Design and Security Assessment of Usable Multi-factor Authentication and Single Sign-On Solutions for Mobile Applications

A Workshop Experience Report

Roberto Carbone, Silvio Ranise, and Giada Sciarretta(✉)

Security and Trust, FBK, Trento, Italy
{carbone,ranise,giada.sciarretta}@fbk.eu

Abstract. In this interactive workshop we focused on multi-factor authentication and Single Sign-On solutions for mobile native applications. The main objective was to create awareness of the current limitations of these solutions in the mobile context. Thus, after an introduction part, the participants were invited to discuss usability and security issues of different mobile authentication scenarios. After this interactive part, we concluded the workshop presenting our on-going work on this topic by briefly describing our methodology for the design and security assessment of multi-factor authentication and Single Sign-On solutions for mobile native applications; and presenting a plugin that helps developers make their mobile native application secure.

1 Introduction

This paper is a report of the workshop "Secure and Usable Mobile Identity Management Solutions: a Methodology for their Design and Assessment" presented at the *13th International IFIP Summer School 2018 on Privacy and Identity Management - Fairness, accountability and transparency in the age of big data* held in Vienna, Austria.

Context. We focused on the design and security assessment of solutions for mobile native applications (hereafter native apps) with two features: Single Sign-On (SSO) and Multi-Factor Authentication (MFA); these two features are extremely important indeed: SSO allows users to access multiple apps through a single authentication act performed with an identity provider (IdP), for example Google or Facebook; while a MFA is a procedure that enhances the security of an authentication process by using two or more authentication factors (e.g., a password combined with the use of a fingerprint). A good design choice is to combine these features to have a good balance between usability and security.

While there exist many secure MFA and SSO solutions for web apps, their adaptation in the mobile context is still an open challenge. The majority of mobile MFA and SSO solutions currently used are based on proprietary protocols

Published by Springer Nature Switzerland AG 2019
E. Kosta et al. (Eds.): Privacy and Identity 2018, IFIP AICT 547, pp. 51–66, 2019.
https://doi.org/10.1007/978-3-030-16744-8_4

and their security analysis lacks standardization in the structure, definitions of notions and entities, and specific considerations to identify the attack surface that turns out to be quite different from well understood web scenarios. This makes a comparison among the different solutions—in order to choose the proper solution for a specific scenario—very complex or, in the worst case, misleading. Due to this lack of specifications and security guidelines, designing a mobile MFA and SSO solution from scratch is not a simple task; and as its security depends on several trust and communication assumptions, in most cases, could result in a solution with hidden vulnerabilities. In addition, it is necessary to take into account the legal aspects of the country where the MFA and SSO solution will be deployed. However, when innovative solutions are analyzed it is not an easy task to understand which legal obligations follow.

The main goal of the workshop was to make participants aware of the current security and usability issues of MFA and SSO solution in the mobile context. Participants were first introduced to the context and then, through the use of exercises we openly discussed several illustrative scenarios. Finally, we described our methodology for the design and security assessment of mobile MFA and SSO solutions. The design space is characterized by the identification of: *(i)* national (e.g., Sistema Pubblico di Identitá Digitale - SPID for Italy [3]) and European (e.g., electronic IDentification Authentication and Signature - eIDAS [15]) laws, regulations and guideline principles that are particularly relevant to digital identity; *(ii)* a list of security and usability requirements that are related to authentication solutions; *(iii)* a set of implementation mechanisms that are relevant to authentication and authorization on mobile devices and provide an easy way to satisfy the requirements in *(ii)*. To validate our approach, we applied it to a number of real-world scenarios that represent different functional and usability requirements. In this workshop, we applied our methodology to a real use-case scenario (called *TreC*) that supports the usage of mobile health apps. TreC (acronym for "Cartella Clinica del Cittadino", in English Citizen's Clinical Record) is an ecosystem of services that supports doctors and patients in the health-care management, by enabling all citizens living in the Italian Trentino Region to access, manage and share their own health and well-being information through a secure access (currently used by more than 80.000 patients).

Workshop Objectives. The main objectives of the proposed workshop were the following:

- to enable the audience to acquire the basic notions and the state of the art of MFA and SSO solutions for native apps;
- to create awareness of usability and security problems together with legal provisions related to authentication in mobile computing;
- to provide an overview of the techniques commonly used to analyze the security of an authentication solution;
- to perform an experimental evaluation of security and usability of MFA solutions for native apps.

Expected Contributions from the Audience Members. To raise the participants' awareness of the current possible limitations on usability and security of mobile MFA and SSO solutions, several questions and exercises were discussed together. The information extracted from the discussion has been useful in two ways. On the one hand, we were able to validate our hypothesis on usable solutions and to understand which security level is perceived. On the other hand, we were able to evaluate our methodology asking feedback to possible user.

Intended Audience, Including Possible Assumed Background of Attendees. The workshop was oriented to academic researchers, (PhD) students, security experts from industries that work on or want to approach the field of identity management. The attendees did not require a specific background on authentication to follow the main part of our workshop, as our step-by-step teaching approach enabled them to grasp the information presented even if some of the concepts were new or not consolidated. A dozen technical (IT security) and legal researchers took part to our workshop. The slide of the workshop are available at https://st.fbk.eu/workshop-ifipsc-18.

Paper Structure. In Sect. 2, we describe the content of the workshop on MFA and SSO solutions for native apps. Section 3 details the workshop structure and the assigned exercises. In Sect. 4, we present the exercises and discuss the outcomes. Finally, in Sect. 5 we discuss some lessons learned and describe our on-going work on this topic.

2 Content of the Workshop

We make a large use of our digital identities in our everyday life, from accessing social apps to security critical apps like e-health or e-banking apps. Underlying these transactions there is the exchange of personal and sensitive data, which could be exploited by a malicious intruder to impersonate or even blackmail a user. For this reason, many Identity Management (IdM) solutions have been designed to protect user data. In general, IdM refers to different aspects of the digital identity life-cycle (e.g., the creation and the provision of identities, password management and so on). In this workshop, we focused on the aspects related to authentication.

Password-Based Authentication. A common authentication mechanism is the password-based authentication, however its use is resulting in many attacks (e.g., identity theft). There are two main reasons. First, users are very bad in inventing and remembering passwords. [11] shows the list of the top 100 worst passwords of 2017, where at the first place there is "123456". This password is very easy to remember but it is also easy to crack; attackers have rainbow tables and dictionaries that contain this kind of credentials. Second, users re-use their passwords on several services: as reported in [9], more than the 54% of people use only 5 or fewer different passwords across their entire online life. This means

that, if their credentials are compromised, for example after a guessing attack (easily performed by an hacker if the password selected is one of the 100 worst passwords [11]) then the attacker can access all the user data in different online services.

There exist "complexity tips" which allow users to choose proper passwords. For example, though being nine digits long, a password such as '123456789' could be instantly and easily cracked. According to former NIST recommendations, properly complex passwords should contain lower- and upper-case letters, as well as special symbols and numbers, for them to be secure enough that it could take years to crack them (via brute-force or dictionary attacks). However, the current NIST guidelines [6] recommend to follow an entirely different scheme, i.e. a good password should be made of a random, long phrase.

Password-Based Authentication and SSO. To permit the user to choose a complex password and access different services, an advisable design choice is to combine the password-based authentication with a Single Sign-On (SSO) solution. SSO allows users to access multiple apps through a single authentication act performed with an IdP, for example Google or Facebook. A common practice is to adopt the state-of-the-art standards, like SAML 2.0 [8] and OpenID connect [20] (OIDC). SAML 2.0 is pervasively used in the corporate environment, while OIDC is mainly used in social apps. There are two main advantages of using SSO. First, users do not need to register with an app to access it. Thus the user can choose a single complex password (providing usability and security). Second, if a user has already an active login session with an IdP, then she can access new apps without entering her IdP credentials anymore (providing usability). A SSO security drawback is that users are using only one password to access many services. Again, the user could select a more complex password, but still there is a single point to failure.

Multi-factor Authentication and SSO. A better design choice is to combine SSO with MFA solutions. Where a MFA is a procedure that enhances the security of an authentication process by using two or more authentication factors, such as combining a password that is something you know, with a hardware token that is something you have or the use of a fingerprint that is something you are.

Usually a MFA procedure requires the generation of a OTP (One Time Password). That is an una-tantum code that proves the possession of the OTP generator and optionally, if protected by a PIN, proves the knowledge of the PIN as well. There are different OTP generation approaches, in this workshop we focus on the Time-based OTP approach, where the OTP is generated starting from the current time of the operation and a secret key shared between the OTP generator app and the IdP. IdP must validate this value: only OTPs that fall into a short temporal range are accepted.

There are many MFA solutions on the market, and some of them are based on FIDO [5]. FIDO is a standard for password-less and MFA authentication that allows online services to augment the security of their existing password-based

solution by adding a MFA procedure. To provide a FIDO two-factor solution, the organization must provide to its users a physical FIDO U2F device, like a USB key.

MFA and SSO Solutions for Native Apps. In this workshop, we focused on the design and security assessment of solutions for native apps with two features: SSO (for usability) and MFA (for increasing security). We focused on native apps—which differ from browser-based mobile apps as they are not accessed through a browser but they need to be downloaded from a marketplace—as the market is pointing on their use. Think about the many times you are suggested to download the app while you are navigating it in the browser; or the limitations that you can have from a browser-based version of an app. For example, this is the case of the browser-based version of TripAdvisor that provides less functionalities compared to the native app, such as the management of the reviews: users can read reviews on the browser version, but if they want to contribute on one review they have to download the native app.

As we will detail in Sect. 4, the known standards and solutions currently available for browser-based authentication (e.g., SAML or OIDC) cannot be easily reused in the mobile context, as browser-based and native apps are based on different security assumptions. Even if these are very good solutions there are still some limitations. Being proprietary protocols they cannot be customized and they do not necessarily satisfy the security requirements of a company. For example, an identity used in the social network solution is self-declared by the user. And so, it cannot be used by a company which needs to be sure of the real identity of the user. A first attempt of designing a solution for mobile authentication was carried on by big companies (e.g., Google and Facebook) that have designed their own solutions based on their security assessment. At the same time, the OAuth working group has released some guidelines. The current best practice was released in 2017. The solution proposed is called "OAuth 2.0 for Native Apps" [18]. Even if it is a good starting point, it does not cover some aspects. For example, it does not mention how to extend the protocol to support MFA or more complex environments, where for example different standards are used. Thus, in some specific cases and based on the requirements, a company is required to design a new ad-hoc solution.

The Importance of a Careful Design Phase. Designing a security protocol from scratch is not a simple task, as many aspects must be taken into account, such as how to establish trust, how to choose the right communication channel or how to evaluate the compliance of the designed solution with the current legal obligations, thus it is not recommended. Moreover, after the design, it is not simple to choose the right method to evaluate the corresponding security. Given all these aspects, it is clear that the design phase is not trivial and wrong design choices could lead to serious security and usability problems.

An example of a wrong design choice is the use of SMS as a second-factor authentication. This authentication method consists of the following steps: first the user has to enter her credentials into the app, then she will receive an SMS

containing a OTP, and finally this OTP is entered by the user in the app. NIST [7] points out that this solution could be vulnerable to two kinds of attacks:

Social Engineering. "An out of band secret sent via SMS is received by an attacker who has convinced the mobile operator to redirect the victim's mobile phone to the attacker" (e.g., using SIM swap [14]).

Endpoint Compromise. "A malicious app on the endpoint reads an out-of-band secret sent via SMS and the attacker uses the secret to authenticate".

Even if these attacks are well known in the security community, there are still many companies that are using this authentication method. This is causing the spread of many security breaches. For example, this is the case of the social network platform Reddit, attacked in August, 2018. In [10], the Reddit security experts specified that the attack was related to the use of SMS and that they are now moving to a token-based second factor authentication method. Another example is described in [2], where a user was victim of two SIM hijacking attacks and now he is suiting the telecommunication company for a total of 224 million dollars. This example clearly demonstrates how a wrong design choice could damage not only the end-user but also the company.

Our Methodology. Given that designing a new security protocol from scratch is not an easy task and could result in a solution with hidden vulnerabilities, we have contributed with the definition of: a *reference model* for MFA and SSO for native apps, and a *methodology* to assist a designer in the customization of our model and in the analysis of its security and usability.

Our reference model is inspired by the Facebook solution [4] and OAuth 2.0 for native app [18]. We have extended these solutions in a way that they can be used by any IdP willing to provide its own SSO solution, meaning that the resulting SSO does not necessary leverage on identity provided by social IdP, and (optionally) a MFA. Currently, our models support two different OTP generation approaches: TOTP and Challenge-Response. Full details about the reference model based on TOTP can be found in [21].

Together with the reference model we have defined a methodology to assist a designer in the customization of our reference model and in the analysis of the resulting security and usability. In the first phase, we ask the designer to clarify the application scenario by filling a table that we provide (specifying the entities involved, the type of data that will be processed and which are the authentication requirements). Given this table, in the customization phase we are able to instantiate our model for this specific scenario and we provide as output a message sequence chart of the flow and a set of assumptions and security goals. These values are then given as input to the security analysis phase. We provide a semi-formal and a formal analysis. The output of this phase is a security analysis report. If some serious attacks are found then the designer has to go back in the customization phase and change the design otherwise the designer can proceed with the last phase. In the usability analysis phase we are asking to validate the usability satisfaction. If no problems are reported then the

final solution is generated; otherwise the designer has to go back to the definition of the requirements and refine the design accordingly.

To validate our methodology, we have applied it to different real-world scenarios which consider different authentication and usability aspects. During this workshop we had detailed the e-health scenario TreC.

3 Structure of the Workshop

To raise the participants' awareness of the current limitations on usability and security of mobile MFA and SSO solutions, together with the background context described in Sect. 2, several questions (labeled with 🗪) and exercises (labeled with ✎) were discussed together. Figure 1(a) shows the background and current position of the participants (divided in two groups) and the outline of the exercises. The workshop followed the following structure:

Introduction and Problem Statement. In this introductory part we provided participants with the background described in Sect. 2 and we pointed out which are the current limitations related to the development of usable authentication solutions that are also secure in the mobile context.

🗪 *Browser-based vs Mobile Authentication.* After the description of the browser-based OIDC standard, we asked the participants if—in their opinion—this solution (and more in general, any browser-based solutions) can be reused also in the mobile context. We decided to ask this question to evaluate the participant's awareness and understanding of the differences between a mobile and a browser-based solution and the fact that we cannot easily reuse solutions that are developed in one context in another.

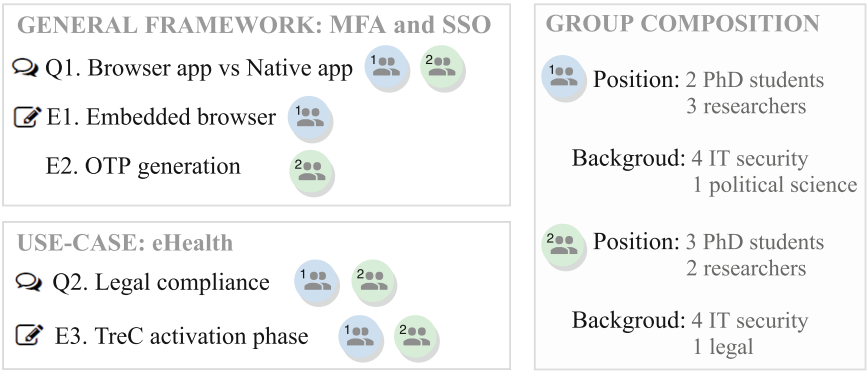
Design Choices: Security and Usability Problems. Designing a security protocol from scratch is not easy and the design phase is very important in terms of striking the right balance between security and usability. To raise this warning, during the workshop we asked participants to identify which are the security and usability problems related to two wrong design choices:

✎ *User Agent Choice: embedded browser.* This exercise is related to the choice of using an embedded browser as user agent. So we asked the participants to evaluate the usability and security of a solution where users enter their credentials in an embedded browser, namely a browser that is managed by an app.

✎ *OTP Choice: app that shows the OTP value to the user.* During this exercise, we asked the participants to evaluate the usability and security of an OTP generator app that shows the OTP value on the smartphone screen.

Methodology Overview: TreC Scenario. In this part, we described our methodology for the design and security assessment of mobile authentication solutions, applied directly to a real-world use case scenario, called TreC.

🗪 *e-Health Legal Compliance.* Being TreC a personal health record platform, we briefly mentioned the Italian legal aspects concerning health data, and more in general to sensitive data. Then, with the aim of having a broader



(a) Question and exercise structure. (b) Group Composition.

Fig. 1. Interactive workshop structure.

view on the legal aspects, we asked the participants to present the legal obligations of their country.

Usability Discussion on TreC. In relation to the TreC solution, we discussed with the participants some usability problems that resulted from proposing our solution to patients.

TreC activation phase. We asked the participants to suggest some changes in order to simplify the TreC activation phase taking into account security.

Conclusions and On-Going/Future Work. Finally we summed up the main points of the workshop and presented our on-going and future work.

The answers and the discussions are reported in the following section.

4 Outcomes of the Exercises

In this section, we report the solutions to the exercises and discussions introduced in Sect. 3. To promote an interdisciplinary approach, we divided the participants in two groups based on their backgrounds and current position (see Fig. 1(b)). Each group included four men and a woman.

4.1 Browser-Based vs Mobile SSO Solutions

Q1. Can we use browser-based authentication and SSO solutions for native apps? We gave 5 min to discuss the problem and then we asked for the individual answers.

Participant's Answers: 5 "yes" and 5 "no". With 2 participants that voted "yes" saying that actually they would prefer to vote for "it depends", clarifying that it depends on the scenario and the security level required.

Our Answers: The use of a browser-based authentication protocol in the mobile context requires a detailed understanding about the differences between the two scenarios [13,22], and at the end it is pretty clear that we need to design a new flow ad-hoc for the native apps. Let us have a look at the differences.

The first difference is the Service Provider (SP) type: SP is not an app running in the browser but it is a native app. So we have to consider all the vulnerabilities related to a mobile platform.

Secondly, the User Agent (UA) that is used by the user to interact with the SP could be of different types: it could be a browser embedded inside the SP app, or an external browser that is installed in the user smartphone, or even an app released by the OIDC provider. So, we must take into consideration that now the SP app and the UA could not be played by the same entity as was in the browser case.

In addition, the redirection mechanisms between the OIDC provider and the app are different. Indeed, in a browser redirection you can uniquely identify the SP using its hostname. This ability is not always available in the mobile case as you are required to redirect to a specific app in the user's smartphone.

Finally, in the mobile case, the use of a SP backend is optional (you can have a native app that does not require a backend), so we have to adapt the flow by directly managing the authentication from the mobile. So in this case, we cannot use the client secret for authenticating the SP since in a mobile device we cannot store a secret as all the stored values are readable, at least by the owner of the smartphone.

For all these reasons, the reuse of available browser-based solutions in a mobile context is not obvious and it is necessary to redesign them taking into account the differences highlighted above.

Evaluation: The question was not fully clear and some participants were not able to answer.

4.2 Wrong Design Choices

In some specific cases and based on the requirements, a company or an organization could be forced to design a new authentication solution to provide secure mobile authentication solutions to their employees. However, designing a security protocol from scratch is not a simple task, as many aspects must be taken into account. In this section, we report the two scenarios that have been proposed to the workshop participants to highlight examples of wrong design choices. We proposed two exercises and asked Group 1 to tackle Exercise 1 (*E1*) and Group 2 to solve Exercise 2 (*E2*) in parallel. We allowed 10 min to elaborate a solution to each group.

✎ User Agent (UA) Choice: Embedded Browser

E1. We asked the participants to evaluate the usability and security of a solution where users enter their credentials in a browser that is managed by a native app

(so called embedded browser). In detail, we asked the members of Group 1 to focus on the following questions:

1. How does an embedded browser work?
2. Are there any security issues?
3. How would you rate the user experience when accessing multiple apps?

Participant's Answers:

1. An embedded browser is a component inside your app that opens a website URL.
2. A native app has full control of the embedded browser, so if you are typing a password, the attacker can read it.
3. Not so good. Since an embedded browser is a separate browser-instance for all the native apps, users have to re-enter their password for all the native apps.

Our Answers:

1. An embedded browser is defined in [18] as "a user-agent hosted inside the native app itself (such as via a web-view), with which the app has control over to the extent it is capable of accessing the cookie storage and/or modifying the page content". The relevant bit from the point of view of security is that a native app is in control of the embedded browser.
2. The use of this type of browser is widely discouraged, as there is a loss of isolation between the app and the browser [19]. If the app is malicious, then it can steal the user credentials or change the authorization permissions. This is an example of a JavaScript added by a malicious app to steal user credentials:

```
webView.evaluateJavascript(
''(function() {return document.getElementById('pwd').value;})();'',
new ValueCallBack<String>() {
@Override public void onReceiveValue(String s){
Log.d(''WebViewField'',s);
}
});
```

3. An additional limitation when choosing an embedded browser is that it does not provide a SSO experience. Indeed, if the browser is integrated within the app, then the login session information is stored in (and only accessible to) the app and it is therefore not available to other apps. This forces the user to re-enter credentials even if she has an active login session with an IdP. This is a frustrating experience, especially due to the small-virtual keyboard of a smartphone.

Evaluation: The participants were able to answer in the correct way.

OTP Choice: App that Shows the OTP Value to the User

E2. This exercise is related to the choice of using a native app for showing an OTP value to the user.

1. How would you rate the user experience when accessing a native app?
2. Is there any security issue?
3. List one or more OTP choice alternatives.

Participant's Answers:

1. We thought that usability really takes a hit if you have to switch between your native app and the OTP app all the time. This is the main thing. The usability can also be affected if you have multiple devices lost or broken.
2. Malicious apps trying to exfiltrate the OTP code or shoulder-surfing attacks (you are just behind someone and take a look at the OTP code that can be memorized or can be picked using a mobile camera).
3. Hardware tokens.

Our Answers:

1. Moving from an app to another is burdensome for the user in terms of time and difficulty.
2. To avoid the burden of remembering the OTP value, some apps (e.g., MySielteID[1] app of the Sielte SPID IdP) have a button for coping the OTP value. This is a serious security issue as the clipboard can be accessed by any app installed in the smartphone so that malicious apps can easily steal the OTP that has been copied.
3. An alternative choice is the use of a solution that does not ask the user to enter the OTP value, but after the PIN input, the OTP value is sent to the IdP server in a transparent way (namely, without any involvement of the user). The other alternative is the use of external OTP generators, such as FIDO keys or eID smartcard with capability of implementing a challenge-response OTP approach (using NFC and a smartphone as a card reader). The advantage of using an eID card over other external hardware tokens is that usually users bring it with them to permit an in-person identification; thus we can assume that an eID card is always available to the user also for online access.

Evaluation: The participants were able to answer in the correct way.

4.3 e-Health Legal Compliance

Being TreC a personal health record platform, we briefly mentioned the Italian legal aspects concerning health data. Then, with the aim of having a broader view on the legal aspects, we asked the participants to present the legal obligations of their country.

[1] https://play.google.com/store/apps/details?id=it.company.sielte.

At the time of running the workshop, when dealing with sensitive health data—that are a particular type of personal data—in Italy, we had[2] to follow the Data Protection Code [16]. [16] says that the data controller shall adopt the minimum security measures in order to protect these data. More technological details can be found in CAD [12] that is the Italian code for the public administration. In particular, [12] specifies which digital identities can be used in this context: CNS (a smartcard used to access online services of the public administration), CIE 3.0 (the Italian electronic identity card) or the Italian national ID scheme called SPID (from the second level up).

Q2. Which legal obligations do you have to follow when dealing with e-health data in your country?

Participants' Answer (from Germany): We apply the GDPR, especially for sensitive health data. The basic approach is: you are not allowed to process any data if you cannot provide the level of security that is necessary to protect the data; it is not a minimum standard but it must be the state-of-the-art. Additionally to the GDPR there are also rules/guidelines specific for health data.

Extra Discussion: we report here different interesting discussions that came up in relation to this question.

The first is a discussion on the minimum security measures required in [16]. We explained that the Annex B of [16] specifies a set of specific security measures, such as: sensitive data must be protected with an authentication method with passwords of at least 8 characters. [16] was in force pre-GDPR and we agree with the participants on the fact that the GDPR approach is the opposite: they do not suggest any kind of measures; they should be state-of-the-art, and it should be proved that companies have done their best to comply and provide enough security.

Then, we have discussed who is in charge of managing health data. In Italy, each region is responsible for applying and developing solutions for healthcare. In Germany, it depends of what you do. For example, hospitals have to follow the federal state law, while the health insurance is the same for all the country.

Finally, we briefly discussed about the privacy concerns arising by the adoption of a SSO solution. A participant draw our attention to the fact that the usage of a SSO protocol every time a user wants to log-in creates a single point of knowledge that can become a major threat to privacy. This problem is particularly acute when considering the so-called "consumer SSO": what happens to consumers data when they enable access to other applications or accounts from Facebook, Gmail, LinkedIn, Twitter, or a host of other providers? Indeed,

[2] During the preparation of this workshop, the Italian government was in the process of adopting the EU General Data Protection law (GDPR [17]), with some delay compared to other member states due to the national elections. Now, [16] was amended by the decree adapting the national legal system to the GDPR 2016/679 (Legislative Decree No. 101 of 10 August 2018).

this enables service providers with the capability of monitoring and collecting information on consumers habits and preferences as they browse the Web. While acknowledging the importance of these and related privacy concerns, we consider them outside the scope of the workshop which focuses on the trade-off between security and usability of SSO solutions. We just observe how security and privacy may be contentious even in SSO solutions: the capability of tracking and profiling users can be used by an identity provider to spot when an attacker is trying to impersonate a legitimate user. This is known as behavioral authentication and is provided by, e.g., Google which alerts if a user account is accessed from a location which is not among the usual ones.

Evaluation: Different interesting discussions arose from this question.

4.4 ✎ TreC Activation Phase

TreC activation phase is performed by the patient, only once and after she download the OTP-PAT app. It is performed partially on her laptop and partially on her smartphone:

- On her laptop, patient logs in using her CNS (the card is read by desktop smartcard reader), and generates a temporary code (it lasts 5 min).
- On her smartphone, patient downloads the OTP-PAT app using an official marketplace. Then, she enters the temporary code together with her credentials into OTP-PAT. If the login is successful, the activation phase is completed by patient with the creation of a PIN code.

As a consequence of this phase, OTP-PAT obtains two values: a token that is used as a session token in place of the user credentials to provide a SSO experience; and a seed value—stored encrypted with the PIN code selected by patient during this phase—that is used to generate OTPs.

For the TreC scenario, we have performed a pilot involving a controlled set of patients. Regarding usability, we found out that the activation phase is considered too complex. The main reason was the use of a smartcard reader that for being used, needs the installation of a specific software, which sometimes does not work properly. In addition, users were annoyed by the requirement of choosing complex passwords inside the mobile and they tend to easily forget them.

E3. What would you suggest to change in order to simplify the activation phase taking into account security? Note that the activation phase must provide a good level of assurance on the real identity of the user. In the previous solution this was implied by the use of a smartcard and the generation of an activation code, specific for the particular installation of the app. We gave 10 min to debate this issue.

Participant's Answers: they came up with two ideas:

1. Use one of the existing identity infrastructures. For example, in Austria they can use an identification method provided from the post system or in Sweden, if you have a bank account, then you can request a BankID that you can use for accessing online services;
2. Perform a face-to-face authentication and then send an activation code via email.

In addition, they observed that the objection on the password complexity is just an educational thing: people do not understand the security behind this design choice.

Our Answer: We change the activation phase as follows:

- On her laptop, patient logs in with one of the authentication solutions that are available (for example SPID) by using a high level of assurance (e.g., requiring a second factor authentication) and obtains a QR code.
- On her smartphone, patient downloads the OTP-PAT app using an official marketplace. Then, she scans the QR code using OTP-PAT app and enters a temporary code obtained on her email. If the login is successful, then the activation phase is completed by patient with the creation of a PIN code.

As an alternative, if some users do not want to activate the app online, they can go to one of the office of the healthcare organization, prove their identity by using an identity card (in-person identification) and finally they receive a printed version of the QR code. Note that this solution avoids the need to manually enter username and password since the identity information of the patient is inside the QR code obtained after a strong authentication (in-person or online).

Evaluation: The participants were able to propose valid alternatives.

5 Lesson Learned and On-Going Work

During this workshop, we had the opportunity to discuss our work with researches from technical and legal backgrounds. On the one hand, our main goal was to create awareness of usability and security issues related to authentication in the mobile context. On the other hand, the information extracted by their answers and discussions gave us the opportunity to validate our hypothesis on the usability and security of the solution that we have proposed for the TreC scenario.

Regarding the exercise session, we observed a high interest and participation. The participants were able to answer in the correct way to almost all questions. Only the first question related to the possibility to re-use a browser-based solution in the mobile context was not fully understood. The problem was that—being us security experts—we intended that question from a security perspective, while in a broader context this was unclear.

In the workshop, we made an effort to clearly define the security and usability problems in each one of the proposed exercises and questions. Indeed, this is rarely the case in a real-world scenario whereby striking the best possible balance between security and usability turns out to be a daunting task. This is so because of tight development schedules, focus on functionalities rather than security-by-design, and the unawareness of developers about the security implications of certain implementation decisions. The combined effect of these factors results in the presence of severe (and exploitable) vulnerabilities in a large amount of applications. To alleviate this state of affair, we developed a plug-in for the automated synthesis of secure authentication solutions in mobile applications in the context of the EIT Digital activity "Security Tools for App Development" [1] (STAnD). The basic idea underlying STAnD is to provide native app developers with tools that help them to take into consideration more and more security aspects. STAnD will be composed by a plugin for code hardening and a wizard that allows developers to configure and customize their solutions: developers are presented with a series of choices and then the code is automatically produced according to their answers. A second feature of STAnD will be the possibility to validate the app code by submitting the APK to a managed service that will check if there are security problems (e.g., the need for obfuscating the code related to the handling of a key).

Acknowledgments. This work has partially been supported by the Activity no. 18163, "API Assistant - Automated security assessment of 3rd party apps for the API economy", funded by the EIT Digital.

References

1. API Assistant: automated security assessment of 3rd party apps for the API economy. https://st.fbk.eu/projects/api-assistant/
2. AT&T Sued Over $24 Million Cryptocurrency SIM Hijack Attacks. https://www.databreachtoday.com/att-sued-over-24-million-cryptocurrency-sim-hijack-attacks-a-11365
3. DPCM of 24 October 2014, (SPID). http://www.agid.gov.it/agenda-digitale/infrastrutture-architetture/spid
4. Facebook: Getting started with the Facebook SDK for Android, May 2017. https://developers.facebook.com/docs/android/getting-started/facebook-sdk-for-android/
5. FIDO. https://fidoalliance.org/about/what-is-fido/
6. NIST Special Publication 800–63B: Appendix A - Strength of Memorized Secrets. https://pages.nist.gov/800-63-3/sp800-63b.html#appendix-astrength-of-memorized-secrets
7. NIST Special Publication 800–63B: Section 8.1: Authenticator Threats. https://pages.nist.gov/800-63-3/sp800-63b.html#81-authenticator-threats
8. Profiles for the OASIS: Security Assertion Markup language (SAML) V2.0. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf
9. Telesign Consumer Account Security Report. https://www.telesign.com/resources/research-and-reports/telesign-consumer-account-security-report

10. We had a security incident. Here's what you need to know. https://www.reddit.com/r/announcements/comments/93qnm5/we_had_a_security_incident_heres_what_you_need_to/
11. Worst passwords of 2017 - Top 100. https://s13639.pcdn.co/wp-content/uploads/2017/12/Top-100-Worst-Passwords-of-2017a.pdf
12. CAD: Codice dell'Amministrazione Digitale - D.Lgs.n. 82/2005 (2014). http://www.altalex.com/documents/codici-altalex/2014/06/20/codice-dell-amministrazione-digitale
13. Chen, E., Pei, Y., Chen, S., Tian, Y., Kotcher, R., Tague, P.: OAuth demystified for mobile application developers. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2014). https://doi.org/10.1145/2660267.2660323
14. Cranor, L.: Your mobile phone account could be hijacked by an identity thief. https://www.ftc.gov/news-events/blogs/techftc/2016/06/your-mobile-phone-account-could-be-hijacked-identity-thief
15. European Parliament: eIDAS. http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014R0910&from=EN
16. Garante Privacy: Personal Data Protection Code. Legislative Decree no. 196 of 30 June 2003 (2003). http://www.privacy.it/archivio/privacycode-en.html
17. General Data Protection Regulation: Regulation EU 2016/679. http://www.eugdpr.org
18. OAuth Working Group: OAuth 2.0 for Native Apps (2018). https://tools.ietf.org/html/rfc8252
19. Luo, T., Hao, H., Du, W., Wang, Y., Yin, H.: Attacks on WebView in the android system. In: Twenty-Seventh Annual Computer Security Applications Conference, ACSAC 2011, Orlando, FL, USA, 5–9 December 2011, pp. 343–352 (2011). https://doi.org/10.1145/2076732.2076781
20. OpenID Foundation: OpenID Connect Core 1.0. (2014). http://openid.net/specs/openid-connect-core-1_0.html
21. Sciarretta, G., Carbone, R., Ranise, S., Viganò, L.: Design, formal specification and analysis of multi-factor authentication solutions with a single sign-on experience. In: Proceedings of the 7th International Conference on Principles of Security and Trust (POST), pp. 188–213 (2018). https://doi.org/10.1007/978-3-319-89722-6_8
22. Shehab, M., Mohsen, F.: Towards enhancing the security of OAuth implementations in smart phones. In: IEEE International Conference on Mobile Services (MS), pp. 39–46 (2014). https://doi.org/10.1109/MobServ.2014.15