrate how the adversary can use Facebook's third-party tracking Javascript to de-anonymize visitors to their website (e.g., by inferring visitors' phone numbers).
- We propose a mitigation that modifies how de-duplication is implemented. In brief, instead of de-duplicating at the *user* level, Facebook should de-duplicate at the *PII* level.

Ethics Throughout this study, we have ensured that all of our experiments meet community ethical standards. *First*, all of the input voter records that we used as a basis for creating threshold audiences are available as public records. *Second*, we did not collect or obtain the personal information of any Facebook users; our small set of users only provided us with their email addresses and phone numbers for validation. Our attacks did not interact with their accounts in any way, reveal any additional information about them that we did not already know, or provide to Facebook any information about them that Facebook did not already have. *Third*, we have responsibly disclosed all of the attacks to Facebook, providing them with early drafts of this submission. Their security team has acknowledged the vulnerability and has put into place a fix that is a variant of the fix we propose in Section VI.

II. Background

Most of the popular web and mobile application-based services today are funded by advertising, where the end users are provided the service for free in exchange for being shown ads.

Site	Name	Email	Phone number	City or ZIP	State or Province	Birthday, Gender	Employer	Site user ID	Mobile advertiser ID	Min. Size
Facebook	✓	✓	✓	✓	✓	✓	✗	✓	✓	20
Instagram	✓	✓	✓	✓	✓	✓	✗	✓	✓	20
Twitter	✗	✓	✓	✗	✗	✗	✗	✓	✓	500
Google	✓	✓	✓	✓	✗	✗	✗	✓	✓	1,000
Pinterest	✗	✓	✗	✗	✗	✗	✗	✗	✓	100
LinkedIn	✗	✓	✗	✗	✗	✗	✓	✗	✓	100

TABLE I: User attributes that advertisers can upload to create custom audiences in various advertising platforms. Also shown is the minimum custom audience size that the sites allow.

Most online advertising platforms are implemented using auctions, where advertisers bid on keywords or search terms in the traditional search-based advertising platforms (e.g., Google, Yahoo, Bing); or user demographics, interests, behaviors, and other user information in online social network advertising platforms (e.g., Facebook, Twitter, Pinterest). Whenever a user views a page, the platform runs an auction and displays the ads requested by the winning advertiser(s).

On most online social media advertising platforms, advertisers are allowed to create *audiences* (simply, groups of users) for convenience. For example, an advertiser may wish to advertise to a certain group of users multiple times; the advertiser is allowed to first define this audience (e.g., all users living in a certain region), and then can submit ads (and corresponding bids) to be shown to these users. Today, there are two primary ways an advertiser can create an audience:

1. Attribute-based audiences Advertisers can create audiences by specifying the properties of the users whom they wish to target. For example, advertisers can create audiences using any combination of targeting attributes, such as location (city or ZIP), age, gender, languages, likes, interests, etc.

2. PII-based audiences Recently, a new method for creating audiences has been developed: audiences created based on information that uniquely identifies specific users (i.e., using personally identifiable information, PII). These PII-based audiences come in two flavors. *First*, the advertiser can upload the PII of the users they wish to target (e.g., their email addresses, their names and cities of residence, etc.). The advertising platform then creates an audience of platform users matching the PII.[1] We refer to these audiences as *custom audiences*, with the nomenclature referring to the fact that the audiences are created in a custom fashion by the advertiser.

We provide an overview of the PII types that advertisers can upload when creating custom audiences on various platforms

[1]Most platforms provide advertisers with guarantees that the platforms themselves are not capturing customer information uploaded for custom audience creation. This is typically implemented using *hashing*, where all advertiser-provided data is hashed before being uploaded to the platform (in fact, some platforms allow advertisers to upload already-hashed data records) [41], [34], [16], [4].

in Table I. As can be observed, Facebook (and Instagram, owned by Facebook) allows the widest variety of personal information to be uploaded. However, all platforms allow advertisers to create custom audiences based on email addresses, and most allow creation based on mobile advertising IDs (essentially per-mobile-device identifiers) [40].[2] We note that the custom audience feature is given different names on different platforms: Facebook's Custom Audiences [42], [35], Twitter's Tailored Audiences [36], Google's Customer Match [1], Pinterest's Audiences [26], and LinkedIn's Audience Match [3].

Second, advertisers can create audiences based on users who interact with the advertiser's Facebook applications or (external) website. To facilitate external website tracking, the platform gives the advertiser some code (often referred to as a tracking pixel, as it was historically implemented as a one-pixel image) to include on their website; when users visit the advertiser's website, the code makes requests to the social media platform, thereby adding the user to an audience. We refer to these audiences as *tracking pixel audiences*.

It is important to underscore the distinction between PII-based audiences and traditional attribute-based audiences: with attribute-based audiences, advertisers could only specify the *attributes* of the users they wanted to target (e.g., an advertiser could create an audience of female users in Seattle). With PII-based audiences, advertisers instead specify the particular *users* they want to target, either by uploading known email addresses, names, or other personal information, or by selecting users who visited an external website the advertiser controls.

III. Facebook's PII-Based Audiences

For the remainder of the paper, we focus on Facebook's advertising platform, as it is the largest and most successful social media advertising platform. An advertiser can use a number of different features of Facebook's advertising interface when placing an ad. Only the first few of these steps are relevant to our attacks; the adversary does not need to actually place an ad. In this section, we describe the important features of Facebook's advertising interface relevant to PII-based audiences that we use throughout the paper.

A. Creating custom audiences

Advertisers create custom audiences using a web interface where they are allowed to upload 15 different types of user information; these include external unique identifiers such as Email, Phone number, and Mobile advertiser ID,[3] as well as fields related to name (First name, Last name), age (Date of birth, Year of birth, Age), sex (Gender), and location (ZIP/Postal code, City, State/Province, Country). In addition, advertisers can specify Facebook app user ID and Facebook page user ID, which are obfuscated identifiers that are generated when a user installs an advertiser's Facebook application or likes their page. Behind the scenes, Facebook first hashes advertiser-provided data before uploading it for matching, ostensibly to assure advertisers that the matching process will not reveal the advertisers' customer data [41].

We experimented with this interface to determine what set(s) of fields are *required* in order to initiate the matching process (e.g., would it be enough to simply upload a First name of "John" and match all users named John on Facebook?). Among the 15 fields, we found that only five of them can be used *alone* to create a custom audience: Email address, Phone number, Mobile advertiser ID, Facebook app user ID, and Facebook page user ID. If the advertiser wishes to use only the remaining 10 fields, they must provide First name, Last name, and one of the following sets of fields: (City, State/Province), (ZIP), or (Date of birth); they can provide additional information if desired. Finally, we note that advertisers are allowed to upload files with different information for different users (e.g., a file with the email addresses of some users, and the phone numbers of others).

It typically takes up to a few hours for Facebook to create the custom audience after the advertiser uploads the PII. As far as we can tell, there is no limit on the size of lists that can be uploaded; the audience creation process takes proportionally longer for longer lists, e.g., a few hours for 10 million records.

To prevent advertisers from targeting individual users (likely as a response to Korolova's work [18]), most platforms have policies about the minimum number of matched members for each PII-based audience to be usable. We show this information in Table I. While Facebook requires at least 20 individuals when creating a custom audience [24], most other sites, such as Google AdWords, Twitter, Pinterest, only allow advertisers to use a custom audience if it contains of hundreds of users or more. In the case of Facebook, it will actually create audiences with fewer than 20 users, but will not allow an advertiser to advertise to *only* that audience (more on this later).

Finally, once the audience is created, Facebook reports success and tells the advertiser the number of matched records (note that Facebook may not be able to match all of the PII records that were uploaded to Facebook accounts). We refer to the number of matched records as the *audience size*. The custom audience size is obfuscated, and we explore the mechanism by which it is obfuscated in the next section.

B. Creating tracking pixel audiences

Advertisers create tracking pixel audiences using a separate web interface by simply providing a name for the audience, and then including the Javascript code provided by Facebook in their external website. As with custom audiences, Facebook provides an obfuscated audience size and requires at least 20 individuals for the audience to be usable.

[2]LinkedIn is a business-centric social network and offers an entirely different feature, Employer, as their account-targeting option. LinkedIn allows advertisers to upload lists of up to 30,000 companies, and can target ads to those companies' employees. However, this feature is not self-service and involves working with the LinkedIn account team.

[3]Mobile advertiser ID represents a mobile-OS-provided identifier that is unique for each device (but can be reset by the user). This is useful for advertisers who wish to target mobile users who have already installed the advertiser's app.

C. Obtaining potential reach

The second advertising feature that we use is the potential reach. To place an ad, the advertiser can choose an existing PII-based audience to advertise to, or *combine* that audience with any other PII-based audiences that the advertiser has previously created. This avoids requiring the advertiser to submit multiple ads (one for each audience) or to re-upload the same data multiple times.

When combining audiences, Facebook allows advertisers to *include* users who appear in any number of existing PII-based audiences, and then *exclude* users who appear in any number of such audiences. When convenient, we denote these operations as union ($\cup$) and set-minus ($\setminus$). In testing the interface, we found that exclude trumps include (e.g., if a user is part of any excluded audience, the user will not be in the final combined set regardless of how many included audiences they appear on). For simplicity, we refer to the resulting set of users as the *combined audience*. It must be noted that Facebook allows the inclusion or exclusion even of audiences that have fewer than 20 users.

Once the advertiser has selected the combined audience they wish to advertise to, they then proceed to place a bid and upload the actual advertisement. As these steps are not necessary for our attacks, we do not discuss them in detail.

However, there is one feature of Facebook's interface that we do use: the number of *active* users that are in the combined audience. Facebook provides advertisers with this number, called the *potential reach*. Facebook [43] defines it as[4]

> *... the number of daily active people on Facebook that match the audience you defined through your audience targeting selections*

We refer to users who are "daily active" as *targetable*.

It is important to note how potential reach and audience size differ: the audience size only applies to PII-based audiences and includes *all* matched Facebook accounts, while the potential reach can be applied to both PII-based audiences and combinations of such audiences, but only includes "daily active" Facebook accounts. Thus, for a single audience, the potential reach is always less than or equal to the audience size. Additionally, we can obtain the potential reach for combinations of audiences via includes and excludes.

Similar to the audience size, the potential reach that Facebook reports is obfuscated; we explore this in the next section.

D. Determining audience intersection size

The final advertising feature that we utilize is the audience comparison page. Advertisers are likely to have created multiple PII-based audiences; in order to help them understand these audiences and decide which to use, Facebook's advertising interface allows advertisers to measure the overlap between different pairs of PII-based audiences that they have created. When convenient, we denote this operation as intersection ($\cap$).

[4]While this definition was provided at the time we conducted the experiments, Facebook has since changed the definition of potential reach [2]. However, the new definition still captures the notion of active users.

We interacted with this feature using some of the PII-based audiences we created, and observed it had two important characteristics. *First*, the interface only supports PII-based audiences with an audience size of at least 1,000; smaller audiences are not available when using this interface. *Second*, the intersection size shown is based on audience size, and *not* on potential reach. *Third*, this intersection size is obfuscated; we explore the obfuscation mechanism in the next section.

IV. Facebook Implementation Characteristics

Having described the key features of Facebook's interface, we now begin our analysis. We examined how the key features are implemented, and found two implementation characteristics that enable our attacks; we first describe the datasets we used for testing before describing the two characteristics. The experiments described in this section were performed between January and March 2017; Facebook's fix in response to our disclosure has since changed how some of these features work.

A. Datasets

To study how the PII-based audience interface is implemented, we need PII-based audiences to test with. To create custom audiences, we use two sources of data: *First*, we use the phone numbers and email addresses of 100 recruited friends and family members. For all of these users, we only use the data that Facebook already had listed in their accounts and was visible to us. *Second*, for experiments where we needed large numbers of records, we used public voter records from North Carolina [37]. In brief, records of registered voters in North Carolina are publicly available and contain voters' names and ZIP codes, among other fields. Obviously, not all registered voters will have Facebook accounts and both the Facebook and voter data could be out-of-date or inaccurate; thus, when uploading sets of voter records to create custom audiences, we expect that not all records would match a Facebook account.

To create a tracking pixel audience, we set up a test website with a Facebook tracking pixel created under our advertising account. We recruited 20 friends and family members with active Facebook accounts to visit the page.

B. Calculating size statistics via rounding

As noted in the previous section, there are three different mechanisms within Facebook's advertising interface that report the "size" of a PII-based audience: (1) the *audience size* representing the total number of matched accounts of a custom audience or accounts in a tracking pixel audience, (2) the *audience intersection size* representing the total number of accounts two PII-based audiences have in common, and (3) the *potential reach* representing the number of "daily active" users in a PII-based audience (or combination of such audiences). Recall that all three of these numbers were obfuscated in some way; we now determine how they are actually calculated.

Potential reach We first examine potential reach, described in Section III-C. We use our browser's developer tools to examine the network requests made by Facebook's advertising interface when we choose audiences to include or exclude. We

find that the interface makes `GET` requests to an API endpoint, which returns JSON objects that include the corresponding potential reach. We created a script to automate the process of making requests; we then used our script to collect the potential reach of over 58K different combinations of custom audiences (based on voter data) and targeting attributes, and found the output to have the following properties:

Granularity Looking at the distribution of potential reach values, we never observed any values smaller than 20, or that were not a multiple of 10; besides, we observed that larger values had few significant digits. Looking closely, we found that the values all come from the set

$$\{20, 30, ... \ 990, 1000, 1100, ... \ 10000, 11000, ..., 100000, ...\}$$

meaning they are multiples of 10 up to 1,000, multiples of 100 up to 10,000, multiples of 1,000 up to 100,000, and multiples of 10,000 beyond 100,000.

Consistency Next, we examine how consistent the potential reach values are over time. We first measure the consistency over short time scales by running 1,000 API queries back-to-back (roughly one query every second) for the same audience and targeting attributes. We repeat this experiment five different times with two different audiences, and find all the potential reach values within each run of 1,000 were exactly the same. This result shows that the audience size statistics are consistent over short intervals.

To measure consistency over longer time scales, we queried the API every five minutes for the potential reach of four different audiences, issuing 150 queries for each audience over a period of over 12 hours. The potential reach of three of the audiences remained constant, while the potential reach of the fourth audience changed slightly at one point during the 12 hour experiment and stayed at the new value. This result shows that across longer time scales, the potential reach values are largely consistent; the small changes in potential reach are consistent with Facebook's definition of potential reach as the number of *active* users in the audience.

Finally, we measure the consistency of custom audience creation by re-uploading 15 lists of PII multiple times within the same day to create custom audiences. We find that the potential reach remained consistent for 14 of the lists, with the final list changing by 10 users at one point. Again, this small change is consistent with the definition of potential reach.

Monotonicity Next, we examine whether the potential reach monotonically increases when we add more users to the list we upload. Specifically, we upload a series of lists such that each list has one record added beyond the previous one. Thus, in the corresponding series of custom audiences created, each audience would contain *at most* one additional Facebook account compared to the previous one (if Facebook could not find a match for the added record, or if the matched user was not active, we would expect the potential reach of the audience to be the same as that of the previous one).

We then study whether the corresponding potential reach values increase monotonically. For example, starting with a random sample of 300 voter records, we sequentially add 16 records to the list and upload all 17 lists. The potential reach for the initial sample was 30. This value increased to 40 as the first record was added, and then increased to 50 after the 14th record was added, showing that the potential reach statistics increase monotonically. We repeated this experiment with multiple audiences; in all cases, the same pattern held: the potential reach values increased monotonically, using the minimum granularity steps described above.

Summary Overall, our results strongly suggest that Facebook is *rounding* the raw potential reach value in steps of 10, 100, 1,000, or 10,000, depending on its magnitude.

Audience size Next, we examine the audience size, described in Sections III-A and III-B. We repeated all of the experiments that we conducted above for potential reach; for brevity, we omit the details and simply describe the inferred behavior. We observe that the audience size has values in $\{20, 30, ... \ 90, 100, 200, ...\}$, meaning they are multiples of 10 up to 100 and multiples of 100 thereafter. We also observe that audience size has greater consistency than potential reach, as we did not observe any cases where the number of matched accounts changed over short or long time periods; this makes sense, as audience size is only the number of accounts, and does not consider user activity. Finally, we observe that audience size has similar monotonicity to potential reach.

Summary We again observe that Facebook appears to be rounding the actual size of the audience in steps of 10 or 100, depending on its magnitude.

Audience intersection size Finally, we examine the characteristics of the audience intersection size, as described in Section III-D. We repeated all of the experiments that we performed on potential reach in Section IV-B on the intersection size calculation. We found the calculation to be consistent and monotonic, but with different granularity: the interface rounds the intersection size in steps of 5% of the *smaller* audience size. For example, if we intersected a audience with size 1,000 and another with size 4,500, the answer would be a multiple of 50.

Summary We observe that Facebook appears to be rounding the actual size of the intersection in steps of 5% of the smaller of the two audience sizes.

C. De-duplicating PII from the same user

Next, we describe the second implementation characteristic that enables our attacks. We found that when Facebook has an audience—or combination of audiences—that was created with different PII that both refer to the same user, Facebook only "counts" that user once when reporting the potential reach, audience size, or audience intersection size. For example, suppose that an advertiser uploaded two custom audiences: one containing the phone number of one of their customers (list A), and another containing the email address of the same customer (list B). If the advertiser then asked for the potential

	Audience Size	Potential Reach			
		Alone	$\cup C_{1p}$	$\cup C_{1e,2p}$	$\cup C_{1e,1p}$
C_{1e}	50	40	40	90	40
C_{1p}	50	40	–	90	40
$C_{1e,2p}$	100	90	–	–	90
$C_{1e,1p}$	50	40	–	–	–

TABLE II: Audience size and potential reach returned by Facebook for different audiences, and combinations of audiences (described in Section IV-C). In all cases, Facebook appears to be de-duplicating records both *within* and *across* audiences before calculating the statistics.

reach of targeting users in the combination $A \cup B$, this user would only be counted once.

To demonstrate this behavior, we describe experiments we conducted on each of the three "size" mechanisms using data from our 100 friends and family members. We randomly divided these into two distinct groups: G_1 containing 50 users, and G_2 containing the remaining 50 users.

Audience size To test how Facebook de-duplicates audiences, we created four custom audiences:

- C_{1e}, with the email addresses of all users in G_1 (50 records)
- C_{1p}, with the phone numbers of all users in G_1 (50 records)
- $C_{1e,2p}$, with the email addresses of all users in G_1 (50 records), and the phone numbers all users in G_2 (50 more records)
- $C_{1e,1p}$, with the email addresses of all users in G_1 (50 records) and the phone numbers of all users in G_1 (50 more records)

We then record the audience size that Facebook reports. As shown in the second column of Table II, we found that C_{1e} and C_{1p} have an audience size of 50, and $C_{1e,2p}$ has an audience size of 100, as expected. However, we found that $C_{1e,1p}$ has an audience size of 50 users. We repeated this experiment with different sets of users of varying sizes, and found the same behavior. Thus, this experiment shows that when creating a custom audience, PII referring to the same user—even if present in separate records in the uploaded list—is de-duplicated when creating the custom audience.

Potential reach To test how Facebook de-duplicates combinations of audiences when calculating potential reach, we request the potential reach for different combinations of the audiences we created. For example, we ask for the potential reach if we "include" audiences C_{1e} and C_{1p}. The results of this experiment are shown in Table II, and we make a number of observations. *First*, the third column of the table shows the potential reach value; this is typically lower than the audience size, as the potential reach only includes active users. *Second*, we observe that Facebook appears to be de-duplicating records, even *across* audiences. For example, when we "include" audiences C_{1e} and C_{1p}, we observe that the total potential reach is 40, even though the potential reach of each audience individually is also 40. We ran similar experiments by combining custom audiences and tracking pixel audiences, and found the same behavior.

Audience intersection size Finally, we explore whether Facebook de-duplicates records when computing the audience intersection size. To do so, we need to modify our custom audiences, as the minimum custom audience size for using the audience intersection interface is 1,000. We create "extended" versions of C_{1e}, C_{1p}, $C_{1e,2p}$, and $C_{1e,1p}$ (denoted by EC_{1e}, EC_{1p}, etc.) by adding "padding" records from the North Carolina voter list; for each audience, we repeatedly add random records and re-upload until the audience size reaches 1,000 users.

We then measure the audience intersection size between the EC_i lists; we find that Facebook again appears to be de-duplicating *across* custom audiences. For example, the intersection size between EC_{1e} and EC_{1p} is 40, even though those two custom audiences were created using different PII. We repeated this experiment with a variety of different audience sizes and found the same behavior.

In summary, Facebook appears to be de-duplicating users when calculating the audience size, potential reach, and audience intersection size, both within a single audience and across combinations or intersections of audiences.

V. Attacks

In this section, we show how Facebook's advertising interface can be abused by an adversary to link different identities belonging to the same user. We describe three distinct attacks in this section, each with different threat models.

A. De-anonymizing web visitors

We begin by demonstrating the first of our three attacks, which allows an adversary running a website to determine if a particular Facebook user has visited their site.

Threat model We assume that the adversary runs a website where they have installed a Facebook tracking pixel, that the adversary wishes to determine whether or not a particular victim Facebook user has visited this website, and that the adversary knows enough PII of their victim that they can include them in a Facebook custom audience (e.g., their email address; see Section III-A). We assume that the adversary has access to Facebook's advertising system (it is important to note that any user can sign up for Facebook advertising, and no authorization or approval process is necessary). We also assume that the victim is a "daily active" Facebook user.

Threshold audiences The insight behind the method is to create a *threshold audience*: an audience or combination of audiences where the size statistic of interest—potential reach, audience size, or audience intersection size—lies right *before* or right *after* Facebook's rounding threshold. We call an audience with a size right before the rounding threshold a *lower threshold audience*, and an audience with a size right after the rounding threshold an *upper threshold audience*. For example, if Facebook rounds to the nearest 10 using the "5-or-higher" rule, an audience with a size of 84 would be a lower threshold audience (with size reported as 80), an audience with a size of 85 would be an upper threshold audience (with size

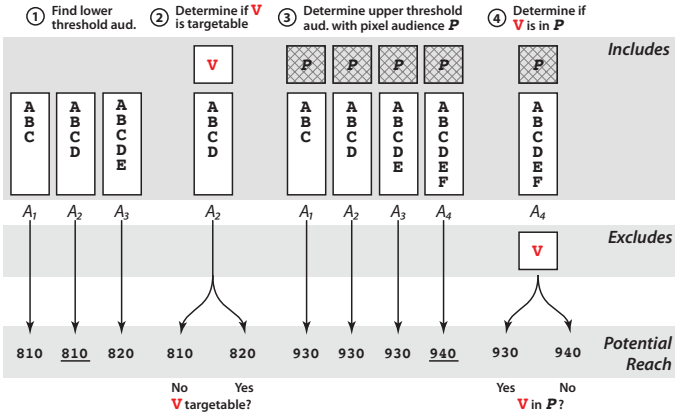

Fig. 1: Diagram of algorithm for determining if victim **V** visited the adversary's website. ① We first create custom audiences $A_1 ... A_n$ with increasing numbers of records and identify a lower threshold audience (e.g., A_2). ② We then determine if **V** is targetable by obtaining the potential reach of this audience union a custom audience V consisting only of **V** (i.e., $A_2 \cup V$). To determine whether **V** is in the tracking pixel audience ***P***, ③ we first identify an upper threshold audience containing ***P*** by successively combining ***P*** with each A_i (i.e., $A_4 \cup$***P***).; ④ we then determine whether **V** is in ***P*** by excluding V from this combination (i.e., $A_4 \cup$***P*** $\setminus V$) and seeing if the potential reach drops.

reported as 90), and an audience with a size of 86 would *not* be a threshold audience. We use threshold audiences throughout the paper to enable our attacks.

Is the victim targetable? De-anonymizing web visitors boils down to determining whether a given user is part of a tracking pixel audience. To do so, we use the potential reach statistic, which only counts targetable (daily active) users. Thus, we first need to determine whether the victim is targetable; to do so, we upload a series $\{L_1, L_2, ..., L_n\}$ of lists to Facebook, each containing one additional record beyond the previous one. To create these lists, the adversary can use any source of PII (voter records, random phone numbers, etc.). Facebook creates custom audiences $\{A_1, A_2, ..., A_n\}$ corresponding to each of these lists. We also create a custom audience V consisting only of our victim user.

For each A_i, we obtain the potential reach of that audience. We then identify a lower threshold audience as the last audience before the potential reach changes (if multiple such audiences exist, we can simply choose one of them). For example, if the potential reach of A_1 and A_2 is 810 and the potential reach of A_3 is 820, then A_2 is our lower threshold audience (as A_3 is simply A_2 with one additional user).

Once we have identified the lower threshold audience, we then ask for the potential reach of the lower threshold audience *union* V; if the potential reach changes, we know that we can target the victim, otherwise, the victim is not a "daily active" user or our external information was not sufficient to target them. Continuing the example above, we examine the potential reach of $A_2 \cup V$: if the potential reach is 820, the victim has an active Facebook account; if it is 810, they do not. This process is shown in the first two steps of Figure 1.

Is the victim in the tracking pixel audience? If the victim is targetable, we can then proceed to determine whether they are in the tracking pixel audience P (i.e., have visited the adversary's webpage). We first find an upper threshold audience for potential reach by successively combining P with the A_i lists we uploaded. For example, if the potential reach of $A_3 \cup P$ is 930 and $A_4 \cup P$ is 940, then $A_4 \cup P$ is an upper threshold audience. Finally, we take this upper threshold audience and exclude V, containing only our victim (e.g., we ask for the $A_4 \cup P \setminus V$). If the potential reach drops, we know the victim is in P; if not, we know the victim is not in P.

Evaluation To evaluate the effectiveness of this attack, we recruited 40 volunteers from our friends and family; each provided us with the email address with which they log into Facebook. Using our Facebook advertiser account, we created a Facebook tracking pixel and installed it on a webpage we control. We divided the group in half, and had 20 volunteers visit this webpage from the browser they normally use; we had the remaining 20 volunteers not visit the website.

We then conducted the attack using the 40 email addresses to determine whether they had visited our webpage. We successfully inferred that all 20 of the volunteers who did not visit our webpage were not part of the tracking pixel audience. We also successfully inferred that 18 of the volunteers who did visit the webpage were part of the tracking pixel audience. We closely investigated the two users where the attack did not succeed: we found that both had an anti-tracking browser extension installed that prevented the Facebook tracking pixel from communicating with Facebook. Thus, our attack succeeded for all users who visited our webpage and became members of the tracking pixel audience.

B. Inferring a victim's PII

We now turn to demonstrate a more powerful attack: an adversary with knowledge of only a victim's email address (or other PII) can infer a victim's other PII (e.g., their phone number, or their name and city/state). This attack also demonstrates how an attacker can use audience size statistics instead of potential reach, meaning all Facebook users are vulnerable to this attack (i.e., not just "daily active" users).

Threat model Our threat model for this attack is similar to the threat model in Section V-A, with two modifications: *First*, the adversary is not running a website, but has a piece of PII for a victim user (e.g., their email address) and wishes to infer other PII. *Second*, the victim no longer needs to be a "daily active" user; they only need to have a Facebook account.

Is the victim in a custom audience? Our attack relies on the ability to determine whether the victim is a member of a custom audience the adversary uploaded; let us call this audience A, created from PII list L. In contrast to the previous

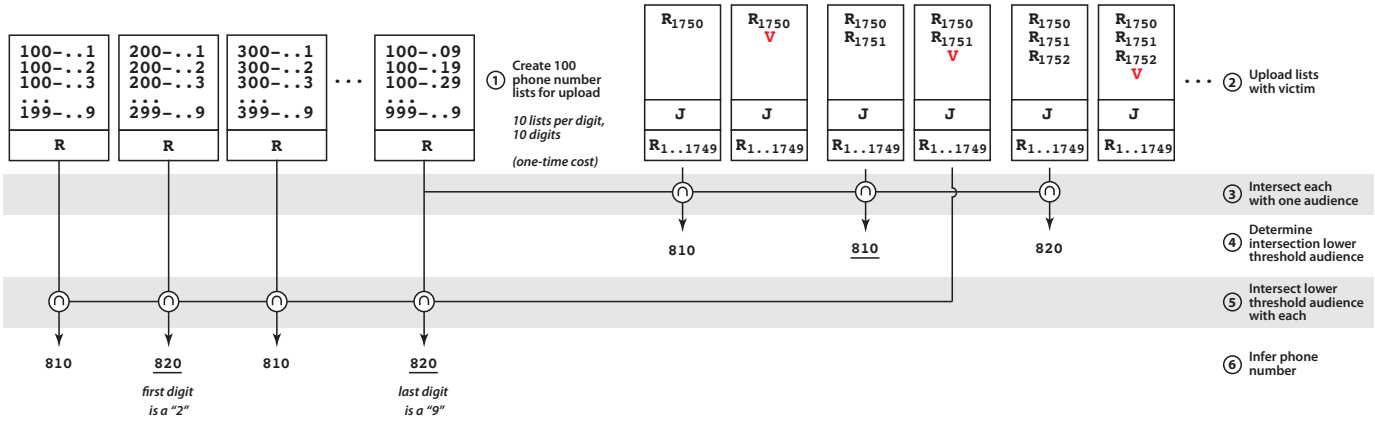

Fig. 2: Algorithm for determining phone number of the victim V, using two sets of extra records: R, a set of 1,949 records, and J, a set of 200 records. ① We upload lists of all phone numbers sharing a certain digit along with R. ② We then upload lists of 1,749 members of R with increasing additional members from R, both with and without V. ③ Next, we identify the threshold audience by intersecting each of the audiences without V with one of the phone number audiences; ④ the threshold audience is right before the intersection size changes. ⑤ We then intersect the threshold audience with V with each of the phone number audiences. ⑥ We infer each digit of V's phone number by looking for where the intersection size is higher.

attack, we aim to use the audience size statistic to do so; thus, we need a different approach than before.

To conduct the attack, we rely on the custom audience intersection size feature described in Section III-D. The naïve approach would be to upload a custom audience containing only the victim and then ask Facebook for the size of the intersection between A and this audience; if there is any intersection, we know the victim is in A (and if there is not, we know the victim is not in A). Unfortunately, there are two properties of the intersection feature that make this approach infeasible: Facebook only allows intersections of custom audiences of at least 1,000 users each, and they only show the intersection at the granularity of 5% of the smaller audience size. We therefore must go through extra steps in order to determine whether the victim is in A.

We use a set R of 1,949 records of other users' PII (e.g., voter records) and a separate set J consisting of 50 additional records (R, J, and L should not share any users). We will describe the attack first making a simplifying assumption for clarity; we will show how to remove the assumption shortly. Let us assume for the moment that all records in R and J can be matched to Facebook accounts. We then upload two custom audiences to Facebook:

- C_1: The union of R and L
- C_2: The union of R, J, and the victim's PII

We can observe that both C_1 and C_2 have at least 1,000 users (due to the size of R), and can therefore be used in the custom audience intersection feature. We can further observe that the intersection between the two audiences is determined solely by R and whether the victim is a member of A (recall that R, L, and J share no users, and that list L matches audience A). Finally, we can observe that the smaller of the two custom audiences (C_2) has exactly 2,000 users, and that, by construction, the intersection is a lower threshold audience: we know the intersection size is either $|R|$ (if the victim is not in A) or $|R|+1$ (if the victim is). Since the interface will round to the nearest 5% of 2,000 (i.e., 100) and $|R|$ was selected with 1,949 users, and since Facebook de-duplicates users when computing intersections, the custom audience intersection size that Facebook returns for C_1 and C_2 will either be 1,900 (if the victim is not in A) or 2,000 (if the victim is). Thus, we can determine whether the victim is in A, and therefore in PII list L.

Dealing with unmatched users We now show how to remove the assumption that all records in R and J will match Facebook accounts. To do so, we need to address the fact that removing this assumption will prevent us from creating a lower threshold audience by design. Instead, we must find a lower threshold audience in a similar manner to how we did before. To do so, we "hold back" 10% of R (i.e., 200 records, twice the rounding threshold[5]) and upload multiple versions of C_2; we also increase the size of J to account for the fact that not all uploaded records will match (e.g., we can create J from 200 records). Specifically, we upload a series of custom audiences C_2^i and C_3^i, for i =1749...1949, where

- C_2^i: The union of $R_{1..i}$, J, and the victim's PII
- C_3^i: The union of $R_{1..i}$ and J

where $R_{1..i}$ denotes PII records 1 through i in R. We can then use the C_3^i to first locate a lower threshold intersection audience in a similar manner as before: we intersect each C_3^i with C_1, and choose the C_3^i before the intersection size changes. Let us call this lower threshold intersection audience C_3^k. We then intersect C_1 with C_2^k (the latter being simply C_3^k with the victim's PII added in); if the intersection size is the same as before, we know that the victim is not in A. If the intersection size increases, we know the victim is in A.

[5]We choose to hold back *twice* the rounding threshold worth of users as we need to ensure that we "cross" a rounding threshold when creating the C_3^i audiences, enabling us to find a lower threshold audience

Applying the attack to phone numbers We now show how to apply this attack to infer a victim's phone number. To do so, we partition the space of all possible phone numbers into sets that all share a common digit. Specifically, we create subsets L_{ij} of all possible phone numbers, where L_{ij} is a set of phone numbers whose i-th digit has a value of j. For example, if phone numbers are eight digits, L_{12} will contain `20 00 00 00`, `20 00 00 01`, up through `29 99 99 99`. This gives us $10d$ sets, each of size 10^{d-1}, where d is the number of digits in the number.

With each of these lists, we are now ready to conduct the attack. We simply repeat the methodology proposed above, determining whether the victim is a member of each L_{ij}. Note that because J and R should never share any users with L_{ij}, the adversary should choose users for R and J from a different country than the one the victim is from. For each i, we should find exactly one j such that the victim is in L_{ij}. Figure 2 gives a diagram of this attack.

Evaluation To evaluate the effectiveness of this attack, we recruit 22 friends and family members who have Facebook accounts, from two regions: 14 from the Boston area and 8 from France. In order to infer these users' phone numbers, we create the L_{ij} lists for both regions.

For Boston, we create a total of 140 lists: two area codes (617 and 857), where phones of each area code have 7 digits we need to infer (recall that US phone numbers have the structure `XXX YYY ZZZZ` where `XXX` is the area code). Each list contains 1M phone numbers, all with a single digit in common. Each of these lists took around 30 minutes to upload.

For France, we create a total of 82 lists: French mobile phone numbers have nine digits where the first digit is either 6 or 7. We generate all 200M possible numbers starting with 6 or 7 and use this to construct our sets (and hence each L_{ij} contains 10M phone numbers); it took over four hours to upload each list. We uploaded all 82 lists of phone numbers (two to determine the first digit, and 10 for each of the remaining eight digits) over a period of a week.

It is important to note that uploading these initial lists is, by far, the most expensive part of the attack. However, the resulting audiences can be re-used to infer the phone number of any user (i.e., the L_{ij} audiences are not victim-specific).

We then conduct our attack to infer the phone numbers of all 22 users. We find that we are successfully able to infer the numbers of 11 of the 14 users in Boston, and of all 8 of the users in France. In the cases where we succeeded, we were able to infer each users' phone number in under 20 minutes. We carefully examined the three users on whom the attack failed, and we found that one user had never provided their phone number to Facebook, while the other two users had actually provided *multiple* phone numbers to Facebook. As a result, we inferred that these two users were members of multiple L_{ij} for a given digit i (e.g., we inferred that a user's second digit is both a 2 and a 5); in the Appendix, we describe and evaluate a modified approach for determining both of the underlying numbers. For the one user that had never provided their phone number, no L_{ij} matched for any digit.

C. De-anonymizing users en masse

Recall that in Section V-A, we demonstrated an attack that allowed the adversary to determine if a specific user was a member of a tracking pixel audience. Utilizing the techniques developed in the previous attack, we now present a much more powerful version of that attack: the ability to infer en-masse the PII of the members of any audience. For example, we show how this can be used to de-anonymize all visitors to a webpage: by placing Facebook's tracking pixel on the webpage, the adversary can determine the phone numbers of all members of this audience.

Threat model As in Section V-A, we assume that the adversary runs a website where they have installed a Facebook tracking pixel, and that the adversary wishes to de-anonymize all visitors to this website. We assume that the visitors are "daily active" Facebook users (those visitors who are not "daily active" are not subject to the attack). In addition, as before, we assume that the adversary has access to PII of some Facebook users—e.g., from voter records—in order to be able to create threshold audiences.

Inferring the first digit(s) Let us denote our tracking pixel audience as P. We first find an upper threshold audience containing P using the same technique as in Section V-A: we upload a series of increasing A_i audiences and identify an A_t such that $A_t \cup P$ falls just over the rounding threshold.

We then use the L_{ij} from Section V-B, where each L_{ij} contains all possible phone numbers whose i-th digit has a value of j. Specifically, we check each of the L_{1j} to determine if any of the users in P have each possible $j \in 0..9$ as the first digit. To do so, we simply exclude the L_{1j} from $A_t \cup P$, denoted $A_t \cup P \setminus L_{1j}$; if the potential reach of this combination is lower than the potential reach of $A_t \cup P$, we know at least one member of P has a phone number with a first digit of j. At this point, we have determined all of the first phone number digits (e.g., we might determine that all users in P have phone numbers starting with 2 or 8).

Inferring further digits Now, we have a set of prefixes (e.g., 2 and 8 from above), and we wish to determine the next digit(s) for each prefix (e.g., 21, 26, or 87). The natural way to do so would be to create lists L for every possible prefix (e.g., all phone numbers starting with 20, ...), but this quickly becomes unmanageable as there are 10^d possible prefixes of length d. Instead, we effectively construct these lists using the exclude feature and the L_{ij} we already uploaded. Specifically, let us denote L^c_{ij} as the *complement* of L_{ij}—all phone numbers that do *not* have j as the ith digit—we can express it as $L^c_{ij} = \cup_{k \neq j} L_{ik}$. We can then exclude L^c_{ij} from our audience to focus only on the prefix of interest.

Let us suppose we have the prefix 8 and we wish to determine all following digits for phone numbers that start with 8. We can then determine an upper threshold set containing $P \setminus L^c_{18} = P \setminus (L_{10} \cup L_{11} ... L_{17} \cup L_{19})$; let us denote this $P \cup A_t \setminus L^c_{18}$. We can then exclude each L_{2j} from this as

well and see if the potential reach drops; if so, it indicates there is at least one user with the prefix $8j$. For example, if the potential reach of $P \cup A_t \setminus (L_{18}^c \cup L_{24})$ is lower than the potential reach of $P \cup A_t \setminus L_{18}^c$, then we know there is at least one user whose phone number starts with 84.

This approach naturally generalizes to longer prefixes by excluding the complement of each prefix digit. Suppose we have a prefix of 379 and we wish to determine the set of next digit(s). We would construct an upper threshold list containing $P \setminus (L_{13}^c \cup L_{27}^c \cup L_{39}^c)$, denoted $P \cup A_t \setminus (L_{13}^c \cup L_{27}^c \cup L_{39}^c)$. We can then additionally exclude each L_{4j} to determine if any of the phone numbers starting with 379 have j as the next digit.

Recursively inferring all phone numbers Given the methodology above, it is straightforward to infer all of the phone numbers in an audience. We first infer the first digits of phone numbers corresponding to members of P. Subsequently, in iteration k, we take each phone prefix inferred from iteration $k-1$ (which would be of length $k-1$), append to it a digit $d \in \{0, 1, \cdots, 9\}$ and check whether the resulting phone prefix of length k matches at least one member of P. If so, we add the prefix of length k to the list of prefixes inferred in iteration k. This terminates by outputting all the phone numbers of members of P for which Facebook has a phone number.

Evaluation We place a Facebook pixel to track visitors to a webpage that we created specifically for this experiment, and recruit 14 friends and family from the Boston area that have Facebook accounts, and have a phone beginning with area code either 617 or 857. The volunteers were asked to visit the webpage while logged in to their Facebook account. Using the methods from section V-A with the email addresses provided by the volunteers, we found that two volunteers had installed ad-blockers and were not in the pixel audience, that another volunteer was not a "daily active" user, and a final volunteer did not have a phone number associated with their Facebook account. Thus, we expect that we should be able to infer 10 phone numbers from our pixel audience.

Our method output nine phone numbers that correctly matched the phone numbers of nine out of the 10 remaining volunteers. For the 10th user, we found that user had provided multiple phone numbers to Facebook; we describe and evaluate a modified approach for how such users can be handled in the Appendix. Our method took around one hour to do the entire search and output the phone numbers by sending queries to Facebook in a serial fashion; this time could be cut down significantly by parallelizing the queries sent to Facebook.

D. Discussion

We now discuss a few issues our attacks bring up.

Cost We first briefly quantify the cost of launching our attacks, and discuss potential optimizations to reduce this cost. The costs are summarized in Table III.

Number and size of phone number lists: If we represented the enumerated phone numbers from section V-B using a base b different from 10, we would need $b\lceil \log_b N \rceil$ lists of size $\frac{N}{b}$ each, where N is the number of possible phone numbers. For example, to reduce the size of each list (e.g., to avoid detection), the attacker could use a base of 100, which would require 400 lists of 100K numbers each for a particular US area code (compared to 70 lists of 1M numbers each, as before). Though these numbers seem large and potentially easy to detect, the attacker could partition these lists across multiple accounts to avoid detection or rate limiting (see Section VI-A).

Number and size of padding lists: In addition to the lists of enumerated phone numbers, our attacks require the uploading of lists of padding records. For our email–phone linkage attack (Section V-B), we require up to 200 lists of padding records (containing fewer than 2,000 records each) to find a threshold list, and 200 more with the victim added to each of the previous 200 lists. Alternately, by first finding the threshold list in the first round of uploads, and then adding the victim only to the threshold list, we only need one list (as opposed to 200) with the victim added. Furthermore, by searching for a threshold list using two rounds of uploads, the number of lists can be reduced to 30, where in the first round we upload lists whose sizes increase in increments of 10 to find a range of 10 consecutive sizes within which the threshold list lies. In the second round, we find the actual threshold list from within the appropriate range.

For our attacks de-anonymizing particular web visitors (Section V-A) and de-anonymizing users en-masse (Section V-C), we need r lists containing $1 \cdots r$ users respectively (to find threshold lists), where r is the granularity of rounding applied to the website's victim audience size. For audiences of size greater than 1,000, the value of r can be in the hundreds or greater (see section IV-B). However, by working with subsets of the audience of less than 1,000 users, the value of r can be reduced to 10 and the lists can be re-used with different subsets. For example, to infer the phone numbers of a website audience with 8,000 members, it would be necessary to use around 100 padding lists to find a threshold list. However, if at a time we only attempt to learn any matching phone numbers that end with a particular digit (by excluding those that have phone numbers that end with other digits), the size of the reduced audience would be roughly a tenth of the original size ($\sim$800), which would be rounded to the nearest 10 and would only require 10 padding lists.

Time taken: For our email–phone linkage attack (Section V-B), there is a setup time of a few days to upload the phone number lists (only done once) and of a few hours to find a threshold audience (done once for every batch of victims); parallelization can reduce this time. Then, it typically only takes an additional hour to find the phone number of any victim.

For our attack de-anonymizing particular web visitors (Section V-A), once the phone number and padding lists are all uploaded (similar cost as above), we only need to upload a record with the victim's information (roughly 20 minutes). Then, we only need to run around r queries, taking a few minutes per victim.

Attack step	Num of lists uploaded	Size of each list	Queries	Time taken	Times performed
Upload phone lists	82 (France)	20M (France)	0	$<$ 1 week (France)	Once
	70 (US)	1M (US)	0	$<$ One day (US)	Once
Upload padding lists for V-A	r	Up to r	0	1-2 hours if $r = 100$	Once
Upload padding lists for V-C	r	Up to r	0	1-2 hours if $r = 100$	Once
Find threshold list for V-B	200 (one upload round)	$<$ 2,000	200	3–4 hours	Once per batch
	30 (two upload rounds)	$<$ 2,000	30	4–6 hours	Once per batch
V-A inference step	1 (victim's record)	1	$\sim r$	Up to 20 min	Once per victim
V-B inference step	1 (threshold list plus victim)	$<$ 2,000	One per phone list	Up to 1 hour	Once per victim
V-C inference step	0	–	$O(md(r+10))$	$O(md(r+10))$ seconds	Once per audience

TABLE III: The cost of performing various steps for the three attacks described in the paper.

For our en-masse de-anonymization attack (Section V-C), once the lists are all uploaded (similar cost as above), the worst-case requires $O(md(r + 10))$ queries (where d is the number of digits in the phone number, m is the total number of matching phone numbers, and r is the rounding applied to the audience size as previously described); e.g., these would require around two hours when $m = 10$, $d = 7$, and $r = 100$ (at a query each second). However, this worst-case estimate assumes that phone numbers do not share any prefixes; when phone numbers share prefixes, the time taken for inferring a particular prefix is amortized across all the phone numbers that share the prefix. Hence, for larger audiences, the attack can potentially scale significantly sub-linearly. Also, as pointed out in Section V-C, by uploading the lists to multiple advertiser accounts and inferring the matching phone numbers in parallel, the time taken can go down proportionally.

Generality Any similar advertising platform would be incentivized to provide the functionality that Facebook's advertising platform provides (custom audiences, de-duplication, size statistics, etc.), as many of these are requested by the advertisers. Indeed, as described in Section II, other advertising platforms provide PII-based targeting, and provide similar size statistics. While Instagram, Google [38], and Twitter allow the use of both email addresses and phone numbers, Pinterest and LinkedIn allow the use of both emails and mobile advertiser IDs (identifiers for the user's mobile device). This suggests the potential for similar attacks on other platforms, but due to space constraints, we leave exploring them to future work. However, it is worth noting that most other platforms have larger minimum sizes for audiences; this alone does not significantly raise the difficulty of our attacks, as the attacker need only pad any uploaded audiences with additional records (e.g., voter records).

In order to check if other platforms are vulnerable, a similar process as presented in this paper can be followed: (i) determine the properties of size statistics (e.g., consistency, monotonicity, presence of noise), and whether the platform de-duplicates different kinds of PII; if so, (ii) collect padding lists with appropriate PII and size (based on the granularity of size statistics) and evaluate the attacks presented in Section V.

Other PII linkages When inferring a victim's PII (Section V-B), the adversary need not be limited to phone numbers; any PII that can be enumerated can be used by assigning a unique d-digit number as an index to each enumerated PII element, and applying the same attack to infer the matching d-digit index. For example, since Facebook's advertising platform allows audiences to be created using combinations of (First name, Last name, City, State) or (First name, Last name, Date of birth), the same attack could be used to infer the phone number of a victim knowing only their name and city, or their name and date of birth. Worse, the adversary could enumerate combinations of common names and potential dates of birth in the U.S., assign them indices, and then infer the name and birthdate of a victim given only their email address.

VI. Defenses

We now address how Facebook can fix the attacks.

A. Defense approaches and non-fixes

Recall that all of our attacks were enabled by two implementation characteristics from Section IV: (1) the rounding of audience size statistics, and (2) the de-duplication of users. Simply removing all size estimates is likely to be impractical, as this feature is likely useful to advertisers. Thus, Facebook must find a tradeoff between providing utility to advertisers and ensuring user privacy; using coarse-grained audience size estimates alone is insufficient, as we showed an adversary can construct audiences that fall on the rounding threshold. We first discuss potential defense approaches that raise the cost for the attacker, but do not prevent the attack.

Anomaly detection A platform could use anomaly detection (e.g., on the rate of queries, the size of lists, the contents of lists, etc.) to identify malicious advertisers. The main challenges with this approach are that the attacker could spread their queries out across multiple accounts, across time, and mix in other legitimate queries to evade detection. Moreover, the attacker need not upload easily-detectable lists (e.g., lists with all phone numbers that begin with a single digit); the attacker can easily make it more difficult to detect a pattern. Thus, as observed in other contexts (such as fake account detection), approaches based on anomaly detection struggle to provide guarantees of robustness.

Rate-limit API queries Alternatively, the platform could rate-limit API queries; while this will increase the time taken to conduct the attack, the attacker can again overcome the constraints by using multiple accounts. Thus, this raises the bar for attackers, but is insufficient for determined or powerful attackers, or for attackers who are only interested in targeting certain users (e.g., celebrities or dissidents).

Financial disincentives Alternatively, an advertising platform could employ financial disincentives by charging advertisers for every audience created or size estimate obtained. While such disincentives would increase the cost to the attacker, they may not be well-received by advertisers, who often create a large number of audiences and run many campaigns. Moreover, advertisers who wish to target particular victims would likely not be dissuaded by such costs.

Noisy size estimates While the addition of random noise would make the proposed attacks harder, an adversary could still circumvent that by making many repeated queries and performing statistical analysis on the results. For example, if Facebook adds uniform noise between 0 and $n-1$ (without rounding), then $\frac{log(\epsilon)}{2log(1-\frac{1}{n})}$ samples of each of two audiences are necessary to say whether the sizes come from the same distribution with probability $1-\epsilon$. If we set ϵ = 0.01 and $n = 20$, then we require only 45 samples of each audience size.

Applying differential privacy [7] might be robust against the described attacks. However, as pointed out by Korolova [18], deploying differential privacy is challenging in advertising platforms as the adversary can easily create multiple advertising accounts and issue many queries to the platform.

B. Secure mechanism that restricts de-duplication

Overall, there does not appear to be a simple and robust solution based on modifying how audience size statistics are calculated that would prevent our attacks while maintaining utility. Instead, we propose a solution based on changing the way Facebook de-duplicates users.

Model of the platform For the purpose of the defense we propose, we assume the advertising platform provides advertisers the following functionalities: (1) creating PII-based audiences by either uploading records or using tracking pixels, (2) obtaining the size of these PII-based audiences, and (3) combining these PII-based audiences using $\cup$, $\cap$, and $\setminus$, and obtaining the size of the resulting audience.

Proposed mechanism As is done on Facebook today (see Section III-A), the platform should define a set of *PII types* P_i, where each PII type is defined by a set of attributes that uniquely identify users. The PII types can contain a *single attribute* (e.g., {Email}), or *multiple attributes* (e.g., {Name, Date of Birth}). However, all PII types should be *minimal*, meaning if any attribute is removed, it is no longer a PII type (e.g., if {Name, Date of Birth} is a PII type, then {Name} cannot be a PII type).

Since tracking pixel audiences are not created by explicitly uploading records, we need to treat them differently: each tracking pixel audience is viewed as a unique PII type P_i.

We require that the platform have a *priority order* of PII types $\{P_1, P_2, \cdots\}$; it can be arbitrary, as long as it is static.

Advertisers can create custom audiences by uploading a list of records L, with each record containing an arbitrary number of attribute values. Note that we impose no constraint on what advertisers can upload; the security guarantees will come from how this upload is treated by the platform. Hence, a record might contain attributes that comprise zero, one, or several PII types and can potentially contain some attributes of a multi-attribute PII (without containing the entire multi-attribute PII).

The platform then turns the uploaded records into a custom audience, internally represented as a *set* of *reduced records*. For each uploaded record, the platform finds the smallest i such that all the attributes comprising P_i are present in the uploaded record; the platform then discards all other attributes in the record. This reduced record is added to the set representing the custom audience (with de-duplication). If no complete PII type from the ordered list is contained in the record (e.g., the record only contains a value for attribute Name which alone does not constitute a PII type), then the record is dropped.

Finally, the platform provides the size estimate of the custom audience created from L, denoted $s(L)$, by *matching* each of the reduced records against the user database. For each reduced record, if the attributes actually match an account, $s(L)$ is increased by 1. Thus, a user may be counted twice if they appear in the custom audience as two different reduced records (e.g., if the user's {Email} and {Phone} are in two separate reduced records).

The precise security guarantees provided by our solution are given in the following result.

Theorem 1. No size estimate $s(L)$ for any L can give any extra information about a PII type P_k of a user to an attacker who knows another PII type P_j of the user, and knows all existing values of P_k in the database. Formally, for any user u, any PII types P_j and P_k ($k \neq j$), any y such that there exists some user with $P_k = y$, and any list of records L,

$$Pr(P_k^{(u)} = y | P_j^{(u)} = x, s(L)) = Pr(P_k^{(u)} = y | P_j^{(u)} = x). \quad (1)$$

where $P_k^{(u)}$ denotes the value of PII type P_k for user u as per the platform's internal database.

Theorem 1 shows that our proposed method to compute size estimates $s(L)$ is secure against the attacks in this paper. Specifically, we consider an attacker that knows a PII type P_j about a victim u (i.e., knows that $P_j^{(u)} = x$), and we show that the result of $s(L)$ for any L gives no extra information about any other PII type P_k of the same individual. Theorem 1 follows directly, using Bayes rule, from the following lemma.

Lemma 1. Consider any user u, any PII types P_j and P_k ($k \neq j$) and any y such that there exists a user with $P_k = y$. For any list L, given that $P_j^{(u)} = x$, $s(L)$ is independent of whether $P_k^{(u)} = y$.

Proof. We prove that for any list L, the value of $s(L)$ is the same whether or not $P_k^{(u)} = y$.

First note that the result is trivial if L does not contain at least one record with $P_j = x$ and one record with $P_k = y$ (possibly the same record). Then we distinguish two cases.

Case 1: suppose that there is no record in L with both $P_j = x$ and $P_k = y$. Since we know that there exists a user with $P_k =$

y, those records will be counted independently of whether or not $P_k^{(u)} = y$. Then the result follows from the fact that records with different PII types are not de-duplicated, regardless of whether they belong to the same user.

Case 2: suppose that there is one record with $P_j = x$ and $P_k = y$. Then one of the PII types (or both if there are other higher priority PII types in the record) will be disregarded independently of whether they both correspond to the same user or not, and the resulting list falls in Case 1. □

Operations on the lists Until this point, we have only addressed the size statistics of a custom audience resulting from an uploaded list. We now discuss computing the size statistics of a combination of PII-based audiences, using $\cup$, $\cap$, and $\setminus$ operations. The definition of $s(...)$ for combinations follows the same principle as the definition of $s(L)$. Let L_1 and L_2 be two PII-based audiences. Then $s(L_1 \cup L_2)$, $s(L_1 \cap L_2)$, $s(L_1 \setminus L_2)$ are computed based on the corresponding operations on the reduced records that make up L_i (in the case of tracking pixel audiences, all records are single-attribute and are inherently reduced). As with $s(L)$, the final size statistic is then computed by matching each of the resulting reduced records after applying the operation against the user database, counting users multiple times if they are represented by two separate reduced records.

Then, the Theorem 1 holds on $s(L_1 \cup L_2)$, $s(L_1 \cap L_2)$, $s(L_1 \setminus L_2)$; this follows directly from the following lemma.

Lemma 2. Consider any user u, PII types P_j and P_k ($k \neq j$) and y such that there exists a user with $P_k = y$. For any lists L_1 and L_2, given that $P_j^{(u)} = x$, $s(L_1 \cup L_2)$, $s(L_1 \cap L_2)$ and $s(L_1 \setminus L_2)$ are independent of whether $P_k^{(u)} = y$.

Proof. The proof is similar to that of Lemma 1: we show that the size estimate of the audience resulting from $L_1 \cup L_2$, $L_1 \cap L_2$, or $L_1 \setminus L_2$ as defined above is the same whether or not $P_k^{(u)} = y$. First note that the result is trivial unless one of the lists contains at least one record with $P_j = x$ and the other at least one record with $P_k = y$. Then we distinguish two cases.

Case 1: If there is no record with both $P_j = x$ and $P_k = y$ in either list, then the result follows from the fact that records with different PII types are not de-duplicated, regardless of whether they belong to the same user.

Case 2: If there is one record with $P_j = x$ and $P_k = y$, then one of the PII types (or both if there are other higher priority PII types in the record) will be removed independently of whether they both correspond to the same user or not, and we are back to Case 1. □

C. Defense discussion

We now provide a brief discussion of our proposed defense.

Implementation Our proposed defense—replacing the existing matching algorithm by the one outlined above—is straightforward to implement. While we assumed that custom audiences were internally represented as sets of reduced records, this does not need to be the case to implement our solution; the platform could instead represent them as today, but then *tag* the users in the list with the record by which they were matched (to facilitate calculating the size statistics).

Accuracy of size estimates Our proposed method will overestimate the number of users in the audience if there are multiple records with different PII types corresponding to the same user; this is intentional as de-duplicating these users opens the door to our attacks. However, any defense must make a trade-off between utility to advertisers and privacy for users; our defense guarantees that the PII linkage attacks proposed in this paper will not be possible, while not significantly reducing the utility of size estimates to advertisers. In rounding size estimates (as they do today), Facebook already provides inaccurate statistics to advertisers; this shows that Facebook has decided that obfuscating the true value to provide privacy for users is acceptable. Given this, we believe that our defense strikes a better tradeoff than what Facebook already does. Our defense will not affect advertisers who have only one type of PII, or those who upload lists that do not actually contain multiple references to the same user.

Limitations We note that, as stated in the theorem, $s(L)$ gives no extra information *beyond* information about existence of an individual with a given value y of PII type P_k. Indeed, our mechanism does not prevent an attacker from discovering whether or not there *exists* an individual with $P_k = y$. This is important in particular for the multi-attribute PII case. Suppose for instance that the attacker knows that $P_j^{(u)} = x$ and that he knows parts of the attributes that constitute $P_k^{(u)}$ (a multi-attribute PII). Then the attacker could still potentially learn information on $P_k^{(u)}$ by finding out which of the existing PII type P_k are consistent with his background information. For example, if a user's name happened to be unique globally, the attacker could upload combinations of the {Name, Date of Birth}; the one that matched would reveal the user's date of birth. However, this is a separate attack from the ones in this paper, as it is an attack on the creation of custom audiences themselves; we hope to address it in future work.

Facebook's fix Facebook has acknowledged the vulnerabilities reported in this paper and has confirmed to us that it has deployed a stricter version of our defense: no size statistics will be provided for audiences created using multiple PII attributes. Additionally, no size estimates will be provided when combining audiences that were created using different PII attributes. This stricter defense uses our fix's core idea of preventing linking, but does so by providing no size statistics when multiple PII attributes are used; compared to our suggested defense, it may be easier to implement but provides lower utility to advertisers.

VII. Related work

Obtaining user information Due to the large amount of personal data that online social networking sites collect, they have been the vector for many privacy attacks over the

years. Krishnamurthy et al. [19] found that multiple pieces of users' personal information (name, city, zip code, email address, phone numbers, gender, birthday, age, employer, friends, activities, and interests) are either always available or available by default on most of the 12 OSN sites they examined, implying that potential privacy leakages on these sites can pose a significant issue. For example, the "Download Your Information" tool allowed users to inadvertently download the contact data for their friends, including email addresses and phone numbers, which they were not supposed to have access to [11]. Moreover, studies have developed methodologies to infer user attributes such as gender, political views, and religious views [22], [32], [15]; these use features such as homophily to make predictions about users' attributes.

Even worse, some social networking sites have been observed leaking private user information to third parties: for example, Facebook was observed leaking user IDs, usernames, and even personal details (e.g., name, gender, other profile data, friends, networks, wall posts, photos, and likes) to advertisers, typically via Request URIs and HTTP headers (e.g., Host, User-Agent, Referrer, and Cookie) [20], [9]. Even though Facebook has since fixed the issue [29], Facebook Messenger was observed leaking PII to third parties in 2016 [10].

Exploiting advertising services Korolova [18] demonstrated in 2011 that the microtargeting capabilities of Facebook's advertising system may allow advertisers to draw inferences about users who click on a Facebook ad, such as their age or sexual orientation. Using attribute-based targeting, she targeted the user so precisely that she created an audience of exactly one user;[6] she could then create multiple audiences with different values of the unknown attribute and determine which of the advertisements results in a click.

In comparison to our work, there are two main differences: first, Korolova's attack requires that there be a set of targeting attributes that *uniquely* (or almost uniquely) identify the user among all Facebook users (e.g., Male, lives in St. Louis, age 32, ...). Instead, we only need one piece of personal information (often easily available) sufficient to target the user via a custom audience (e.g., their email address). Second, our attacks show how user information beyond targeting attributes can be inferred (i.e., we show how users' phone numbers or names can be inferred from their email address). Third, Facebook mitigated Korolova's attack by imposing a minimum audience size of 20; our work shows that even with this mitigation, attacks are still possible.

Linking user identities The custom audience feature represents a linking mechanism between external data and Facebook's data. Researchers have examined the related problem of linking the accounts of a single user across multiple services [39], [17], [5], [21], [13], [12], [14]. For example, Vosecky et al. proposed an approach to link identities by exploiting user profile attributes, including name, date of birth, and email [39]. Balduzzi et al. [5] used email addresses to discover multiple identities of a user. However, as was recently measured by Goga et al. [14], only a few attributes like usernames and real names are available in OSN sites reliably, which makes such linkage attacks successful only for a small fraction of users. As a result, recent studies have moved beyond users' personal information and examined using textual and image content for linking [21], [13], [12], [14]. In our work, we are not proposing a new method of linking, but are instead examining the privacy implications of a linking mechanism provided by advertising platforms.

Custom audiences There has been surprisingly little academic study of custom audiences. The most recent related study by Minkus et al. [23] empirically examined how offline information (such as voter records) could be matched to public Facebook profiles, thereby enabling the inference of features such as the user's residential address, date and year of birth, and political affiliation. Tucker [33] investigated how users' perception of control over their personal information affects how likely they are to click on online advertising on Facebook, and found that giving users control over their private information can benefit advertising on Facebook. This implies that users want to control their own data used in online advertising; however, the current privacy settings [8] give users very few options. Even worse, users do not have control over their offline data, which can be used in the custom audiences feature. Our work is the first to study custom audiences directly and to point out the privacy leaks that occur via the custom audience interface.

VIII. Conclusion

The vast amounts of user data that social networking services have collected is now utilized by their advertising platforms to allow advertisers to target users via their PII. In this paper, we have shown how the inclusion of PII-based targeting opens up new privacy leaks in advertising platforms. By giving advertisers fine-grained control over the set of users targeted, and by providing them with coarse-grained statistics of audience sizes, the platforms open themselves to powerful attacks that can let an adversary learn private information about users. While we have proposed a solution to the attacks we uncovered, our work shows that platforms need to carefully audit their interfaces when introducing PII-based targeting.

Acknowledgments

We thank the anonymous reviewers and our shepherd, Carmela Troncoso, for their helpful comments. We also thank Paul Francis and Deepak Garg for their valuable feedback, Vincent Toubiana and Franck Baudot at CNIL for helping us bring the problem to Facebook's attention, and Facebook's security team for implementing the fix. This research was supported by NSF through grants CNS-1563320 and CNS-1616234, ANR through grants ANR-17-CE23-0014 and ANR-16-TERC-0012-01, Institut Mines Telecom through the "Future & Ruptures" program, the Alexander von Humboldt Foundation, and a Data Transparency Lab grant.

[6] At the time, Facebook did not have any minimum size requirement on audiences, thus enabling the attack.

REFERENCES

[1] About Customer Match. https://support.google.com/adwords/answer/6379332?hl=en.

[2] About Potential Reach. https://www.facebook.com/business/help/1665333080167380?helpref=faq_content.

[3] Account Targeting. https://business.linkedin.com/marketing-solutions/ad-targeting/account-targeting.

[4] Audience Targeting. https://help.pinterest.com/en/articles/targeting.

[5] M. Balduzzi, C. Platzer, T. Holz, E. Kirda, D. Balzarotti, and C. Kruegel. Abusing Social Networks for Automated User Profiling. *RAID*, 2010.

[6] T. S. Bernard, T. Hsu, N. Perlroth, and R. Lieber. Equifax Says Cyberattack May Have Affected 143 Million in the U.S. https://www.nytimes.com/2017/09/07/business/equifax-cyberattack.html.

[7] C. Dwork. Differential Privacy. *ICALP*, 2006.

[8] Facebook Ads Preferences. https://www.facebook.com/ads/preferences.

[9] Facebook Leaks Usernames, User IDs, and Personal Details to Advertisers. http://www.benedelman.org/news/052010-1.html.

[10] Facebook Messenger Chatbots Can Leak Your Private Information. https://www.techworm.net/2016/09/facebook-messenger-chatbots-can-leak-private-information.html.

[11] D. Guarini. Experts Say Facebook Leak of 6 Million Users' Data Might Be Bigger Than We Thought. http://www.huffingtonpost.com/2013/06/27/facebook-leak-data_n_3510100.html.

[12] O. Goga. Matching User Accounts Across Online Social Networks: Methods and Applications. Ph.D. Thesis, Pierre and Marie Curie University, 2014.

[13] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira. Exploiting Innocuous Activity for Correlating Users Across Sites. *WWW*, 2013.

[14] O. Goga, P. Loiseau, R. Sommer, R. Teixeira, and K. P. Gummadi. On the Reliability of Profile Matching Across Large Online Social Networks. *ACM KDD*, 2015.

[15] N. Z. Gong and B. Liu. You Are Who You Know and How You Behave: Attribute Inference Attacks via Users' Social Friends and Behaviors. *USENIX Security*, 2016.

[16] How Google uses Customer Match data. https://support.google.com/adwords/answer/6334160.

[17] T. Iofciu, P. Fankhauser, F. Abel, and K. Bischoff. Identifying Users Across Social Tagging Systems. *AAAI ICWSM*, 2011.

[18] A. Korolova. Privacy Violations Using Microtargeted Ads: A Case Study. *Journal of Privacy and Confidentiality*, 3(1), 2011.

[19] B. Krishnamurthy and C. E. Wills. On the Leakage of Personally Identifiable Information via Online Social Networks. *ACM SIGCOMM WOSN*, 2009.

[20] B. Krishnamurthy, K. Naryshkin, and C. E. Wills. Privacy leakage vs. Protection measures: the growing disconnect. *IEEE W2SP*, 2011.

[21] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan. HYDRA: Large-scale Social Identity Linkage via Heterogeneous Behavior Modeling. *ACM SIGMOD*, 2014.

[22] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You Are Who You Know: Inferring User Profiles in Online Social Networks. *ACM WSDM*, 2010.

[23] T. Minkus, Y. Ding, R. Dey, and K. W. Ross. The City Privacy Attack: Combining Social Media and Public Records for Detailed Profiles of Adults and Children. *ACM COSN*, 2015.

[24] Marketing API: Custom Audience. https://developers.facebook.com/docs/marketing-api/reference/custom-audience.

[25] R. Nazarian. Facebook Loophole Exposes Private Phone Numbers, Here's How to Close It. https://www.digitaltrends.com/social-media/facebook-phone-number-hackers-flaw/.

[26] New Targeting Tools Make Pinterest Ads Even More Effective. https://business.pinterest.com/en/blog/new-targeting-tools-make-pinterest-ads-even-more-effective.

[27] N. Popper. Identity Thieves Hijack Cellphone Accounts to Go After Virtual Currency. https://www.nytimes.com/2017/08/21/business/dealbook/phone-hack-bitcoin-virtual-currency.html.

[28] T. Peterson. How Facebook's Custom Audiences Won Over Adland. http://adage.com/article/digital/facebook-s-custom-audiences-won-adland/297700/.

[29] Protecting Privacy with Referrers. Facebook Engineering's Notes. http://www.facebook.com/notes/facebook-engineering/protecting-privacy-with-referrers/392382738919.

[30] S. E. Rasmussen and J. C. Wong. Facebook Was Where Pakistan Could Debate Religion. Now it's a Tool to Punish 'blasphemers'. https://www.theguardian.com/technology/2017/jul/19/facebook-pakistan-blasphemy-laws-censorship.

[31] D. Storm. Hacker Says He Can Get Phone Numbers on Facebook Which Are Not Supposed to be Public. http://www.computerworld.com/article/3158109/security/hacker-says-he-can-get-phone-numbers-on-facebook-which-are-not-supposed-to-be-public.html.

[32] K. Thomas, C. Grier, and D. M. Nicol. Unfriendly: Multi-party Privacy Risks in Social Networks. *PETS*, Springer-Verlag, 2010.

[33] C. E. Tucker. Social Networks, Personalized Advertising, and Privacy Controls. *Journal of Marketing Research*, 2014.

[34] Tailored Audiences File Data. https://dev.twitter.com/ads/audiences/file-data.

[35] Target Facebook Ads to People on Your Contact List. https://www.facebook.com/business/a/custom-audiences.

[36] Target Custom Groups of Twitter Users. https://business.twitter.com/en/targeting/tailored-audiences.html.

[37] US Voter List Information. http://voterlist.electproject.org.

[38] Upload Data Files and Manage Your Customer Match Audiences. https://support.google.com/adwords/answer/6276125?hl=en.

[39] J. Vosecky, D. Hong, and V. Y. Shen. User Identification Across Multiple Social Networks. *NDT*, 2009.

[40] What Are Mobile Advertising IDs and When Should I Use Them with Custom Audiences? https://www.facebook.com/business/help/570474483033581.

[41] What Happens When I Upload My Customer List to Facebook? https://www.facebook.com/business/help/112061095610075.

[42] What's a Custom Audience from a Customer List? https://www.facebook.com/business/help/341425252616329/.

[43] What's the Difference Between Estimated Daily Reach and Potential Reach? https://www.facebook.com/business/help/1438142206453359.

APPENDIX A
HANDLING VICTIMS WITH MULTIPLE PHONE NUMBERS

When describing our attacks in Section V, we found that they did not work as anticipated for users who had uploaded *multiple* phone numbers to Facebook. We now describe and evaluate extensions to our attacks that allow us to handle such users.

A. Extending email-phone inference attack

In Section V-B, we described how our email-phone number linkage attack fails when the victim has provided multiple phone numbers to Facebook. When this happens, we infer *multiple* values for each digit of the phone number, corresponding to the multiple phone numbers.

We first describe a solution that involves the same one-time preparatory steps as the attack described in Section V-B. We then briefly describe how, by conducting additional one-time preparatory steps, the attacker could significantly reduce the effort required per victim to disambiguate and infer the multiple matching phone numbers, thereby scaling up the attack.

Naïve method for disambiguation Let us assume that the attack in Section V-B infers multiple values for some digits of the phone number, and that the maximum number of values inferred for a digit is n (for simplicity, let us assume this occurs only at one digit i). This indicates that Facebook must have at least n phones corresponding to this user. For each value j inferred for the digit i, we perform the following steps:

1) Create the set L_{ij}^{poss} of all possible phone numbers with value j at that digit i, and whose other digits each have one of the values inferred for that respective digit.
2) Represent each phone number in L_{ij}^{poss} with a unique d-digit index (where $d = log_{10}(|L_{ij}^{poss}|)$).
3) Use the attack described in section V-B to link the given email to the d digit index corresponding to the matching phone number; this could be accomplished by dividing L_{ij}^{poss} into $10d$ subsets just as in Section V-B.

This method requires uploading $10d$ custom audiences for each phone to be inferred. However, the size of L_{ij}^{poss} is small compared to the number of all possible phones. For example, when $n = 3$, $|L_{ij}^{poss}| \leq 3^{10}$ for 10 digit phone numbers; hence, it would take less time to create these custom audiences compared to the L_{ij} created in Section V-B. In general, an attacker should be able to use this method to infer all matching phone numbers in a few hours. In case there are multiple matching phone numbers in one of the L_{ij}^{poss}, we can recursively apply the same method to further disambiguate those numbers.

Scaling up the disambiguation We now briefly describe how an attacker could speed up the disambiguation with some additional one-time effort at the beginning. One method to do this would be to upload sets of phone numbers $L_{i_1 j_1 i_2 j_2}$ whose i_1-th digit has a value j_1 and whose i_2-th digit has a value j_2, with one set corresponding to each possible combination of i_1,i_2,j_1, and j_2. Checking whether the victim user exists in $L_{i_1 j_1 i_2 j_2}$ allows the attacker to determine whether at least one of the matching phone numbers has a value j_1 for its i_1-th digit *and* has a value j_2 for its i_2-th digit. It is then trivial to do the disambiguation between the possible phone numbers, by appropriately checking whether the victim user exists in $L_{i_1 j_1 i_2 j_2}$, and to thereby infer all the matching phone numbers.

This method does not require the attacker to create any additional audiences when conducting the attack, allowing the attacker to infer all the victim's matching phone numbers in well under an hour. However, it requires the attacker to upload $100\binom{d}{2}$ sets upfront (one set corresponding to each possible combination of i_1,i_2,j_1, and j_2), where d is the total number of digits in the phone number. For example, a 10 digit phone number would require the attacker to upload 4,500 custom audiences, while a 8 digit phone number would require the attacker to upload 2,800 custom audiences. This would take a few days, and would have to be launched across multiple advertiser accounts (since each account allows the creation of at most 500 PII-based audiences); however, it would not be difficult for a determined attacker.

Evaluation We apply the naïve disambiguation technique to the two volunteers with multiple phones from V-B; we find we are correctly able to infer the matching phones for these two users (who have provided two and four phone numbers to Facebook, respectively).

B. Extending en-masse de-anonymization attack

In Section V-C, we described how our attack to de-anonymize members of a PII-based audience en-masse would fail to infer phone numbers of victims who provided multiple phone numbers to Facebook. We first explain why the en-masse de-anonymization attack fails to infer the matching phone numbers for users who have provided multiple phone numbers to Facebook. We then introduce notation that we use for our modified attack; and then describe the modified attack itself.

Explanation for failure of the attack We explain the failure of the method with an example: assume a user u who has provided two phone numbers to Facebook (`791 3444` and `799 5485`) visits our webpage and is added to the pixel audience. As part of its breadth-first search for matching phone numbers, our attack would first determine that there exists at least one matching phone number in the pixel audience starting with the prefix 79. In the next iteration, the breadth-first search would then construct an upper threshold list A_t such that $P \cup A_t \setminus (L_{17}^c \cup L_{29}^c)$ falls over the rounding threshold, and then additionally exclude L_{3j} to check whether any of the matching phone numbers beginning with 79 have j as the next digit. At the end of the iteration, the search would determine that there exists at least one matching phone number corresponding to each of the prefixes 791 and 799.

The following iteration is where the failure occurs. In order to determine digits following the prefix 791, the breadth-first search would first construct an upper threshold list such that $P \cup A_t \setminus (L_{17}^c \cup L_{29}^c \cup L_{31}^c)$ falls over the rounding threshold, and then additionally exclude L_{4j}. We see immediately that by excluding $(L_{17}^c \cup L_{29}^c \cup L_{31}^c)$ when determining the upper threshold list, we automatically exclude users whose phone numbers start with the prefix 799; since exclude trumps include, and since u has a phone number starting with 799, u is excluded from the list. As a result, we cannot infer any additional digits of this phone number for u; the same problem occurs when trying to infer the digit after the 799 prefix.

Notation As before, we denote the size of a PII-based audience A as $s(A)$. We also assume that indices begin from 1. We also use regular-expression-like notation to refer to a set of phone numbers, with the i-th term of a regular expression r denoting the set of possible values for the i-th digit of the phone numbers in the set (we denote this set of possible values by $r[i]$). For example the set of phone numbers $\{$`791 3444`, `799 5485`$\}$ is denoted by the regular expression $r = 79(1 \vee 9)(3 \vee 5)4(4 \vee 8)(4 \vee 5)$ where $r[3] = \{1, 9\}$ is the set of possible values for the third digits of the phone numbers in the set. Regular expressions are either empty (denoted by $\varnothing$) or of length k, corresponding to the first k digits of the matching phone numbers, where $k \in \{1, ..., d\}$ (d is the total number of digits in the phone number); an empty regular expression is assumed to match all possible phone numbers.

Given a regular expression r of length k that specifies the possible values for the first k digits of matching phone

numbers, and given a set S of possible values for the $(k+1)$-th digit, we can construct a regular expression $r' = r.S$ that is satisfied only by phone numbers whose first k digits satisfy r and whose $(k+1)$-th digit falls in S. Note that as a natural consequence of this definition, if S denotes the possible values for the first digit, $r = \varnothing.S$ is satisfied only by phone numbers that have first digits that fall in S.

In order to select phone numbers that satisfy a regular expression r from a PII-based audience P, we do the following: for each digit index i whose possible values are specified by r, we exclude $(\cup_{j \in r[i]} L_{ij})^c$ from P (this is alternately accomplished by excluding $(\cup_{j \notin r[i]} L_{ij})$ from P). We denote this previously described operation by $P \setminus r$; note that $P \setminus \varnothing = P$. If we want to additionally exclude some list L from $P \setminus r$, we denote this by $P \setminus (r, L)$

We now describe the attack do the en-masse deanonymization and infer all matching phone numbers when users have provided multiple phone numbers to Facebook.

Inferring all matching phone numbers The attack is described in Figure 3, and is conducted by calling the DEANONYMIZEPHONESWRAPPER function with the pixel audience P and the lists of phone numbers L_{ij} as parameters; it returns the set L_{inf} of all phone numbers of all (active) users in P. At a high level the attack has the following steps (the outer loop in the DEANONYMIZEPHONES function):

1) Determine a set of regular expressions, each of which are satisfied by the phone numbers of a minimal number of users in P, and which together cover as many matching phone numbers as possible (line 8 in DEANONYMIZEPHONES).
2) Disambiguate the phone numbers satisfying each regular expression (loop starting at line 9 in DEANONYMIZEPHONES).
3) Exclude the inferred phone numbers from P (by uploading a list with the inferred phone numbers and then excluding the created audience), and repeat the previous steps, until there are no more phone numbers to be inferred from P .

The final step is necessary because in some cases, certain phone numbers can "hide" other phone numbers from being inferred, given the way our algorithm works; by excluding inferred phone numbers from P and performing another round, we can infer these phone numbers that were hidden in the previous round. We first briefly explain how the attack determines a set of regular expressions in step 1, and then briefly explain how it disambiguates the phone numbers satisfying a regular expression in step 2.

Determining a set of regular expressions Note that for users with multiple matching phone numbers, it is not possible to directly determine all matching phone numbers just by using the phone number lists L_{ij} that we created. This is because of the constraint imposed by Facebook's platform that we can only exclude a disjunction of custom audiences (and not a conjunction). The best we can do is to learn all possible values of users' phone numbers at each digit for different users (similar to what we get from the email phone linkage attack for users with multiple phone numbers).

Therefore, we aim to determine a set of regular expressions, each of which is satisfied by a minimal number of users' phone numbers. For each such regular expression, inferring which phone numbers satisfying it are present in the pixel audience (i.e., performing step 2) would require us to repeat the de-anonymization attack focused on the set of phone numbers satisfying the regular expression. Since each of these regular expressions are satisfied by a small set of phone numbers, we would be able to perform step 2 and disambiguate the underlying phone numbers for each relatively quickly.

The procedure to compute these regular expressions (described in GETREGEXES function in Figure 4) works by searching the tree of all possible phone numbers, similar to the procedure described in Section V-C; the only difference being that it recursively computes regular expressions of matching phone numbers, rather than individual matching phone numbers. In iteration k, the procedure takes each regular expression r of length $k-1$, and the set of k-th digits of phone numbers that match r (denoted by $nextDigits[r]$), and then finds the minimal subsets of $nextDigits[r]$ for which we do not run into the problem that the original de-anonymization ran into (of one prefix of a user excluding another); these are then appended to r to find regular expressions of size k. In order to find the minimal subsets of $nextDigits[r]$ described above, we check for each subset of $nextDigits[r]$ if we are able to infer additional digits ($nextDigitSet$) at index $(k+1)$; if no, then we discard that subset, if yes, then save $nextDigitSet$ for use in the next iteration.[7]

Disambiguating phone numbers The procedure for disambiguating the phone numbers satisfying a regular expression (described in the DISAMBIGUATE function in Figure 3) is very similar to the naive disambiguation method proposed earlier in the Appendix to disambiguate phone numbers matching an email address. We find the index i_{max} of the digit for which the regular expression allows the maximum number of possible values; for each possible value in $r[i_{max}]$, we fix that value, enumerate all possible phone numbers which have that fixed value at i_{max} and which satisfy the regular expression, and then recursively call DEANONYMIZEPHONES to learn all the phone numbers within this enumerated set of users that are part of the pixel audience.

Discussion We first note that the de-anonymization attack is performed in multiple rounds of inferring regular expressions, and then disambiguating phone numbers. Phone numbers of users with only one phone number are guaranteed to be discovered in the very first round, and will be represented by regular expressions r that match them uniquely (i.e., $r[i]$ is a singleton set for all i). Also, note that while the attack has been

[7] As a consequence of this procedure, the GETREGEXES function produces regular expressions that have the following properties: (i) For any user, they are satisfied by all his phone numbers, or none of his phone numbers, and (ii) Each regular expression of size k covers the minimal number of users possible for a regular expression of size k (for $k \in \{1, ...d-1\}$)

function DEANONYMIZEPHONESWRAPPER($P, [L_{ij}]$) ▷ Infer all matching phone numbers in P
  $L_{inf} \leftarrow$ DEANONYMIZEPHONES($P, \{\}, [L_{ij}]$)
  return L_{inf}
end function
function DEANONYMIZEPHONES($P, L_{inf}, [L_{ij}]$) ▷ Infer all matching phone numbers in $P \setminus L_{inf}$
  repeat
    $L_{new} \leftarrow \{\}$
    $rSet \leftarrow$ GETREGEXES($P, L_{inf}, [L_{ij}]$)
    for $r \in rSet$ **do**
      $L_{disambiguated} \leftarrow$ DISAMBIGUATE($r, P, L_{inf}, [L_{ij}]$)
      $L_{new} \leftarrow L_{new} \cup L_{disambiguated}$
    end for
    $L_{inf} \leftarrow L_{inf} \cup L_{new}$
  until $|L_{new}| = 0$
  return L_{inf}
end function
function DISAMBIGUATE($r, P, L_{inf}, [L_{ij}]$) ▷ Disambiguate all phone numbers in $P \setminus L_{inf}$ that satisfy r
  $L \leftarrow \cup_{i,j} L_{ij}$
  Find i_{max} such that $|r[i_{max}]|$ is maximum
  For $v \in r[i_{max}]$ create L^v (list of phones in L that satisfy r and have value v at index i_{max})
  Assign each phone in L^v a unique index
  Partition L^v into L^v_{ij} (where L^v_{ij} is the list of phones in L^v whose index has i-th digit j)
  $L_{disambiguated} \leftarrow$ DEANONYMIZEPHONES($P, L_{inf}, [L^v_{ij}]$)
  return $L_{disambiguated}$
end function

Fig. 3: Algorithm to de-anonymize phone numbers when users have multiple phones.

described assuming that the PII being inferred is the phone number, as mentioned previously in the paper, the same attack can be used for inferring other PII (by enumerating possible values for the PII and by inferring the index of matching users in the enumerated set).

Evaluation We apply the extended version of the en-masse de-anonymization attack on the audience containing 14 volunteers from Boston from Section V-C. Recall that in Section V-C we were already able to infer the phone numbers for nine users with single phone numbers, and found one user in the audience who had provided multiple phone numbers to Facebook. The modified attack outputs 11 phone numbers, nine of which correctly match the nine users with single phone numbers, and the remaining two of which correctly match the user who had provided multiple phones. Our method took around two hours to do the entire search and output the phone numbers by sending queries to Facebook in a serial fashion; as mentioned in Section V-C, this time could be cut likely down significantly by parallelizing the queries sent to Facebook.

function GETREGEXES($P, L_{inf}, [L_{ij}]$) ▷ Get regular expressions of user's phone numbers in $P \setminus L_{inf}$
    $rSet \leftarrow \{\varnothing\}$
    $nextDigits[\varnothing] \leftarrow \{0, 1, ..., 9\}$
    for $i \in [1, ..., d-1]$ **do**
        $rSet_{curr} \leftarrow \{\}$
        for $r \in rSet$ **do**
            $rSet_{new}, nextDigits_{curr} \leftarrow$ UPDATEREGEXES$(r, nextDigits[r], i+1, P, L_{inf}, [L_{ij}])$
            $rSet_{curr} \leftarrow rSet_{curr} \cup rSet_{new}$
            for $r' \in nextDigits_{curr}$ **do**
                $nextDigits[r'] \leftarrow nextDigits_{curr}[r']$
            end for
        end for
        if $i < d-1$ **then**
            $rSet \leftarrow rSet_{curr}$
        else
            $rSet \leftarrow \{r.nextDigits[r] : r \in rSet_{curr}\}$
        end if
    end for
    return $rSet$
end function
function UPDATEREGEXES($r, digits, i, P, L_{inf}, [L_{ij}]$)
    $rSet \leftarrow \{\}$, $nextDigits \leftarrow \{\}$
    Create a list D containing subsets of $digits$ in increasing order of size
    $D_{selected} \leftarrow \{\}$
    $index \leftarrow 1$
    while $index \leq len(D)$ **do**
        $digits_{curr} \leftarrow D[index]$
        If $digits_{curr} \supset digits_{selected}$ for some $digits_{selected} \in D_{selected}$, increment $index$ and skip to next iteration
        Find upper threshold audience such that $s(P \cup A_t \setminus (L_{inf}, r.digits_{curr}))$ falls just over rounding threshold τ
        $nextDigitSet \leftarrow \{j : s(A_t \cup P \setminus (L_{inf}, r.digits_{curr}, L_{ij})) \leq \tau\}$
        if $|nextDigitSet| > 0$ **then**
            $rSet \leftarrow rSet \cup \{r.digits_{curr}\}$
            $nextDigits[r.digits_{curr}] \leftarrow nextDigitSet$
            $D_{selected} \leftarrow D_{selected} \cup digits_{curr}$
        end if
        $index \leftarrow index + 1$
    end while
    return $rSet, nextDigits$
end function

Fig. 4: Helper function that finds regular expressions matching the phone numbers of users in audience.