Security and Privacy Measurements in Social Networks: Experiences and Lessons Learned

Iasonas Polakis*, Federico Maggi†, Stefano Zanero†, Angelos D. Keromytis*
* Network Security Lab, Columbia University, USA. {polakis,angelos}@cs.columbia.edu
† DEIB, Politecnico di Milano, Italy. {federico.maggi,stefano.zanero}@polimi.it

***Abstract*—We describe our experience gained while exploring practical security and privacy problems in a real-world, large-scale social network (i.e., Facebook), and summarize our conclusions in a series of "lessons learned". We first conclude that it is better to adequately describe the potential ethical concerns from the very beginning and plan ahead the institutional review board (IRB) request. Even though sometimes optional, the IRB approval is a valuable point from the reviewer's perspective. Another aspect that needs planning is getting in touch with the online social network security team, which takes a substantial amount of time. With their support, "bending the rules" (e.g., using scrapers) when the experimental goals require so, is easier. Clearly, in cases where critical technical vulnerabilities are found during the research, the general recommendations for responsible disclosure should be followed. Gaining the audience's engagement and trust was essential to the success of our user study. Participants felt more comfortable when subscribing to our experiments, and also responsibly reported bugs and glitches. We did not observe the same behavior in crowd-sourcing workers, who were instead more interested in obtaining their rewards. On a related point, our experience suggests that crowd sourcing should not be used alone: Setting up tasks is more time consuming than it seems, and researchers must insert some sentinel checks to ensure that workers are not submitting random answers.**

From a logistics point of view, we learned that having at least a high-level plan of the experiments pays back, especially when the IRB requires a detailed description of the work and the data to be collected. However, over planning can be dangerous because the measurement goals can change dynamically. From a technical point of view, partially connected to the logistics remarks, having a complex and large data-gathering and analysis framework may be counterproductive in terms of set-up and management overhead. From our experience we suggest to choose simple technologies that scale up if needed but, more importantly, can scale down. For example, launching a quick query should be straightforward, and the frameworks should not impose too much overhead for formulating it. We conclude with a series of practical recommendations on how to successfully collect data from online social networks (e.g., using techniques for network multi presence, mimicking user behavior, and other crawling "tricks"') and avoid abusing the online service, while gathering the data required by the experiments.

I. Introduction

Massive user participation has rendered online social networks (OSNs) a valuable target for attackers, and a lucrative platform for deploying various types of attacks, ranging from spam [26] to personalized phishing campaigns [11]. Ample research efforts have been dedicated to explore the potential ways in which OSNs can be exploited and attacked, and subsequently develop the appropriate defense mechanisms that will hinder actual incidents.

Research Challenges. Researching the security of online social networks presents a series of interesting challenges. On one hand, the large-scale nature of such services requires efficient and accurate experimentation methodologies as well as a sturdy infrastructure. Consider that miscreants are known to capitalize on popular events that are expressed through viral behavior on OSNs such as Facebook and Twitter. For example, during the night of the 2012 U.S. presidential election, 31 million Tweets were posted online at a peak rate of 372,452 Tweets per minute [28]. Had researchers wanted to study and analyze such content in search of SPAM or malware campaigns, it would have been a daunting task. Keeping up with the rate of user-generated content, also places significant burden on the network connection both in terms of bandwidth as well as latency. Moreover, maintaining such content for subsequent analysis mandates a large amount of storage space and processing power. On the other hand, the unique nature of security research presents both ethical and legal issues. From the standpoint of the OSN service, a researcher might seem like an attacker and from the standpoint of a researcher, probing the service to identify weaknesses might mean producing tools for the actual attackers. Despite this growing interest, we are not aware of any systematization nor retrospective work on OSN research with a focus on system security.

Our Experience. In this paper we present our experiences from our recent research on Facebook. In our use case, the goal was to build a system to identify Facebook's Social Authentication mechanism characteristics, analyze its behavior and point out the security weaknesses. Our work has been published in the proceedings of 2012's ACSAC [21]. We hereby present our experiences, and the mishaps we encountered and solved while interacting with a large-scale OSN service such as Facebook, with the goal of conducting a user-centered analysis.

We walk through the logistical and technical challenges that we had to take care of, and provide the reader with a series of practical recommendations in order to carry out a measurement experiment in the smoothest way possible. Empirical works on online social networks are probably the most representative example of user-centered measurements and, as such, require time and certain aspects to be taken into account. To this end, we provide a "meta workflow" that other researchers can adapt to their needs, in order to avoid mistakes that we committed when designing and developing our experiments. However, we also learned that a strict plan could sometime become counterproductive: We provide practical examples that explain

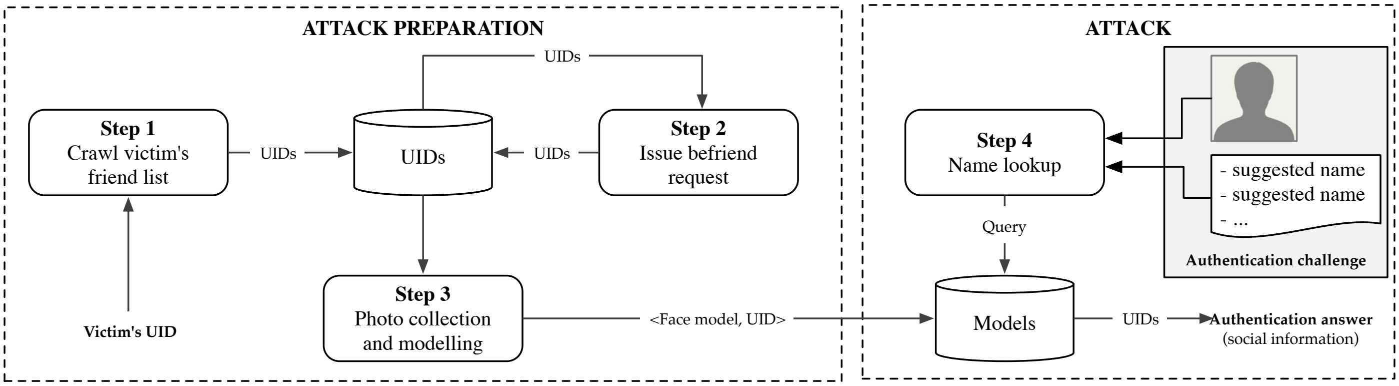

Fig. 1. Use case: An automated system that collects data from Facebook and exploit it to attempt to break its face-based social authentication system. The system operates in four steps. In **Step 1** we retrieve the victim's friend list using his or her UID. Then, in **Step 2** (optional), we send befriend requests, so that we have more photos to extract faces from and build face classifiers in **Step 3**. In **Step 4**, given a photo, we query the models to retrieve the corresponding UID and thus match a name to face. Step 4 takes place in real time while steps 1 through 3 at a previous time.

when and how we needed to revise our plan, and adapt our measurement infrastructure to changing goals. Moreover, we discuss the delicate aspects related to terms of service in OSNs, which often conflict with the goals of the security researcher, who needs to crawl corner regions of the network.

We summarize our conclusions on each aspect in a series of "lessons learned", which provide a starting point for future research in the same areas, or with similar measurement goals.

In summary, in this paper:

- We analyze the typical stages of large-scale experiments in OSNs and present them as a work flow, as we argue that the particularities of OSN security research demand a structured approach, and provide pointers for identifying non-technical challenges and requirements. We are not aware of prior work on the subject.
- We describe the peculiarities of OSN-related experiments, and stress the importance of considering non-technical issues when designing and deploying experiments. Ethical and legal aspects can greatly affect the outcome of an IRB protocol request. We identify the outdated nature of the IRB request procedure, and argue that it should be revised to reflect the requirements of current research.
- We discuss the benefits of cloud services for outsourcing data analysis in large-scale OSN experiments. We present the trade-offs we faced, and how a hybrid approach allowed us to conduct our various experiments efficiently.

II. Case Study

In this section we introduce our use case so as to provide the necessary context for understanding the various issues we came across during our research, and the decisions we had to make. In particular, we briefly describe Facebook's Social Authentication (SA) mechanism, and our experimental methodology and findings while evaluating its security properties. For an in-depth description, we encourage the reader to read our work [21].

A. *Photo Based Social Authentication*

Conceptually, the social authentication mechanism is an "instantiation" of the two-factor authentication scheme. Two-factor authentication offers additional security to traditional single-factor, password-based, approaches. Usually the second factor is something the user possesses, for example a hardware token, and needs to prove its physical presence in real-time during the authentication process. An attacker would have to steal both the password and the physical token to be able to impersonate the user and log into the Web service in the future.

In SA the idea is, essentially, to leverage a user's social knowledge as the second factor, so as to prevent attackers from compromising online profiles, after having stolen their credentials. Facebook's implementation of the social authentication, or SA in short, is activated only when certain security heuristics flag a login attempt as suspicious, for instance when taking place from a country or computer for the first time. In that case, right after the standard, password-based authentication, the user is prompted with a sequence of 7 pages featuring challenges, where each challenge is comprised of 3 photos of an online friend. For each page, the user must find the true name of the depicted friend, with the sole help of 6 suggested names, chosen from the user's social circle. The user is allowed to fail in 2 challenges, or skip them, but must correctly identify the depicted people in at least 5 in order to be granted access. The idea is that nobody but the actual user will possess the necessary social information to correctly pass the test.

B. *Threat Model*

The threat model initially covered all scenarios where the user's password had been compromised. However, Kim and collaborators in [13] showed that users with tightly connected social graphs, such as a university network on Facebook, share enough social knowledge to defeat the secrecy assumption made by Facebook. As such, the threat model was reduced to attacks made by complete strangers half-way around the world.

C. *Research Goals and Findings*

Our research was based on the hypothesis that any stranger (i.e., anyone not in a user's online social circle) can acquire enough knowledge to pass the photo-based challenges. For this, our system crawls the public portion of a user's online social graph, collects a labeled dataset of photos depicting the user's friends, and trains face-recognition algorithms to automatically recognize faces presented during SA challenges. An overview of our system is summarized in Figure 1. Our work resulted in a system able to successfully break Facebook's SA under our original hypothesis.

We first implemented a crawler that measures the amount of sensitive data left publicly accessible by Facebook users. Our experiments showed that an attacker can obtain access to sensitive information for at least 42% of a user's friends that Facebook uses to generate the SA challenges. Next, we measured the efficiency of face-recognition algorithms against the SA mechanism. Using the aforementioned publicly-accessible information, our attacker trains a classifier and builds accurate facial models of the victim's friends. Our findings showed that by relying solely on publicly-accessible information, an attacker can automatically solve 22% of the SA tests, and gain a significant advantage for an additional 56% of the tests.

III. Approaching Online Social Network Research

Before researchers begin their work on a social networking service, they have to make ethical and legal decisions that will determine the guidelines under which they will have to operate.

A. *Ethical Aspects*

The first task that researchers must undergo is to consider the ethical issues that may arise from the type of work they wish to do. Designing a study or an experiment should take into consideration its impact on the participants, and the community in general. For instance, collecting user data, even when publicly available, may violate the privacy of those users who might not be aware of the ways their sensitive information is leaking on the Web. At the same time, even if users provide their explicit consent for data collection, it is the researchers' responsibility to provide assurances regarding the confidentiality and privacy of that information, a data retention time frame, as well as information on its secure disposal. Apart from privacy issues, one needs to consider how active experiments within social networking services will be, especially when interaction with actual users is involved (e.g., through dummy accounts). They must take into account that they might impact the users and the online service itself. Overall, researchers need to set a goal of minimizing any impact, and avoiding any permanent side effects of their actions.

Responsible Disclosure. A special category is research carried out for security reasons. For example, in our case we sought to evaluate the effectiveness of Facebook's SA mechanism. Our analysis of identified weaknesses and the documentation of our methodology could be misused by attackers to defeat this security mechanism. However, we believe that attackers might already be studying the weaknesses of SA and that it is better to point them out first, so as to raise the bar for the attackers. As a matter of fact, we have already detailed ways to enhance the security of this mechanism, and are continuing to explore improved countermeasures, and hope that our work has motivated other researches to do the same. It is crucial to improve the security mechanisms of a service with such a massive user base and amount of personal information. Note that the weaknesses that we identified were *conceptual* rather than *technical*. In other words, we did not find an exploitable technical vulnerability, but rather a flawed design. In case critical technical vulnerabilities are found, the researchers should follow responsible disclosure guidelines[1] and the social network's policy[2].

Moreover, as our goal was to determine the security provided by Facebook's SA against someone that has access to a victim's Facebook password, we had several options for testing our hypothesis. One option was to select real Facebook users at random and utilize dictionaries of common passwords to gain access, and actually test our experiments in the most realistic environment possible. Or we could employ lists of known compromised Facebook accounts available in the Internet underground. However, considering the ethical nature of such actions we decided to create, and attack, dummy accounts of our own. To be able to trigger the SA mechanism we needed to populate our accounts with friends and, thus, issued friend requests to random individuals using those accounts. We did not attack, or otherwise negatively affect those individuals. At the end of our experiments, we severed our links to them by un-friending them and deactivated our dummy accounts.

When in Doubt, Simulate. Another example is when we needed to repeat an experiment to test whether we were able to break the photo-based authentication. To obtain the most realistic conditions, we should have chosen to trigger the authentication mechanism and use our training base to try to break it, and repeat the experiment any time we needed to tune our algorithm's parameters. In addition to slowing down the experiment dramatically, it would have caused Facebook to ban our accounts. We had a better choice for performing an arbitrary number of experiments: We used the millions of photos in our data store to synthetically generate the authentication challenges, by randomly selecting an expected answer (i.e., a user's name), five wrong answers, and three random photos each. To the best of our knowledge, Facebook randomly selected candidate photos based on the likelihood that they contain a face. Thanks to the availability of an offline face-detection algorithm we were able to select—and index—only those photos that contained a face, and use them in our experiments. This allowed us to perform an arbitrarily large number of realistic experiments, while preserving the integrity of our dummy accounts, and avoid possible service abuse.

> **Lesson learned:** Ethical aspects must obviously be taken into account from the very beginning. Being familiar with, or at least making educated guesses about, the internals of the OSN web application helps avoiding their abuse. When feasible, try to replicate the settings offline, rather than polling/abusing the online service.

B. *Institutional review board (IRB)*

The role of an IRB is to supervise research involving human subjects that is carried out by members of that institution. Its mission is to ensure that researchers' actions adhere to the

[1] https://www.schneier.com/essays/archives/2007/01/schneier_full_disclo.html

[2] In the case of Facebook: https://www.facebook.com/whitehat

code of ethics, and protect the participants from physical or psychological harm. Traditionally, the focus of such committees has been biomedical and behavioral research. Nevertheless, an IRB needs to review all research involving human subjects and decide whether to allow it, as well as whether is should be exempt or not from supervision during its duration.

In our case, researchers at Columbia University are asked to provide information, in the form of an IRB protocol request, that seems to be oriented towards traditional forms of research involving humans in a physical location subject to tangible stimuli such as a specific substance, device or signal. Among the information asked to provide are the address of the building and room the research will take place and whether radiation or hazardous materials will be involved.

On the other hand, an IRB protocol request is not oriented towards research taking place online so it is up to the researchers to describe the various parameters in their own words. For example, the issue of collection of user information, including personally-identifying information (PII) [15], is not addressed apart from the case of social-security numbers (SSN). Besides the guarantees that a researcher should provide for strictly limiting the data collection to that needed for the purposes of the research, they should also state the measures taken to safeguard the privacy of the participants—both against first and third parties—when such data is stored and processed for the duration of the research. Moreover, there is no explicit mandate or procedure to properly dispose of collected data in the case of electronic information collected online. Also, the use of cloud services for storage and processing makes an interesting case of its own as the researchers' own privacy policy is tangled with the terms of service and privacy policy of the cloud services being used. In our case we made sure to honor such requirements and inform our participants when cloud services were used. We believe, however, that as researchers are explicitly instructed on how to handle biological material, the same should apply in electronic and online research involving human subjects.

We found that the IRB of Columbia University was knowledgeable and responsive regarding matters of online research. Nevertheless, we believe that future researchers could benefit from additional information, instructions and training.

> **Lesson learned:** Having a study approved by the IRB can be valuable and is generally considered a positive point by the technical program committee during the review process.

C. Terms of Service

Social networking services base their operation in the storage, processing and management of user-provided data. In an attempt to offer assurances to their users and safeguard their business model, they shape their terms of service accordingly, which in certain cases might prove to be counter-intuitive. For instance, Facebook clearly disapproves [7] accessing its pages and respective data (even the *public* portions) when done in an automated way (i.e., crawling). Large Web search engines are explicitly white-listed [8] and everybody else must apply for, and acquire, written permission by Facebook. In 2010, the social networking service did not hesitate to take legal action against an Internet entrepreneur [30] who collected publicly-available user data from Facebook and, subsequently, released an anonymized version for researchers to use. This was, obviously, a very extreme case, because the entrepreneur *released* the collected data, possibly affecting Facebook's business model.

Overall, we argue that the crawling of publicly-accessible data should not be hindered by social networking services as long as the data collection process does not directly impact the normal operation of the service. Moreover, researchers should also be able to collect private user-owned data as long as the respective users have given their explicit consent. A very encouraging step towards this direction has been taken by Twitter, which exposes API calls [27], not only facilitating the collection of publicly-available user-provided content (i.e., the tweets), but also offering specific API endpoints that perform data sampling.

Researcher: The Good, the Bad, the Ugly. A controversial aspect regards when researchers need to "bend the rules" to carry out their work that could benefit the scientific community, the users of the social networking service and even the service itself. For instance, security researchers might need to evaluate the behavior of users as well as privacy-preserving measures taken by the social networking service so as to propose improvements. This could require creating dummy or test accounts for interacting with the service and its users. Such actions are explicitly forbidden under Facebook's terms of service. However, real-world attackers are not bound by the terms of service. If researchers do not bend the rules, the security and privacy of the social networking services might go untested and vulnerabilities might remain hidden from the general community while being known to potential attackers. We argue that, in such cases, deviating from the terms of service is justified as long as the service itself or the users do not suffer irrevocable damage from such actions. In other words, in the case of dummy accounts, as long as they are destroyed by the end of the research project and any data collected or inflicted side effects are reversed, all sides are benefited. While conducting another research, we strove to obtain Facebook's consensus before proceeding, but it required a substantial amount of time that would have delayed our results. So, even though we were aware that we did not entirely adhere to the terms of service, it seemed that no better option existed—except, of course, that of not conducting our research at all.

> **Lesson learned:** Strive to get in touch with the OSN security team, but try to plan in advance. Be aware of what terms of services you are not adhering to, and include detailed recommendations alongside the discovered security vulnerabilities.

IV. HUMAN SUBJECTS

Having pointed out the security inefficiencies of Facebook's SA, we are continuing with research to improve it. For that matter, we carried out a user study to receive feedback from human participants on our efforts to enhance the quality of this authentication mechanism. Our results can be found here [19]. Since we were evaluating our idea of a modified, photo-based authentication scheme for something as popular as a social networking service, we opted for a diverse set of participants that could not be found within an academic institution. Therefore, we explored reaching human subjects through crowd-sourcing services, namely Amazon Mechanical Turk (AMT)[3] and ResearchMatch [4].

Initially we designed the environment of our study and decided on its parameters. We felt it was important to create a respectable and trustworthy presence of our study online, in order to invite strangers to participate in it. For this reason we setup a site informing of our research[5]. In the home page, we adopted a layout that quickly and clearly conveys important information about our study. We identify ourselves to visitors of the page, briefly describe the purpose of the study and what information potential participants would have to release to us for the duration of the study. Finally we describe our privacy policy and include a point of contact.

Next, we developed a Facebook application to facilitate the efficient interaction between the experiments driving our study and the participants. We opted for a Facebook application because, first of all, they are deployed within a sandbox run by Facebook itself and are, thus, governed by a series of permissions that clearly state and, subsequently, enforce their capabilities and access to user data. This is important, as it inspires trust in users. Secondly, as we are using Facebook's SA as an example case for improving this type of security mechanism, it was important to integrate our work as close to the mechanics of the service itself as possible. Finally, as we require participants to grant us access to some of the data in their profile (e.g., their social graph), a Facebook application enables direct access. This is also in accordance with our efforts to respect user privacy and minimize collection of potentially sensitive information. In other words, having direct access to Facebook rather than having users upload that information to our own infrastructure means we are able to minimize the amount of a user's information kept within our infrastructure and operate on a best-effort basis to utilize pointers towards the actual Facebook source.

Once our study was ready to commence, we began the process of inviting Facebook users to participate. Through this process we gained experience on human subject involvement in OSN research, and hereby present the obstacles and issues that arose, and how we chose to deal with them.

> **Lesson learned:** Through a respectable website that transparently informs users about our ongoing research we were able to "engage" the community and gain trust among the participants.

A. *Give good user incentives*

To attract human subjects to participate in the experiments, researchers need to provide incentives. One possible way to entice users is to setup the experiment in such a way that it will appear as a game. Our first attempt was to organize the study as a series of challenges and provide users with a score, depending on how many challenges they solved correctly, and the ability to share that score with their Facebook friends. We thought that this could create a game-like feeling and competitiveness that would attract more users, and provide the incentives for completing multiple tests. This, however, was not the case, as most users found certain parts of our study tedious and stopped after a couple completed challenges. Ideally, the game-like approach would make the participation fun but it turns out that the received satisfaction from solving challenges related to their online friends was not enough to justify the effort required. Therefore, we investigated other incentives to attract a large set of users. The alternative is to provide users with monetary rewards for participating. A method for gaining access to a large set of potential test subjects is through a crowd-sourcing platform. Such platforms, like the Amazon Mechanical Turk service and ResearchMatch, allow one to express his interest for participants or "workers" (or "turks") for a specific task and it is up to the users to contact the initiator of the task. We did not consider active advertisement campaigns (e.g., via email), as that may give the wrong first impression.

B. *Crowd-sourcing $\neq$ Easy workforce*

Our first attempt was to leverage the Amazon Mechanical Turk (AMT) platform, where users receive monetary compensation for completing tasks. We created a task where users were asked to install our Facebook application and participate in the study. However, our task was rejected by Amazon, as it violated their terms of service due to the following reasons: a) we asked for the participants' (Facebook) identity, b) we required them to register at another website (the site we set up for the study), and c) we required them to install an application. AMT is oriented towards finding humans to carry out simple data processing jobs (e.g., image classification). They could be asked to use a search engine to locate some information, transcribe audio to text or answer statistics-gathering questionnaires. Even though we argue that we were very careful and responsible when designing our study and took measures to ensure the security and privacy of the participants, AMT's automated screening process was unable to realize that and, therefore, did not allow us to proceed. Even though research has been powered by AMT even in security-related projects [25], the service is geared towards more restricted tasks.

An encouraging example is that of ResearchMatch, which pairs researchers with potential participants. Researchers describe the type of study they are running as well as the profile of suitable participants. Users of the service identify interesting cases and apply to join. Unlike the AMT model, monetary compensation is not a part of the process. While users are called volunteers, this does not mean researchers cannot incentivize with monetary compensation. This, however, takes place outside the service. The site's orientation towards the research community is evident by the fact that it requires a valid IRB protocol for any initiated study, and identifies researchers by their affiliation with selected institutions. At the moment it has a little over 30,000 volunteers [22] compared to the over 500,000 workers of AMT [18]. Nevertheless, its environment seems significantly more research-friendly, although it is currently limited to the population of the United States.

In order to take advantage of AMT, we ended up modifying the type of tasks. Instead of requiring the workers to identify their friends, we asked them to recognize well-known people (e.g., celebrities). This allowed us to remove the requirement of installing an application and asking for the workers' identity. However, the downside was that we had to had to translate

[3] http://mturk.com

[4] http://researchmatch.org

[5] http://resoauth.necst.it/

a complex task into a simple image-classification task, which required additional time. Alongside this large-scale study conducted thanks to AMT, we set up a smaller-scale experiment in a controlled environment, which we used as a pilot.

> **Lesson learned:** The anecdotal belief that crowd-sourcing services allow researchers to carry out any type of batch work turned out to be misleading in our case. Once again, having a backup plan (i.e., standalone website and network of contacts) allowed us to finish our study.

C. *Representativeness of Human Subjects Set*

Another issue that arises when outsourcing tasks to human test subjects, is the representativeness of the set (i.e., how they will perform compared to the general population). A biased selection of test subjects might skew the experimental results, and this should be taken into account when designing the experiment. Although this aspect must be taken into account, there is no recipe for ensuring good representativeness, apart from ensuring that each task is solved by many different workers. Platforms such as AMT allow, to some extent, to select workers based on their reputation (e.g., percentage of correctly solved tasks), but there is currently no support to enforce uniform geographic or demographic distributions of tasks. We agree with previous studies that suggested prudent practices when using crowd-sourcing services [23].

D. *Inspecting User Data*

During the implementation process of our custom-built face recognition software, we manually inspected user photos for tweaking our parameter selection as well as for debugging purposes. While this might raise ethical concerns as it entails inspecting personal and potentially-private information, this is often unavoidable in the context of research experimentation. Before installing our application, users were informed that their data may be inspected by researchers.

E. *Securing User Data*

An important factor when storing user data, even if it is only for the duration of the experiments, is to secure it. Typical procedure includes anonymizing the data. In our case, this was not possible because the stored information (e.g., User ID, photo URLs) was used at runtime by our system. Nonetheless, to avoid the leakage of potentially private information, in [21] we conducted the majority of our experiments using publicly-available information and photos. In the cases where we collected photos from real SA tests, which could be private, we deleted all the data after we finished our experiments.

To minimize the chances of user data being obtained by malicious third parties, all data was stored on a single machine located in the proximity of the NECSTLab at Politecnico di Milano, and could only be accessed by a user account created specifically for conducting our experiments. No external (both from other users or from other machines) access to data was allowed, and access to the user account was restricted with our SSH public keys. Also, no files or photos were ever moved from our machine to portable drives or through the network.

V. Workflow vs. Flexibility

On the one hand, in measurement experiments, having a workflow is essential. On the other hand, every measurement must be treated differently. There is no recipe for an experimental workflow, and we are by no means proposing one.

A. *Flexibility Pays Back*

One of the most painful lessons that we learned is that flexibility is paramount: being able to quickly re-design an analysis task was essential for us to probe and measure. A rigid workflow would not have helped. Nevertheless, some high-level procedures can be depicted, in the hope that other researchers find it a good starting point. Honestly, we were able to draw Figure 2 only once the work was completed by 60–70%: Do not expect to meet with your co-authors and prepare the workflow for the next 3–6 months of measurement. The diagram shows how the design and implementation phases are decomposed into tasks. From our experience, and from the examination of previous work on OSNs, we believe that the resulting workflow is relatively generic, and can be used as a guideline by researchers first approaching this subject.

We divide the high-level workflow into two phases. In the **Design Phase** we sketched a high-level outline of the experiments. In this phase, it is very useful to elicit the phases starting from (1) the data that needs to be collected and (2) the questions that need an answer. In this phase, the researchers must strive to identify the details of their experiments and data, so as to identify the most important ethical and legal issues that may arise in the future. In the **Implementation Phase**, the researchers have already created an outline of their experiments, and are required to continue with the detailed design and implementation of their experiments, which is driven by the availability of resources and existing tools. Again, flexibility always pays back. So, our advice is to keep the plan as an indication, and modify it whenever the results require so.

B. *Having a Plan Pays Back*

Even if some degree of flexibility is important, measurements with no plans at all do not go very far. A concrete case is when collecting user data or conducting experiments with human subjects. Usually, an institutional review board (IRB) must approve such experiments and, without a plan, there is no formal way to talk to an IRB. The IRB protocol request must contain a thorough explanation of the type of user data that will be collected as well as the nature of the experiments involving human test subjects that will take place. Therefore, researchers have to undergo a preparation phase where experiments are designed in detail and their goals are clearly stated. This deviates from other types of security research, such as exploring methods of misusing a system [12] or searching for vulnerabilities, that may follow more of a “hit and miss” approach. As such, after conceptualizing the experiment and clarifying the desirable end results, both technical and non-technical issues have to be considered. During the design of the data collection and analysis process, researchers must also identify the ethical and legal issues that arise. The handling of user data also mandates the design of the procedures that safeguard the data. When these processes have been designed in detail, the researchers can submit the information, and request

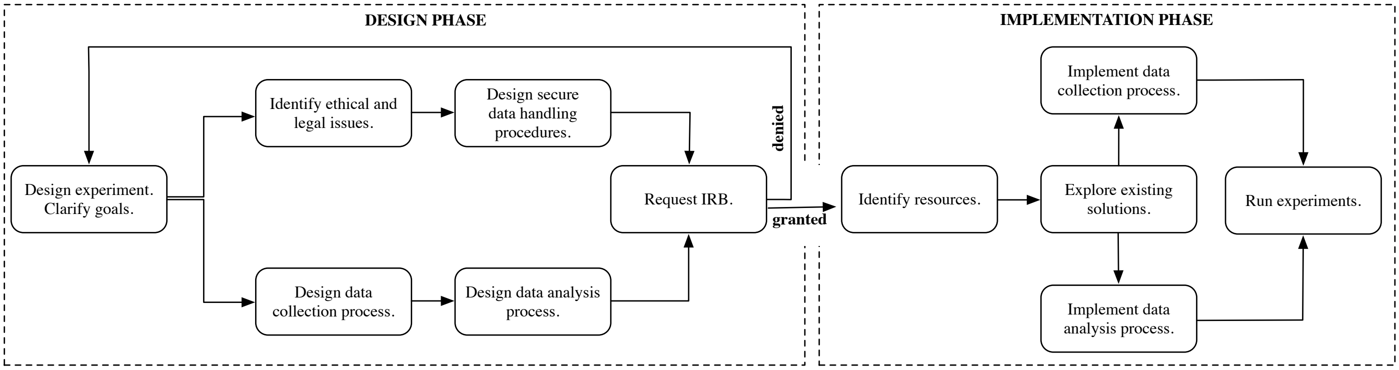
DESIGN PHASE
IMPLEMENTATION PHASE
Design experiment. Clarify goals.
Identify ethical and legal issues.
Design secure data handling procedures.
Design data collection process.
Design data analysis process.
Request IRB.
denied
granted
Identify resources.
Explore existing solutions.
Implement data collection process.
Implement data analysis process.
Run experiments.

Fig. 2. Overview of the process followed when designing and implementing our research work on the security of Facebook's Social Authentication mechanism.

an approval. If the IRB denies the request, the researchers will have to identify the reasons that led to this decision and re-design the experiments until they are approved.

C. Avoid Over Planning

If the IRB committee approves the request, the implementation phase begins. The first step is to identify the resources at the researchers' disposal. Resource availability is fundamental during this process, as it greatly influences many of the implementation decisions that will be made. Reviewing existing solutions and incorporating such systems can greatly reduce the implementation overhead. For instance, when implementing the data collection system, one may leverage distributed crawlers such as crawl-e[6], which supports crawling across multiple threads and machines. On the other hand, when resource availability is minimal, a centralized crawler with a custom, fine-grained allocation of the available resources may be a better option. As an example, we present the details of our crawler implementation in Section VII.

In regards to the data-analysis procedure, one of our goals was to demonstrate the severity of the security vulnerabilities of the SA mechanism, and the feasibility of our attack. In light of that, we wanted to show how the attack could be accomplished using off-the-shelf face-recognition algorithms as well as free cloud services, which are at any attacker's disposal. This affected the implementation phase of our data-analysis process, where we implemented a custom face recognition solution using an existing framework and also integrated a cloud based solution. We saw in practice, the importance of leveraging existing solutions that can provide much-needed functionality and, as it happened in our case, better results.

> **Lesson learned:** A high-level plan of the measurement experiments is necessary when presenting formal approvals or data-access requests. However, avoid over-planning and be ready to adapt quickly.

VI. Design Phase

In this section we detail the decisions when devising the design of the experiments. In our case, we needed to traverse public parts of Facebook's social graph, such that we could collect photos of users which we had previously befriended using a series of dummy accounts (see Section VII-B and VII-C for practical examples on how to create and maintain dummy accounts). We then needed to analyze the photos to produce a dataset of labeled faces, which we would supply to the face recognition algorithm to build models.

The aforementioned dummy accounts were treated as the victim accounts in our experimental scenario, where we assumed the role of the attacker. In this scenario, the attacker knows the password for the accounts, but lacks the social information to solve the SA challenges presented by Facebook. As a matter of fact, we did actually lack the social information even though we owned the victim accounts, as the friends were random strangers that we had befriended.

A. Data Storage: Simplicity Wins

When the experiments mandate the management of large—or, simply, unpredictably large—amounts of data, the selection of the appropriate type of database to be used is driven by two factors: *scalability* (both up and down) and *flexibility*. Scaling down is often under-estimated, although it is a crucial factor. For example, large and complex big-data frameworks pay back in term of scalability (up), but have a very steep setup and learning curve, such that a small modification is impractical.

Scaling Down. Scaling Up. We decided to implement our system upon a lightweight, non-relational database such as MongoDB or CouchDB. Relational databases such as MySQL and SQLite are suitable for small-scale pilots, but are not optimized for handling non-transactional (e.g., purely tabular), big-data repositories with evolving schema (e.g., new attributes or relationships). In addition, OSNs are well represented with graphs data structures, which do not map (natively) onto relational data structures. Last, and most importantly, queries in non-relational databases can scale up easily, if required, thanks to MapReduce. MapReduce has been extensively used in many researches with great benefits (e.g., [9]).

Normalization is Evil. During the phase of data modeling, one must take into account that the principal difference from relational, SQL-like databases is the ability to store and retrieve great quantities of data, and not the relationships between the elements. Thus, JOIN operations are not supported, and data needs to be de-normalized using explicit references between documents for emulating relationships. Even though non-relational databases cannot, necessarily, give full ACID

[6] http://code.google.com/p/crawl-e/

guarantees, at the same time they do offer a distributed, fault-tolerant architecture and the possibility to scale horizontally. These characteristics fit the prerequisites that stem from managing large amounts of data, where performance and real-time responses are more important than consistency.

Forget About Data Consistency. Apart from being suitable for the management of large volumes of data, non-relational databases are also very flexible as and there is no restriction for the mandatory application of a fixed schema. This results in the ability to change the structure of the collected data even after an experiment has started, without the need of rebuilding the entire database to make the old data consistent with the new structure. Most of the time, ensuring data consistency in an always-running experiment can be as easy and non-disruptive as adding proper "if" conditions in data processing routines.

ORMs are Evil: The Model is The Data. In practice, among all non-relational databases, we chose *MongoDB*[7], a document-oriented database, where data is handled in *collections* of *documents*. To draw a comparison between this concept and that of the SQL style, we could say that collections are like tables, and documents can be considered records. While every record in a table has the same sequence of fields, documents in a collection may have fields that are completely different. Additionally, the format of the responses (JSON) returned from the services in our experiments, perfectly matched the native data type of *dictionaries* in Python. Most importantly, having a data structure and model that maps directly in the chosen programming language is crucial because it removes the need for any object-relation or object-document mapping layer (ORM), which is often the source of bugs or bottlenecks. In addition, JSON is the data-exchange format adopted by many web-service APIs (including Facebook's). Also, in cases of multiple institutions collaborating on the same project, MongoDB offers two methods of cooperation: *replication* and *sharding*. The first one occurs through groups of servers, known as replica sets, and ensures redundancy, backup, and automatic failover. The latter distributes each database across a cluster of machines.

> **Lesson Learned**: Lightweight, flexible data-storage engines that easily scale both up and down, with low setup cost, pay back if compared to complex, big-data frameworks.

B. *File Storage*

The next design decision was about the type of file storage to be used. The available options in our case were a typical filesystem versus an embedded storage (e.g., GridFS), a system for storing an unlimited number of arbitrarily-large files directly into our data storage (i.e., MongoDB). The machine we chose for the experiments was already equipped with an ext3 formatted drive. The problem, however, is that the maximum number of i-nodes per directory is 32,000 in ext3, and that could pose serious limitations on the data we were about to gather and its organization on disk. Even though this limitation can be overcome using an ext4 filesystem or modifying some internal parameters by rebuilding the ext3 filesystem, this was not an optimal choice because folder indexing would have taken a considerably large time when the folder was accessed. While caching would have surely reduced this overhead, given the amount of data to be saved in files, it would not have solved the problem completely. Moreover, having a separate location for the files, creates an additional burden when setting up backup procedures. Therefore, we decided to rely on GridFS, which also allowed to easily reallocate the underlying database on a new, larger drive in case more space was needed. GridFS works by breaking large files into multiple chunks: It saves the chunks in one collection (`fs.chunks`) and metadata about the file in another collection (`fs.files`). When a query for a file is submitted, GridFS queries the chunk collections and returns the file one piece at a time.

In conclusion, although a filesystem can be seen as the simplest and fastest way, GridFS presents other advantages as well: data replication facilitates load balancing among distributed servers, millions of files can be co-located in a single logical directory without any performance impact and it has increased portability as file management is independent of the application technologies.

> **Lesson learned:** A file-storage solution integrated in the data-storage engine ensured zero setup time and high reliability, and avoids duplicating and maintaining multiple copies of the meta data (i.e., filesystem and data store).

C. *Develop vs. Offload*

When designing our face-recognition experiments, we identified two type of experiments that we needed to conduct. The first one demanded a more versatile approach towards the selection of algorithm parameters, while the second one was more demanding in terms of face-recognition accuracy. As such, we ended up building a custom solution as well as relying on an existing cloud-based service. On one hand, we did not want to become experts in another field of research. On the other hand, we needed to dig into the implementation details in order to modify some parameters. Given the level of maturity reached by face-detection and recognition technologies, and the availability of open-source libraries, we opted for investing some time to get to know the essential details, for understanding the consequences of our choices. However, we prepared a backup solution in case of unsatisfactory results.

Custom Solutions: Becoming Experts. The major advantage of a custom solution is the versatility in parameter tuning, as every aspect of the algorithm can be rigorously tested with different values. This was very important for specific experiments where we needed to measure the correlation between the size of the training dataset (i.e., number of faces), and the accuracy in recognizing a face. Furthermore, a custom solution allows to conduct multiple offline tests without incurring any limitations from the service provider. Finally, no network latency is present, which greatly affects the experiments' duration when dealing with large amounts of data.

Nonetheless, a custom solution also presents some disadvantages. First and foremost, it takes time and effort to refine it so as to be comparable to state-of-the-art systems. Specifically, the custom solution was effective when simulating a scenario of an attacker that has obtained a fairly large set of photos by infiltrating the victim's social circle with a dummy account. However, in the scenario of an attacker that only relies on a small amount of publicly-available data, the

[7] http://www.mongodb.org

accuracy of our custom solution was not as satisfactory. For this set of experiments, we needed a more accurate process, and explored the possibility of employing a cloud-based service that provides a more accurate face detection algorithm. Also, the computational power requirements are not negligible, as these algorithms are computationally intensive and have long execution times on commodity machines.

Cloud-based Services: Backup Option. Finding a backup option was very easy in our case, thanks to the availability of publicly accessible APIs for face detection and recognition. Ironically, the company offering the API of our choice, face.com, was acquired by Facebook itself, causing some delays to our experiments. However, API directories such as Mashape [8] allowed us access to alternative, bleeding-edge technology.

The advantage of a state-of-the-art solution is the far greater accuracy compared to a custom solution. Development of the face-recognition system is effectively being outsourced to the service, which offers a production-ready tool for researchers to use—although, as mentioned, with limited tweaking options. In terms of available resources, depending on the service and its usual load, researchers may be able to significantly increase their processing capabilities as opposed to utilizing only their local means. By building upon an existing service, no development time is needed for designing and implementing an algorithm that will be far less accurate (unless developed by computer-vision experts), and can be better allocated on other core tasks. Another advantage of a cloud-based solution is that the REST constraints, when present, ensures good scalability.

The most limiting disadvantage of using an existing service, is the restriction of API usage. As the number of API requests per hour is limited, conducting a large number of experiments will last longer than using a custom solution where an infinite number of experiments can be conducted without any restrictions.

> **Lesson learned:** Become familiar with the essential aspects of technologies borrowed from other research areas, and use cloud-based services only if not planning to repeat your experiments in the future, as the services may change or, worse, be shut down at some point.

VII. Implementation Phase

This section describes the practical aspects that we had to deal with during the implementation of our measurement study.

A. *Crawling Online Social Networks*

One of the most important aspects of research in OSNs is the collection of data. The massive user base mandates the retrieval of large amounts of data that will consist a large enough sample to accurately reflect the various properties of the network. As such, the implementation of the crawler was integral to our experimental process.

Scraping vs. API. The first dilemma we encountered was to decide whether our crawler would use Facebook's public APIs to collect the information we needed, or if we would build an entirely custom solution that did not rely on the API. Our first concern was whether a strict rate limit applies for the API usage. However, Facebook allows 100 million API calls per day [9], which is large enough for our intended experiments.

However, for our experiments we wanted to collect the data in the manner of an attacker, who collects any data left publicly available by Facebook users. We were aware that we should have avoided requests at high rates: Indeed, our choice was not dictated by the need of faster crawling speed, but simply of availability of certain segments of the social graph. Therefore, we implemented our solution so as to mimic the behavior of a "normal" user browsing the website. Thus, for every user, we retrieved the actual page that contains the list of friends, followed by the albums and photos. The entire solution was implemented in `Python` and every single web request was issued through the `urllib2` library, which impersonated the HTTP User Agent of a popular Web browser.

Think Asynchronously. The main problem with our tool was that some steps in the crawling procedures (i.e., album and photo retrieval) were much slower than others, which resulted in them becoming a severe bottleneck of the system. To overcome this obstacle, we built our crawler in a completely modular and asynchronous fashion. We built four standalone modules, each consuming data from an input queue and inserting tasks in an output queue that was in turn processed by the following component or saved into the database. Modern programming languages or libraries such as Akka[10] or Pykka[11] encourage this practice, and allow the developers to abstract the tedious low-level details of event-based or reactive-based programming.

The first module was the *Friend Collector*. It took a list of user IDs (UIDs) as input and browsed the Facebook profile of each one. It retrieved all the data of the user's contacts (name and UID) using regular expressions created specifically for that page's structure. If the number of friends was too large and they did not fit on one page, the module performed multiple requests (just like a browser would have done) to get all of them in multiple passes. Every bit of information was saved into our database and marked as *non-crawled*; at the same time the new UIDs were put in the input queue, so that they would eventually be processed and their friends would be retrieved as well. At the end of this step the initial UID was placed in the output queue, ready to be handled by the other modules.

The second module was the *Album Collector*. Each UID in the input queue was used to reconstruct the URLs of their photo album pages, which were subsequently scraped, and the exact URLs of all the albums were saved into the database and put in the output queue. The user with the respective UID was then marked as *crawled*.

This same structure was used by the *Photo Info Collector*. It took as input the queue of album URLs and crawled them one by one, saving into the database the real URL of the photo as well as all the tags each photo contained (coordinates and related UID) and placed the URLs in an output queue.

The last module was the *Photo Downloader*, which downloaded every image it found on its input queue and saved it into the database using the MD5 hash as the key.

[8] Mashape: https://www.mashape.com/search?query=face+detection

[9] https://developers.facebook.com/policy/

[10] http://akka.io

[11] http://www.pykka.org

Once we started the crawling procedure we understood two key factors. First, this crawling process was much faster than relying on the public APIs (although overcoming the speed limits was not our goal). Second, it was crucial to follow a pipeline design and also effectively distribute resources among the modules, as there was no way we could efficiently retrieve all the data following a sequential process of each user. Overcoming this issue was not trivial. In our initial experiments, our system was acquiring user information for a massive amount of users, while the number of downloaded albums and photos was, of course, significantly smaller. That resulted in our database being filled with potential targets for which we did not have any useful information for our experiments (i.e., photos and tags). For this reason we created an entire web application to keep the single queues monitored and be able to modify the behavior of the single modules at runtime: We could change the number of threads they used, change the request rate or even start and stop them at will, so as to de-allocate resources when necessary. Having a way to continuously monitor the experiments was essential. After a couple weeks of fine-tuning, we ended up putting as few resources as possible on the *Friend Collector* allocating no more than one thread and using a low request rate, while the greatest part of the resources was assigned to the downloading of album and photo information, with up to 32 threads per module.

> **Lesson learned:** Web service APIs are not always the fastest option to retrieve data, although abusing of screen scraping may be against the terms of service. Another crucial aspect for achieving an "always-running experiment": When measuring, having fresh data and receiving notifications when something changes or goes wrong is essential.

B. Mimicking User Behavior

The major asset of an OSN is the vast amount of data that OSNs have acquired. Consequently, they deploy various mechanisms for detecting and preventing automated crawlers from collecting that data. As aforementioned, during our experiments we conducted various actions on Facebook, such as creating test profiles, crawling the network to obtain friend lists and photo information (URLs and tags), and downloading photos. As these actions can lead to the account being suspended, any good crawling system should incorporate measures to avoid triggering such mechanisms. A very important measure is to refrain from "flooding" the OSN with a large amount of requests in short periods of time. In addition, by configuring the crawler to conduct other (automated) actions that resemble the behavior of a human user, we were able to perform our crawling experiments with a stealthier approach and avoid triggering the security mechanisms in most cases—triggering them occasionally is unavoidable. Specifically, we had a component that logged in as our dummy accounts, and mimicked user actions such as "liking" posts of other users and posting trivial status updates.

C. Network Multi Presence

During our experiments, we needed a mechanism for triggering Facebook's SA mechanism. During manual inspection we found that the mechanism was triggered when logging in from geographical locations that had not been associated with the account in the past (i.e., from an IP address belonging to a different country). To add this functionality to our system, we resorted to ToR [6]. By enabling our system to access Facebook through the ToR network, we were able to automatically trigger the SA mechanism. Unfortunately, after a number of logins from a specific location, Facebook stopped triggering the mechanism. However, to bypass that restriction we periodically changed the ToR circuits. This demonstrates that the ToR network can be effectively used for experiments that don't relate to privacy matters, but require a virtual presence at dispersed geographical locations. The downside when using ToR is that the bandwidth is reduced substantially. This factor should be accounted for when planning for the time needed to complete experiments, or by subscribing to VPN services.

D. Data-processing Software

The face-recognition software was the core component of our data analysis phase. As such, we had designed various experiments for evaluating the efficiency of our attack, and exploring whether it poses a realistic threat. We built a custom solution, which presented the advantage of versatility as we could fine tune all algorithm parameters. In addition, as this solution lacked the accuracy of state-of-the-art solutions, we also resorted to cloud based solutions, with higher accuracy. Similar design choices hold for other data-processing tasks, given the wide variety of services[12] that provide tempting opportunities for offloading the implementation.

Using existing systems can greatly reduce implementation time and yield better results. If they are not modifiable (as with cloud-based services) one can resort to hybrid solutions of existing and custom-built components. Depending on each experiment requirements, appropriate components can be used.

Custom Solution. In our case, we used a face-detection classifier part of the OpenCV[13] toolkit. Even though there are plenty of face-detection techniques available in the literature, which are more accurate than those implemented in OpenCV, our goal was to demonstrate that face-based social authentication offered only a weak protection. Even with simple, off-the-shelf solutions, an adversary can implement an automated attack that breaks it.

Existing Cloud-based Service. We investigated whether we could employ advanced face-recognition software offered as a cloud service. We selected face.com, which offered a face-recognition platform that allowed developers to build their own applications. The service exposed an API through which developers could supply a set of photos to use as training set, and then query the service with new unknown photos for the recognition of individuals. The service allowed developers to use up to two different sets of training data, referred to as "namespaces". Each set could hold up to 1,000 users, and we found no restriction on the number of photos that could be used to train a user. A restriction was set on API usage, with 5,000 requests allowed per hour.

One major downside in using face.com was that the service was discontinued when Facebook acquired the company. Unfortunately, this happened while we were working on a follow-

[12] E.g., https://www.mashape.com/, programmableweb.com/apis/directory

[13] http://opencv.org/

up experiment, which was postponed until we found alternative cloud-based, face-recognition APIs.

VIII. Related Work

This paper is a retrospective view on our previous and current research. For similar work, we refer the reader to systematization-of-knowledge (SoK) papers. Our work is also orthogonally related to previous research on privacy in OSNs and, partially, to crowd sourcing.

SoK and Experience Papers. Retrospective and systematization studies are becoming popular in the computer-science research community. We believe that this type of publications, which provide the reader with more than a mere survey, are of paramount importance in the field of system security, because they set the ground for good and prudent experimentation practices. Notable examples of retrospective studies include, for instance [5] or [16]. Notable examples of recent systematization efforts include Rossow's paper on how to design malware experiments [24], or Zhou's work on consolidating the common characteristics of Android malware [31]. Also, specific conference tracks (e.g., in IEEE Symposium on Security and Privacy) encourage the submission of so-called systematization-of-knowledge papers.

As briefly overviewed in the remainder of this section, researchers have conducted their studies by focusing on some of the aspects that we systematize in this paper. We are not aware, however, of any systematization or retrospective work on online social networks research with a focus on system security.

Mimicking User Behavior. Stringhini et al. [26] analyzed how spammers who target social networking sites operate. They created a set of fake, realistic-looking "honey profiles" which passively waited for friend requests from other accounts. The intuition behind this approach was that once the friendship request from a spammer had been accepted, the fake profile would start receiving messages from the spammer directly in his own feed. The authors analyzed some parameters to devise heuristics that could detect anomalies in the behavior of users who contacted the fake profiles while demonstrating some type of mis-behaviour. Even though this approach of populating fake accounts with friends was effective in this case where the goal was to befriend spammers, our experiments called for befriending legitimate users. Thus, we followed an active approach of sending friend requests to other accounts.

[10] piggybacked on existing functionality of OSNs, and leveraged human social behavior to augment the efficiency of their research. Instead of directly issuing friend requests to Facebook users, they simply visited their profiles and took advantage of the fact that Facebook subsequently presented the fake accounts as recommended connections to those users. A large fraction of those users would issue friend requests back to the fake accounts, thereby allowing the researchers to operate in a stealthy manner and quickly establish links to random users.

Security Analysis of OSNs. In two similar studies, Polakis et al. [20] and Balduzzi et al. [4] demonstrated how existing functionality of an online social service can be misused for actions other than those intended for. Specifically, they used the search utilities of social networking sites as an oracle; an attacker can search for an email address, and if a user profile has been registered with that address, the profile is returned. This process allows an attacker to map email addresses to social profiles. Kontaxis et al. [14] also used the search functionality of social-networking services, this time for accessing the social graph indexes which are computed by the service. This enabled the efficient identification of potentially cloned profiles within or across OSNs. A more naïve approach would require extensively crawling the social graph to acquire the same information.

Network Multi Presence. Researchers frequently need to perform network experiments that require multiple and distributed vantage points. Apart from us, other researchers have utilized Tor [6] as well for such experiments. Antoniades et al. [3] carried out distributed network measurements through its geographically disperse topology. By explicitly selecting and routing their traffic through select overlay nodes they evaluated replication strategies of content delivery networks and investigated network neutrality violations through port blocking and traffic shaping. Alicherry et al. [2], in an effort to combat man-in-the-middle attacks, validated self-signed SSL certificates and host keys by fetching them from the respective end host through multiple alternate paths realized by distinct exit nodes.

Crowd-sourcing Platforms. Wang at al. [29] conducted a series of online interviews and surveys that investigated the types of posts which Facebook users regretted having shared. During the survey the users were asked what and why did they regret about some of posts they made, as well as what the consequences caused by these posts were. Survey participants were recruited using the AMT service.

Privacy Implications of Face Recognition. Acquisti et al. [1] investigated the feasibility of combining publicly available OSN data with off-the-shelf face-recognition technology, so as to link online (and potentially sensitive) data to someone's face in the offline world. Three experiments were conducted. In the first experiment they mined publicly-available images from Facebook profiles and attempted to map them to user profiles on one of the most popular dating sites in the United States. In the second experiment they used publicly-available images from a Facebook college network to identify students walking around the campus of a North-American academic institution. In the third experiment they inferred personal information from a subject's OSN profile in real time, after recognizing her face through an application installed on a common mobile phone. Additional personal information was then found (or inferred through data mining) online, and displayed on the phone.

IX. Conclusive Remarks

Comprehensive measurement and data-analysis work is very difficult to achieve and time consuming, especially if conducted from the users' perspective. For example, Maggi et al. in [17] wanted to measure how short URLs are used and perceived by the users. The study required 2 years to complete, and more than 6 months just to have enough users spontaneously subscribe (non-spontaneous subscriptions could bias the measurement).

Empirical works on online social networks are probably the most representative example of user-centered measurements and, as such, require time and certain aspects to be considered.

We believe that this retrospective view on our work will be useful to other researchers working on similar problems. Also, we hope that this will inspire follow-up work and provide future extensions containing more insights and lessons learned while conducting large-scale security or privacy measurements on real Web services.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. This work was supported in part by the FP7 project SysSec funded by the EU Commission under grant agreement no 257007, and by the MIUR under the FIRB2013 FACE grant. This work was also supported by the NSF Grant CNS-13-18415. Author Keromytis was also supported by (while working at) the NSF during the conduct of his work. Any opinions, fundings, conclusions, or recommendations expressed herein are those of the authors, and do not necessarily reflect those of the US Government or the NSF.

References

[1] A. Acquisti, R. Gross, and F. Stutzman, "Faces of Facebook: How the largest real ID database in the world came to be," BlackHat USA, 2011, http://www.heinz.cmu.edu/∼acquisti/face-recognition-study-FAQ/acquisti-faces-BLACKHAT-draft.pdf.

[2] M. Alicherry and A. D. Keromytis, "Doublecheck: Multi-path verification against man-in-the-middle attacks," in *SCC*. IEEE, 2009.

[3] D. Antoniades, E. P. Markatos, and C. Dovrolis, "Mor: Monitoring and measurements through the onion router," in *CPACM*. Springer, 2010.

[4] M. Balduzzi, C. Platzer, T. Holz, E. Kirda, D. Balzarotti, and C. Kruegel, "Abusing social networks for automated user profiling," in *RAID*. Springer, 2010.

[5] A. Cui and S. J. Stolfo, "Reflections on the engineering and operation of a large-scale embedded device vulnerability scanner," in *BADGERS*. ACM, Apr. 2011.

[6] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *USENIX Security Symposium*, 2004.

[7] Facebook, "Automated Data Collection Terms," http://www.facebook.com/apps/site_scraping_tos_terms.php.

[8] ——, "robots.txt," http://www.facebook.com/robots.txt.

[9] S. Hanna, L. Huang, E. Wu, S. Li, and C. Chen, "Juxtapp: A Scalable System for Detecting Code Reuse Among Android Applications," in *Proceedings of the 9th ...*, 2012.

[10] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu, "Reverse social engineering attacks in online social networks," in *DIMVA*. Springer-Verlag, 2011, pp. 55–74.

[11] D. Jacoby, "Facebook Security Phishing Attack In The Wild," http://www.securelist.com/en/blog/208193325/Facebook_Security_Phishing_Attack_In_The_Wild.

[12] A. Kapravelos, I. Polakis, E. Athanasopoulos, S. Ioannidis, and E. P. Markatos, "D(e — i)aling with voip: Robust prevention of dial attacks," in *ESORICS*, 2010, pp. 663–678.

[13] H. Kim, J. Tang, and R. Anderson, "Social authentication: harder than it looks," in *FC*. Springer, 2012.

[14] G. Kontaxis, I. Polakis, S. Ioannidis, and E. P. Markatos, "Detecting social network profile cloning." in *SESOC*. IEEE, 2011, pp. 295–300.

[15] B. Krishnamurthy and C. E. Wills, "On the leakage of personally identifiable information via online social networks," in *WOSN*. ACM, 2009, pp. 7–12.

[16] S. Li, "Juxtapp and DStruct: Detection of Similarity Among Android Applications," Master's thesis, EECS Department, University of California, Berkeley, May 2012.

[17] F. Maggi, A. Frossi, S. Zanero, G. Stringhini, B. Stone-Gross, C. Kruegel, and G. Vigna, "Two years of short URLs internet measurement: security threats and countermeasures," in *WWW*. International World Wide Web Conferences Steering Committee, 2013, p. 861872.

[18] Natala@AWS, "MTurk CENSUS: About how many workers were on Mechanical Turk in 2010?" https://forums.aws.amazon.com/thread.jspa?threadID=58891.

[19] I. Polakis, P. Ilia, F. Maggi, M. Lancini, G. Kontaxis, S. Zanero, S. Ioannidis, and A. Keromytis, "Faces in the distorting mirror: Revisiting photo-based social authentication," in *CCS*. ACM, 2014.

[20] I. Polakis, G. Kontaxis, S. Antonatos, E. Gessiou, T. Petsas, and E. P. Markatos, "Using social networks to harvest email addresses," in *WPES*. ACM, 2010.

[21] I. Polakis, M. Lancini, G. Kontaxis, F. Maggi, S. Ioannidis, A. Keromytis, and S. Zanero, "All your face are belong to us: Breaking facebook's social authentication," in *ACSAC*. ACM, 2012.

[22] ResearchMatch, "ResearchMatch Metrics," https://www.researchmatch.org/index_pubsitemetrics.php.

[23] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson, "Who are the crowdworkers?: Shifting demographics in mechanical turk," in *CHI*. ACM, 2010, pp. 2863–2872.

[24] C. Rossow, C. J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, and M. van Steen, "Prudent Practices for Designing Malware Experiments: Status Quo and Outlook," in *SSP*. IEEE, 2012.

[25] S. Sheng, M. Holbrook, P. Kumaraguru, L. F. Cranor, and J. Downs, "Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions," in *CHI*. ACM, 2010, pp. 373–382.

[26] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *ACSAC*. ACM, 2010, pp. 1–9.

[27] Twitter, "REST API v1.1 Resources," https://dev.twitter.com/docs/api/1.1.

[28] ——, "Twitter Blog - Election Night 2012," http://blog.twitter.com/2012/11/election-night-2012.html.

[29] Y. Wang, G. Norcie, S. Komanduri, A. Acquisti, P. G. Leon, and L. F. Cranor, ""I regretted the minute I pressed share": a qualitative study of regrets on Facebook," in *SOUPS*. ACM, 2011.

[30] P. Warden, "How i got sued by Facebook," http://petewarden.typepad.com/searchbrowser/2010/04/how-i-got-sued-by-facebook.html.

[31] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in *SSP*. IEEE Computer Society, May 2012.