Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Understanding and mitigating OpenID Connect threats

*Jorge Navas, Marta Beltrán**

Department of Computing, ETSII, Universidad Rey Juan Carlos, Móstoles, Madrid 28933, Spain

ARTICLE INFO

Article history:
Received 9 November 2018
Revised 8 February 2019
Accepted 2 March 2019
Available online 11 March 2019

Keywords:
Authentication
Federated Identity Management
Identity providers
OpenID Connect
Threat modelling

ABSTRACT

Federated Identity Management (FIM) specifications have been massively adopted in web, cloud and mobile environments during the last years. Facebook, Google, Twitter, Linkedin, Amazon, Microsoft or Salesforce, to mention only some significant examples, are actively supporting standards such as OAuth or OpenID Connect, becoming in many cases identity providers. This last specification is able to solve identification, authentication, authorization and accounting (IAAA) with one unified flow and two tokens; making logging easier, safer and more secure when compared with previous solutions. Naturally, experts are predicting a widespread adoption of OpenID Connect in the next years not only in web, cloud or mobile environments but also in Fog Computing, IoT or Smart Places. To better understand the threats that this specification poses, this work presents a thorough threat modelling of OpenID Connect core specification and its current implementations. Threats for security and privacy and up to 16 different attack patterns have been identified, analysed and described. Furthermore, possible mitigations and solutions are proposed for both, specification and implementation aspects.

1. Introduction

The nature of identity is changing to the point of considering that identity is the new money. Main technology providers have been trying for years to leverage existing user accounts in order to provide new services regarding identity and access management while users has been looking for effortless solutions allowing them to consume different services from different devices with a Single Sign-On approach (SSO).

Federated Identity Management (FIM) allows end users (EU) to access different resources, applications and services through a single Identity Provider (IdP), avoiding the need of having an account (with its related password and/or authenticators) for each resource, application or service. Resource, application and service providers are clients or Relying Parties (RP) in these schemes, relying on IdPs to support identification, authentication and authorization decisions and to store accounting information. IdPs usually provide RPs different Software Development Kits (SDKs) and Application Programming Interfaces (APIs) to help their development teams in implementing access control functionalities.

Many standards and protocols have been specified in the last few years following this kind of scheme. OpenID (OIDF) is an authentication protocol providing a way to prove that an end user controls a specific identifier. OAuth (IETF) is an authorization protocol typically focused on managing access delegation. OpenID Connect (OIDC) (OIDF) is an authentication and authorization protocol based on building OpenID on top of OAuth, and therefore, extending it to solve authentication besides authorization. For example, if a user needs to check in for a flight, and the airline's website supports OpenID Connect, the user clicks on the Identity Provider logo as log-in option (Facebook or Google, for example) and she begins an

* Corresponding author.
E-mail addresses: jorge.navas@urjc.es (J. Navas), marta.beltran@urjc.es (M. Beltrán).
https://doi.org/10.1016/j.cose.2019.03.003

interaction with this provider to solve identification, authentication and authorization. As soon as this interaction is performed, the Identity Provider will verify her identity and privileges and she will be logged in to the airline website, with no need of having an account or of remembering a specific combination of user name and password at all.

The main contributions of this work are (1) the first, to our knowledge, in-depth threat model of OpenID Connect focused on its core specification (since discovery, dynamic registration and session management are optional) and taking into account security and privacy threats. (2) A complete description of each identified threat, summarizing its main impacts and consequences on security and privacy. (3) A systematic analysis of different attack patterns that an adversary may use to materialize this set of threats. (4) A set of mitigations, countermeasures and remediation options capable of avoiding or mitigating impacts of identified threats, acting on aspects of both, specification and implementation.

The rest of this paper is organized as follows. Section 2 summarizes the more important aspects of the OpenID Connect core specification and analyses previous security assessments of this standard (and/or of its foundations) as well as proposed improvements. Section 3 presents the performed threat model, including the considered context and assumptions and all the important information regarding the identified threats and analysed attack patterns. Section 4 proposes different alternatives to improve security and privacy levels of OpenID Connect specification and implementations and discusses their application in real world scenarios, their advantages and disadvantages, etc. Finally Section 5 summarizes the conclusions of this work and the most interesting lines for future research.

2. Background and motivation

2.1. *OpenID Connect core specification*

Fig. 1 provides a high level explanation of the typical OpenID Connect flow following its specification (OIDF). The end user participates in this flow, usually, trough a web browser acting as the User Agent and mediating between the IdP and the RP. The whole specification is based on HTTP. First the end user asks for access to the RP (step 1), which can be a web application, a mobile app or a native application. Then, the RP prepares (step 2) and sends an Authentication Request to the IdP (step 3). The IdP authenticates the end user or, if she has already been authenticated and has an active session in the IdP (this is often checked using session cookies although this aspect is not explicitly discussed in the OpenID Connect specification because it is considered beyond its scope), the IdP only has to ask for her consent (steps 4 to 6). If the user's authentication/consent is performed successfully, the RP will receive two different JSON (JavaScript Object Notation, IETF) tokens: the ID token, regarding end user's identity and the authentication process (see Table 1), and the Access token, regarding end user's privileges and the authorization process. If these tokens are successfully validated, the end user gets access to the requested resource. After this validation, the RP may (optionally) ask the IdP about user's attributes which can be useful for their interaction (for example, to auto-complete a form). These additional and optional steps of the flow take place without user explicit consent or awareness.

There are two different scenarios depending on how the ID token and the Access token are sent to the RP. In the Implicit Flow of OpenID Connect, the IdP sends these tokens to the end user and requests her to redirect these tokens, via User Agent (the browser), to the RP (steps 6 and 7). In the Authorization Code Flow, the end user only redirects to the RP an Authorization Code (steps 6 and 7 again), the RP will use this specific code to retrieve tokens directly from the IdP without any User Agent mediating this communication (steps 8 and 9).

These flows cannot be performed each time the end user initiates an HTTP request, again, session cookies are the most widespread solution: the RP stores a persistent cookie in the EU's browser, as long as this cookie is valid, the EU is still logged in the RP and there is no need to involve the IdP in a new IAAA flow. But again, this is an implementation issue beyond the scope of the OpenID Connect specification.

2.2. *Security analysis of federated identity management specifications*

There are some interesting previous works focused on presenting different kinds of security analysis of Single Sign-On and federated identity management specifications such as OAuth or OpenID Connect.

A first group of these works has analysed different implementations of these specifications, their vulnerabilities, possible attack patterns and common specification ambiguities that cause misunderstandings. Some good examples are Chen et al. (2014), Hu et al. (2014), Li and Mitchell (2014), Li and Mitchell (2016), Sun and Beznosov (2012) or Wang et al. (2012). Even some works, such as Zhou and Evans (2014) or Mainka et al. (2017) (this last tool is available in CNDS) propose tools to automate security testing of specific implementations of IdPs or RPs based on OAuth or OpenID Connect searching for known vulnerabilities.

The second group of researches is focused on performing formal security analysis and threat modelling, proposing different approaches to model authentication and authorization flows and to use these models to find new weaknesses in the specifications and in their implementations. Some good examples are Fett et al. (2016), Yang et al. (2016), Fett et al. (2017), Sciarretta et al. (2017) or Ghasemisharif et al. (2018). The IETF published a thorough threat model of OAuth 2.0 in 2013 too (IETF).

As a summary, Table 2 shows a comparison of previous works (the most significant for this research), where "Spec." is the specification focus of the research, "Sec. an." means that this work is focused on analysing or testing security threats, "Priv. an." means that this work is focused on analysing or testing privacy threats, "Tokens" means that this work proposes some kind of security/privacy improvement based on modifying tokens structure or treatment, "Crypto." means that this work proposes some kind of security/privacy improvement based on using cryptographic mechanisms (signatures, blockchain, etc.), "Flows" means that this work proposes some kind of security/privacy improvement based on modifying the steps followed by the IAAA flow described in the analysed

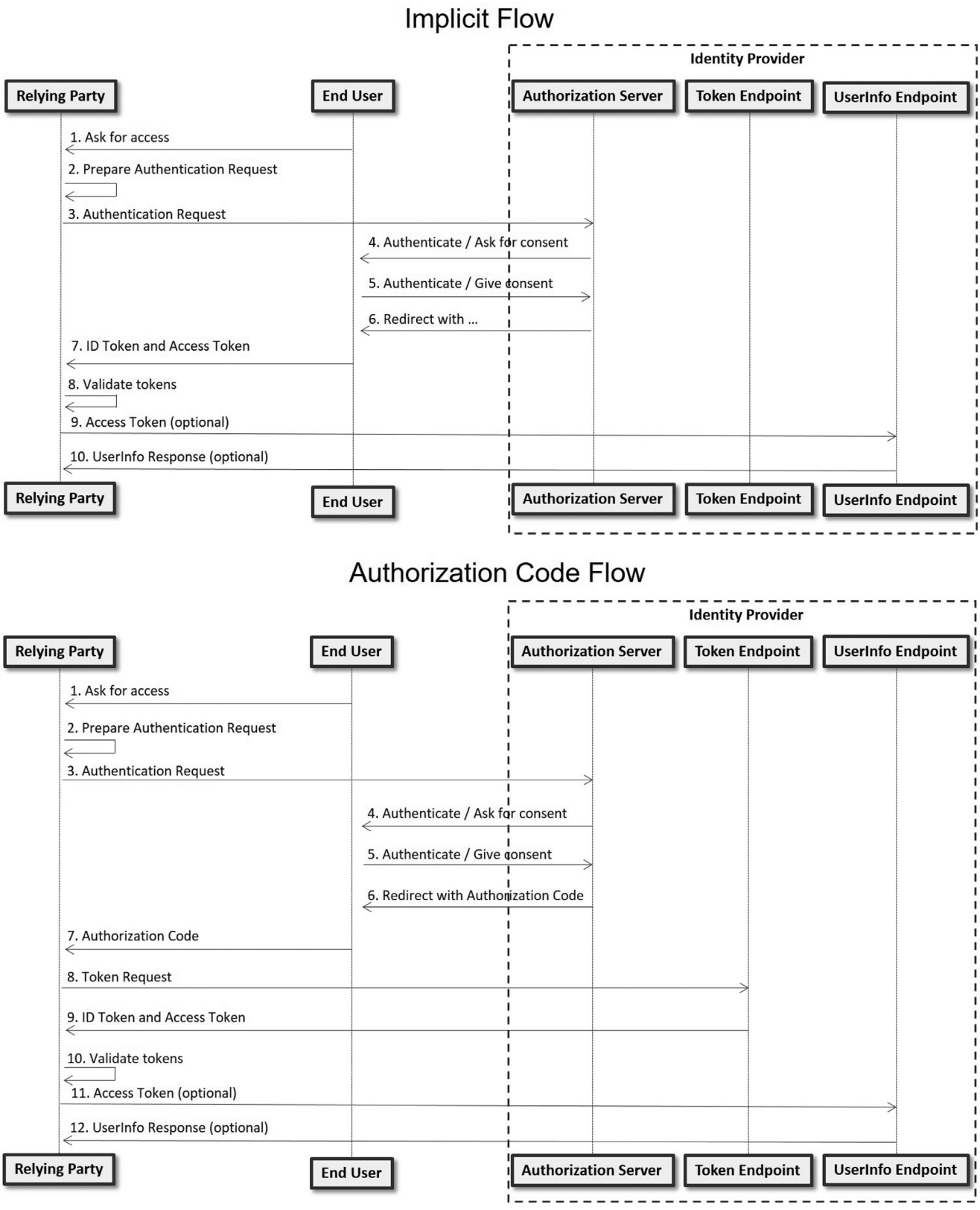

Fig. 1 – Implicit flow and authorization code flow in OpenID Connect.

Table 1 – ID token in OpenID Connect (mandatory parameters).

Param.	Explanation
iss	Issuer Identifier for the issuer of the token, the IdP – a case sensitive URL, using HTTPS-
sub	Subject Identifier, a unique and never reassigned identifier within the IdP for the end user – case sensitive and not exceeding 255 ASCII characters in length-
aud	Audience that this token is issued for, an array of case sensitive strings that must include, at least, the *client_id* of the RP that sent the Authentication Request in first place
exp	Expiration time for the ID token in seconds since the epoch, after this time the ID token must not be validated
iat	Time when the ID token has been issued in seconds since the epoch

specification, "Imp." means that this work proposes some kind of security/privacy improvement based on changing the traditional way of implementing the analysed specification (SDKs and APIs, tokens validation, use of cookies, etc.) and "Pol." means that this work proposes some kind of security/privacy improvement based on the definition of policies, agreements, reputation systems and other similar techniques.

2.3. *Motivation*

Threat modelling is the process by which potential threats are identified, enumerated and understood in order to provide defenders with a thorough analysis of attackers' motivation and profile, the most likely attack vectors and patterns, etc. It is considered one of the best methods for improving the security of an infrastructure, application or project by helping the security team to accurately determine the attack surface and to drive the mitigation process. A threat model should answer questions like What are the most relevant threats?, "How these threats can be materialized", "What attack vectors can be used?", "Which attack patterns could be executed to exploit my vulnerabilities?" or "What kind of vulnerabilities might be exploited?".

To perform a threat modelling process a security team needs to understand the security requirements and scenarios, to identify external dependencies and to define security assumptions. Different works quantify in around 60.000 the websites and apps currently supporting different versions of OpenID Connect (Ghasemisharif et al., 2018 is the most recent, the number is constantly growing), being Facebook, Google and Twitter the most used identity providers. How are supposed all these websites and apps to understand the security impacts of relying on this complex specification and on depending on one of its specific implementations if there is no threat model for this external component? Recent attacks with significant impacts such as Rosen shows the importance of understanding and mitigating OpenID Connect threats, because they may have significant effects on a large number of websites and apps.

To the best of our knowledge there are not previous publications devoted to produce a complete threat model of OpenID Connect, taking into account both, security and privacy aspects and considering both, specification and implementation issues. This model is essential given the current state of federated identity management in the Internet and should be the basis of a motivated proposal of different kinds of improvements in the specification and in its implementations. Next section of this paper present our main results in this area.

3. OpenID Connect threat model

3.1. *Assumptions and methodology*

This section presents our comprehensive threat model of OpenID Connect core specification when used in the traditional scenario (RP, IdP and end user). This model is general, i.e. relying parties are not bound to a specific implementation or product deploying OpenID Connect, our goal is to analyse security and privacy aspects of the specification and of the common way of implementing it, not of specific versions. Our model capture threats coming from different adversaries with

Table 2 – Comparison of previous works.

Ref.	Spec.	Sec. an.	Priv. an.	Tokens	Crypto.	Flows	Imp.	Pol.
Hu et al. (2014)	OAuth	*						
Yang and Manoharan (2013)	OAuth	*						
Bansal et al. (2014)	OAuth	*						
Chari et al. (2011)	OAuth	*						
Fett et al. (2016)	OAuth	*						
Yang et al. (2016)	OAuth	*		*			*	
Birrell and Schneider (2013)	Many		*					
Mainka et al. (2017)	OIDC	*		*			*	
Mainka and Schwenk (2016)	OIDC	*						
Werner and Westphall (2016)	OIDC				*			*
Fett et al. (2017)	OIDC	*		*		*	*	
Halpin (2017)	OIDC				*	*		
Weingärtner and Westphall (2017)	OIDC		*		*	*		
Our work	OIDC	*	*	*	*	*	*	*

different goals, strategies, levels of access, capabilities, etc. To perform this analysis some assumptions have been considered:

- Adversary assumptions:
  - (A1) Adversaries attacking OpenID Connect are focused on spoofing the end user, therefore, on assuming her attributes and using her identity with different motivations. Or on achieving some kind of information disclosure, getting access to private details about the end user, her habits, etc. If following the STRIDE model (OWASP), our model is focused on identifying threats regarding the Spoofing Identity category (the 'S' in the STRIDE acronym) and the Information Disclosure category (the 'I').
  - (A2) Adversaries have unlimited resources to make effective their threats.
  - (A3) Adversaries may have different levels of access to different components of OpenID Connect flows: to the IdP or RP infrastructure, to the end user's smart phone, to the communication channels, etc. All these access levels will be considered during our analysis.
- Architectural assumptions:
  - (A4) A reliable confidential communication channel between the IdP and the RP and between the end user and the IdP is available when required by the IdP or the RP.
  - (A5) The following data elements are stored in or can be retrieved from the IdP infrastructure: user names, phone numbers and identifiers/credentials, RPs identifiers (*client_id*) and secrets, HTTPS certificates and keys (to enable confidential communication channels, since the specification is focused on HTTP).
  - (A6) The following data elements are stored in or can retrieved from the RP infrastructure: RP identifiers and secrets, HTTPS certificates (and client secret or corresponding client credential) and transiently, authorization codes, ID and Access tokens. Optionally, user names, phone numbers and identifiers.
- Trust assumptions:
  - (A7) Legitimate IdPs and RPs can be trusted regarding security, that is, IdPs issue only valid and correct tokens and IdPs and RPs are not under control of adversaries. Also, RPs register only redirection URIs that point to themselves and receive only correct signing keys from legitimate IdPs.
  - (A8) Legitimate IdPs and RPs do not collude to threaten end users security or privacy.
  - (A9) Legitimate OpenID Connect APIs can be trusted by relying parties and developers, that is, they are properly built and fortified.
- Identity assumptions:
  - (A10) An end user identity is considered Personally Identifiable Information (PII) since it is composed of a set of attributes (identifiers) that can be used to identify the user such as name, surname, email address, phone number, credit card number, etc.

Our intention with these assumptions is to identify security and privacy threats that can be attributed to the specific use of OpenID Connect, ignoring other well-known and generic threats caused by weak implementations of SSL, denial of service at the IdP, etc. Threats coming from Discovery and Registration specifications, both optional in OpenID Connect, have not been considered either.

Regarding our methodology, all performed experiments and tests, necessary to propose and to validate the different attack patterns that an adversary may use to materialize identified threats, have been performed within a controlled lab due to ethical and legal reasons. End users use different devices relying on different versions of different browsers (three of the most popular: Google Chrome, Mozilla Firefox and Microsoft Edge) with different security and privacy configurations. End users can send requests to to any web page (HTTP and HTTPS), navigate backwards or forward in their browsers, open different windows, start simultaneous IAAA flows using different or the same IdPs, rely on different configurations of cookies, etc. Relying parties have been implemented replicating real web sites and applications of different sectors (social networks, news and media, e-commerce, etc.) with different security and privacy requirements, programmed with different languages and relying on different Certified Relying Party Libraries (OIDF). More concretely we have not developed and deployed complete web sites and applications, only the part of web sites and applications connecting to the End User and to the IdP to execute the IAAA flow with OpenID Connect (since it is the component whose security and privacy levels are going to be assessed). This simplified our work and up to 50 web sites (out of the Alexa top 500) and 25 Android apps (within the 100 Google Play most popular applications) have been developed and deployed. All tested webs and applications rely on Facebook and/or Google as IdP. In order to prevent our experiments to interfere with the operation of these IdPs, we have replaced Facebook and Google APIs' calls in the tested web sites and apps with calls to other providers' APIs.

Specifically, two different identity providers have been deployed with two different OpenID Connect frameworks, both open source, certified OpenID Provider Servers and Services (OIDF), widely used in production environments and following the OpenID Connect current specification as the Facebook and Google implementations, to a high degree but not perfectly (Facebook and Google): ForgeRock Identity Platform (ForgeRock) and WSO2 Identity Server (WSO2).

3.2. *Security threats: spoofing identity*

In the proposed model, threats have been grouped by adversary's goal (threats to security or threats to privacy, consider A1), and subsequently, by adversary's strategy (consider A2) and by adversary's level of access (consider A3). Regarding Security Threats, this level of access may be Physical Access (the adversary has physical access to the victim's device/browser), Network Access (the adversary has access to the communication channel between the RP and the IdP and/or between the end user and the RP, being able to eavesdrop their communications) or Web Access (rest of the cases).

3.2.1. *Replay of legitimate tokens*

This threat is caused by an improper protection of honest ID and Access tokens and/or by an incomplete validation of ID tokens (and their freshness and/or audience) at the target RP. These issues during the validation allow an adversary to access resources in the RP presenting an honest ID token (issued for this RP by the legitimate IdP), maybe already expired. Or, even worse, to access resources in different RPs presenting an ID token issued for other RP by the legitimate IdP, maybe already expired. This threat is present with both Implicit and Authorization Code Flows.

Attack pattern 1 (requiring physical access): stolen token. In order to run this attack, the adversary needs the end user to initiate an authentication flow with the Implicit Flow. Once the EU has been authenticated by the IdP, ID and Access tokens are sent directly through her web browser to the RP. If the adversary gains access to the EU's device, these tokens can be stolen (from a cache, a cookie, a log file or browsing history, for example) and the adversary may try to replay both of them in the legitimate RP (because tokens have not expired yet or because this expiration is not properly checked). Or to replay both of them at a different RP not properly validating the audience of the tokens.

Attack pattern 2 (requiring network access): sniffed token. In order to run this attack, the adversary needs to sniff some of the involved communication channels to obtain ID and Access tokens. After this, she can re-use legitimate tokens at the legitimate RP, unable to detect the replay (because tokens have not expired yet or because this expiration is not properly checked) or at other RPs not validating the tokens' audience. This attack is only possible when the adversary is able to sniff a communication channel not properly protected by HTTPS.

Attack pattern 3 (only web access required): compromised device. One more sophisticated way to obtain legitimate tokens could be infecting the victim's device with some kind of malware or making this victim to install malicious browser extensions or apps. These pieces of evil software are able to retrieve ID and Access tokens directly from caches, log files or browsing histories in the device.

Attack pattern 4 (only web access required): phished token. In order to run this attack, the adversary looks for tokens in the Internet (these tokens may be improperly protected at RP's and could be found using a web crawler, for example) or Dark Web forums. Remember that if certain parameters of the ID token such as *exp* or *iat* are not properly validated at the legitimate RP, the adversary may re-use these tokens to gain access to different resources in this RP impersonating the end user thanks to her tokens. The adversary can also try to present phished tokens in other RPs not properly validating the *aud* parameter of the ID token.

Attack pattern 5 (only web access required): honey-RP. If the adversary is able to run a malicious RP relying on the same IdP that the legitimate RP and the end user, she can trick the victim (the legitimate end user) to log in this malicious RP (using social engineering, for example). In this way, the adversary obtains ID and Access tokens issued by the legitimate IdP. These tokens can be replayed at different RP's where the *aud* parameter of ID tokens is not properly validated.

3.2.2. *Replay of Authorization Codes*

During an Authorization Code Flow, the Authorization Code is used to obtain ID and Access tokens at the IdP. This threat is posed by an adversary obtaining these tokens by some mean captured Authorization Code. This threat is present with Authorization Code Flows due to improper Token Request validation at the IdP and/or to legitimate RP spoofing at some level.

Attack pattern 6 (requiring physical access): stolen code. In order to run this attack, the adversary needs to access end user's device, specifically, its browser. If the adversary is able to access to cache contents or to certain logs or browsing histories, she may retrieve the required Authorization Code. From this point the adversary uses this code to try to obtain ID and Access tokens at the IdP. The IdP should perform a thorough Token Request validation to authenticate the RP (the Authorization Code was issued to this specific RP), to verify that the Authorization Code has not been previously used, to ensure that the RP is the one who sent the initial Authentication Request, etc. If any of these tasks is not performed correctly, this attack pattern may succeed.

Attack pattern 7 (requiring network access): sniffed code. In order to run this attack, the adversary needs to sniff the traffic between the RP and the end user. If the traffic is not properly encrypted during the OpenID Connect flow, the adversary may capture the required Authorization Code. And from this point, again, the adversary will use this code to try to obtain ID and Access tokens at the IdP.

Attack pattern 8 (only web access required): leaked code through headers. The Authorization Code is part of the redirection URI at the RP. If during the interaction between the end user and the RP the user clicks on a link to an external page or if the RP relies on third party external resources (scripts, advertising, etc.), the third party may receive the full URI in the Referrer header, including the Authorization Code.

Attack pattern 9 (only web access required): compromised device. Again, infecting the victim's device with some kind of malware or making this victim to install malicious browser extensions or apps allows the adversary to retrieve authorization codes from the mobile phone (caches, logs, histories, etc.).

3.2.3. *Manufacture of fake tokens*

This threat is caused by an improper or incomplete validation of ID tokens at the target RP. These issues during the validation allow an adversary to access resources in the RP presenting fake ID tokens customized to achieve her goals. Fake tokens can be obtained by modifying certain parameters of a legitimate token issued by a legitimate IdP or they can be manufactured from scratch, for example, by a malicious IdP.

This threat can be made effective with both Implicit and Authorization Code Flows.

Attack pattern 10 (possible with physical access, network access and/or web access): crafted token. The basis for this attack may be any of the attack 2–6 described before. Once the adversary has legitimate tokens, instead of replaying them, some parameters are modified before in order to guarantee a successful validation at the target RP (for example, the *aud* parameter to set the correct token recipient or the *exp* parameter to avoid a rejection due to the expiration of the token). These modifications may make the RP reject the token because the signature verification alerts about problems with token's integrity. To avoid these rejection, the adversary has different alternatives. The first, to perform signature bypassing, i.e., to change token's content without invalidating the signature. The second, enforcing the RP to use wrong keys when verifying signatures. And the third, removing the signature relying on a policy at the RP trying to avoid denial-of-service to end users and accepting tokens without signatures at all (more usual than one could think).

Attack pattern 11 (only web access required): malicious IdP. In this case the adversary sets up a malicious IdP in order to issue a new and fresh ID token with the *iss* and *sub* parameters corresponding to the legitimate values (the legitimate IdP and the legitimate end user). If the RP does not check properly the rest of parameters included in the token (signatures, for example), it will be validated and the spoofing threat will succeed.

3.2.4. *Flow interception*

This threat is caused by the capacity of the adversary of intercepting some of the IAAA flows impersonating the legitimate RP (manipulating the redirection URI, for example) or the legitimate IdP (manipulating the *client_id* values, for example). This threat is different from previously described threats based on replaying legitimate tokens or on replaying Authorization Codes because in this case, the token or codes are used by the adversary for the first time, they are not used by the legitimate end user even once. This threat can be made effective with both Implicit and Authorization Code Flows.

Attack pattern 12 (requiring network access): MitM in order to manipulate redirection URI. Redirection URI is the HTTP or HTTPS direction which the response of the IdP (codes, tokens, etc.) will be sent though end user's browser (consider A7). This URI must be pre-registered at the IdP and must be properly validated before any interaction. If this parameter is modified at some point of a flow, and it is not properly handled by the IdP, information such as Authorization Codes or ID tokens can be retrieved by the adversary using a malicious RP. In order to run this attack, the adversary needs the end user to begin a IAAA flow. As the adversary has access to the RP's network, she can obtain and modify the Authentication Request, by manipulating the redirection URI parameter. If the IdP does not validate this parameter properly, the legitimate ID token or Authorization Code will be sent (redirected for the end user's browser to be precise) to the adversary's RP.

Attack pattern 13 (only web access required): remote manipulation of redirection URI. In order to run this attack, the adversary needs the end user to begin a IAAA flow following a malicious link, issuing a a manipulated Authentication Request containing a redirect URI pointing to a RP controlled by the attacker. Again, if the IdP does not validate the redirection URI properly, the legitimate ID token or Authorization Code will be redirected to the adversary.

Attack pattern 14 (only web access required): IdP hot-swapping. In this case the adversary needs to control a malicious IdP and the end user to begin a IAAA flow following a malicious link. If the target RP has the same *client_id* at the legitimate IdP and at the malicious IdP, during an Authorization Code Flow the adversary is able to intercept the legitimate Authorization Code, the RP is not able to determine with which IdP is interacting at different moments of the OpenID Connect flow. This is, perhaps, the most sophisticated attack pattern tested during the threat modelling process since it involves manipulating different messages and communications during the IAAA flow taking advantage of the lack of a binding between them. The focus of this paper is not providing a detailed explanation of all possible alternatives to build proposed attack patterns step by step, but, since this attack is not immediate to understand or to reproduce, more details can be found in Mainka et al. (2017), where it has been tested within a specific scenario and called "IdP Confusion".

3.2.5. *IdP account compromise*

This threat can be made effective with both Implicit and Authorization Code Flows and it is caused by the nature of the IdP as a single point of failure: if the IdP account is compromised by an adversary, this adversary is able to impersonate the victim in all RPs supporting this IdP as Identity Provider.

Attack pattern 15 (possible with physical access, network access and/or web access): compromised IdP password. The adversary may rely on different techniques to obtain the victim's password in the IdP. For example, when she has access to the victim's device it can be recovered from a cache, a cookie, a log file, etc. This password can also be obtained by the adversary installing a malicious software on the victim's device, a key logger, for example. The adversary could capture the victim's password sniffing the network, specially easy when the victim is navigating through an open wireless network, something very common nowadays. Last, social engineering techniques must be mentioned, because with only Web Access an adversary is able to obtain the victim's password using phishing techniques (even brute force could be successful in a reasonable time is useful information about the victim is available – because almost all IdPs limit the number of attempts when entering the password), perhaps the most powerful attack pattern of all the included in this category. Once the adversary knows the victim's password in the IdP, she is able to interact with any RP supporting this IdP as the victim: when the IdP asks the adversary to authenticate/consent (steps 4–6 in Fig. 1), the adversary will spoof the victim's identity using her password and will continue the IAAA flow impersonating her. It has to be noted that this is an implementation issue, because almost all IdPs currently rely on a user name and password combination to perform end user's authentication. The specification of OpenID Connect explicitly says "the methods used by the Authorization Server to Authenticate the End-User

(e.g. username and password, session cookies, etc.) are beyond the scope of this specification".

Attack pattern 16 (possible with physical access, network access and/or web access required): session hijacking. In this case the adversary is not able to capture the victim's password but she is able to hijack her session at the IdP, i.e. she is able to obtain the session cookie of the IdP (directly from the browser, sniffing it when being transmitted over an insecure network, using malicious software, relying on some kind of XSS attack, etc.). If the RP does not force the IdP to authenticate the end user at the beginning of each session and it simply allows the IdP to check if the end user is already logged in (and this checking is performed through the session cookie), the adversary is able to impersonate the victim by simply loading the obtained session cookie in her browser. This attack is successful as long as the IdP and/or the RP does not ask the end user to perform a new authentication introducing her password again.

3.3. *Privacy threats: information disclosure*

In this case threats have been grouped again by adversary's strategy and by adversary's level of access: Privileged Access (to the RP or IdP infrastructures, i.e. within these organizations or acting on their behalf) or Non-Privileged Access (rest of the cases). It has to be considered that our analysis does not assume that legitimate IdPs and relying parties can be trusted by end users regarding privacy (there is not an equivalent assumption for A7 regarding privacy aspects). This is because, somehow, business models of IdPs usually rely on data about end users' habits, because IdPs and RPs can be lazy about privacy or just curious, etc.

3.3.1. *Lack of control over required PII*

This threat is caused by the loss of control of end users over their PII (consider assumption A10). Once the Identity Provider and/or the Relying Party has asked for different attributes, end users do not have means to manage how these attributes are used, stored, etc.

This threat can be made effective only by agents involved in the IAAA flows (Privileged Access), therefore, by the Identity Provider or by the Relying Party. The first gathers users' attributes during the initial Registration phase of OpenID Connect (enrolment or on-boarding procedure at the Identity Provider to enable the service). The second, optionally, after the IAAA flow, presenting the Access token at the IdP to retrieve these attributes (for example, to auto-complete a form with usual data: name, surname, address, etc.). Depending on the IdP and on the RP, different attributes may be required to perform the Registration process or to complete the access to the requested resource in the RP. Very often, these agents have unlimited access to these attributes while this is not the case for end users.

3.3.2. *Lack of transparency in the sharing of PII*

This threat is caused by privileged agents (IdP and RP) of the IAAA flows when they share PII (A10 again) with third parties without end users awareness.

Again this threat can be made effective only by agents involved in the IAAA flows (Privileged Access), the IdP or the RP, both with access to PII and very often, not providing end users with the means to know how this PII is being disseminated or shared. This lack of transparency may have different causes: because of opaque business models at the IdP or RP, because a fine-grain control over PII dissemination can be a great burden for different involved agents, etc.

3.3.3. *PII leakage*

This threat is caused by breaches at some point of the IAAA flows, resulting on the unintended revelation of sensitive attributes to third parties. This threat cabe made effective with both, Privileged and Non-Privileged Access.

Although IdPs and RPs are trustworthy agents (consider assumption A7, if there is not trust, a federation would be impossible to build), adversaries may have different levels of access to their infrastructures (consider assumption A3). Even trusting an organization, one cannot trust all its employees or stakeholders, for example. Since assumptions A5 and A6 makes possible to retrieve PII from these infrastructures, if sensitive information is not properly encrypted significant leakages can be observed.

If sensitive information is transmitted without proper encryption over non-secure communication channels, an adversary with network access could use a sniffing tool to retrieve this information (name, surname, email, phone number, credit card details, etc.). In the case of adversaries with only Web Access, PII leaked from an IdP or RP could be retrieved from or bought in the Internet or the Dark Web.

3.3.4. *User profiling*

This threat is caused by the ability of different agents to track end users activity and correlate this activity across different applications, services or resources, trying to learn specific aspects regarding user's behaviour, habits, interests, devices, etc.

This threat, requiring Privileged Access, can be made effective by agents involved in the IAAA flows, given that the subject identifier included in the ID token unambiguously corresponds to an end user but from outside the IdP and RP infrastructure the correspondence of this identifier with the real person is not known. Static user profiling involves analysing user available data and static attributes. Dynamic profiling, on the other hand, involves the integration of available data with other sources of information (from social networks, for example), the consideration of data from other similar users, the application of prediction techniques, etc.

3.3.5. *Location tracking*

This threat is caused by the ability of different agents within the IAAA flows to track end users locations over time through their laptops and mobile devices, mainly using information from mobile phone towers, GPS satellites, location logs from different apps, Wi-Fi history, IP addresses, etc.

This threat can be made effective only by privileged agents involved in the IAAA flows, RPs and mainly, IdPs; since the information regarding location is not transmitted during these flows in any way. Location information, obtained by different means by providers, can reveal not just where an end user lives, works or travels, but also her routines, interests, relationships with others, etc.

Table 3 – OpenID Connect threat model (summary).

Threat	Goal	Attacks	Level of access	Cause
Replay of legitimate tokens	S	Stolen token, sniffed token, compromised device, phished token and honey-RP	Physical, network and web	Imp.
Replay of Authorization Codes	S	Stolen code, sniffed code, leaked code through headers and compromised device	Physical, network and web	Imp.
Manufacture of fake tokens	S	Crafted token and malicious IdP	Physical, network and web	Imp.
Flow interception	S	MitM in order to manipulate redirection URI, remote manipulation of redirection URI and IdP hot-swapping	Network and web	Spec.
IdP account compromise	S	Compromised IdP password and session hijacking	Physical, network and web	Imp.
Lack of control over required PII	I	Loss of control of end users over their PII	Privileged	Spec.
Lack of transparency in the sharing of PII	I	PII sharing without end users awareness	Privileged	Spec.
PII leakage	I	Unintended revelation of sensitive attributes to third parties	Privileged and non-privileged	Spec. and Imp.
User profiling	I	Track end users activity and establish correlations	Privileged	Spec.
Location tracking	I	IdP tracks end users locations over time through their devices	Privileged	Spec.

4. Proposed mitigations to identified threats

4.1. *Preliminaries*

Table 3 summarizes the proposed threat model. For each threat this table shows its goal (spoofing, S, or information disclosure, I), the set of described attacks, the required level of access and if they can be attributed to flaws of the OpenID Connect specification or to common implementation mistakes (at the IdP and/or RP). We do not consider this model finished, it will probably be extended with new attack patterns although we hope that threat categories (attending to the adversary's goal, strategy and level of access) will remain permanent.

As main conclusions, it is essential that end users rely on device locking solutions, and, when their devices are stolen or lost, on revoking tokens and blocking their OpenID Connect account at the IdP. It is also essential that developers at the RP and the IdP strictly and thoroughly follows the OpenID Connect specification, paying special attention to validation procedures (requests validation, tokens validation, signatures validation), to potential specification ambiguities and to the handling of exceptional situations (a token without signature, etc.).

Although it is not caused by the specification itself, the Implicit Flow can be considered more insecure than the Authorization Code Flow, mainly due to the proliferation of mobile malware, third party malicious scripts, browser extensions and plugins or apps, etc. (all of these techniques can be used to steal legitimate tokens, to modify them, to manipulate the redirection URI). Furthermore, a correct and intensive use of TLS in all the IAAA flows is required to mitigate and/or to avoid many of the identified threats.

A specific situation must be remarked regarding amplification effects that can be observed when a RP is at the same time an identity provider (think about resources, applications and services offered by Facebook or Google where you can login using different IdPs). If an end user's account is compromised in the RP role, this can lead to a more critical and extensive compromise when this account is binded to the IdP role. To prevent this kind of amplification, potentially compromising all end user accounts that can be accessed from a dual-role IdP, this kind of providers must be specially careful, guaranteeing a secure access for resources, applications and services on its own platform, implementing OpenID Connect correctly for all of IdPs it supports and properly separating users' accounts in the RP and in the IdP role.

Following sections propose a set of specific mitigations, countermeasures and remediation options capable of avoiding or mitigating impacts of identified threats, acting on aspects of both, specification and implementation.

4.2. *Security-focused mitigations*

4.2.1. *Specification: Level of Assurance (risk-based authentication)*

Although the end user authentication at the IdP is out of the scope of current OpenID Connect specification, our threat model shows that a bad approach for this authentication implies significant risks. Therefore we propose a modification of the specification in its future versions to provide some guides regarding this aspect.

The Mobile Connect specification (GSMA) (based on OpenID Connect, relying on Mobile Network Operators as identity providers) introduces the concept or Level of Assurance within the authentication flows. In this paper a similar approach is proposed, future implementations of OpenID Connect should seriously consider this improvement.

Using this mitigation, the RP has to specify, when preparing the Authentication Request (step 2 in Fig. 1), the degree

of confidence that is required when the end user is authenticated by the IdP. This specification can be performed through the *acr_values* parameter of the request, optional in the current version of the specification: it should be mandatory. The greater the risk associated with an erroneous authentication, the higher the Level of Assurance recommended. Following the ISO/IEC 29115 standard (ISO/IEC) there are four possible levels of assurance. OpenID Connect should not allow RPs to work with Level 1 because it is too risky (it provides little or no confidence in the asserted identity's validity) and it would not make sense in the typical OpenID Connect scenarios. It would be the case of a long-live browser cookie, for example. Level 2 is based on "Something you know" authentication: if the end user proves to know a password, for example, the authentication is successful. This is the default Level of Assurance used now in all OpenID Connect implementations (step 5 in Fig. 1) and it provides some confidence in the asserted identity's validity.

We propose here to add the possibility of performing authentication with Level 3 and Level 4 of assurance, i.e. high confidence and very high confidence in the asserted identity's validity respectively. In order to guarantee a higher degree of confidence, Level 3 forces the end user to use a second authentication factor and to perform some kind of cryptographic information exchange with the Identity Provider. Therefore, step 5 in Fig. 1 needs to add a "Something you have" based authentication or "Something you are o you do" based authentication including a cryptographic mechanism. Different scenarios should be considered because different kind of end users with different kinds of devices and available hardware may be relying on OpenID Connect.

Second factor authentication based on SMS (considering that the "Something you have" factor is the end user's mobile phone) or on specific authentication apps (such as Google Authenticator or LastPass Authenticator, to mention only two well known examples) should be avoided due to their vulnerabilities (Dmitrienko and Sadeghi, 2014; Polleit and Spreitzenbarth, 2018) and to the lack of strong underlying cryptographic mechanisms. We suggest relying on universal second factor keys (U2F) following the FIDO Alliance standard Alliance or on Trusted Platform Modules (TPM). Both alternatives can be integrated within OpenID Connect (during step 5 in Fig. 1, the Authorization Server should be improved) and the required specific hardware may even been securely emulated if the end user has not this hardware available when needed (Reimair and Marsalek, 2016) in order to avoid this hardware to become an adoption barrier.

The assurance with Level 4 should only be required when accessing to critical resources, services and applications because it implies a complex on-boarding within the identity provider requiring in-person identity proofing. This could be the case when this Identity Provider is a government, public administration or a bank, all of them entities with physical facilities where this proofing could be performed by authorized agents. From a technical point of view, the authentication requires the same steps that as in Level 3, being a very interesting alternative the use of a mobile phone or touch device to add a "Something you are or you do" biometric authenticator (Al-Ghamdi et al., 2017; Teh et al., 2016).

4.2.2. *Specification: re-authentication at the IdP always required*

Current specification allows the IdP to perform the IAAA flow without re-authenticating the end user if she has already logged in at the IdP. We propose to modify the specification in order to force the user to re-authenticate for each new session at the RP. At least, an Authentication Request with LoA = 2 should be achieved at the beginning of a new session with the RP, always.

4.2.3. *Specification: enriched requests and tokens*

The OpenID Connect specification proposes some optional parameters for the Authentication Request (specifically *state, nonce* and *acr_values*) and for the ID token (specifically *nonce, acr* and *amr*), usually not included in real world implementations but essential for security in certain cases.

In this work we propose a modification of the specification in order to make *nonce, acr* and *amr* mandatory parameters within the ID token; correspondingly, *state, nonce* and *acr_values* should be mandatory within the Authentication Request.

Some of these parameters are essential to properly implement the first proposed improvement regarding the utilization of the Level of Assurance concept. As it has been mentioned before, the RP specifies the required LoA using the *acr_values* parameter in the Authentication Request. The *acr* parameter in the ID token is a string indicating the achieved LoA while the *amr* parameter is an array of strings identifying the authentication methods used to achieve this specific Level of Assurance. The IdPs and the RPs should agree during the registration process about the meaning of these values. For example, "PWD" for password, "OTP" for one-time password using a SMS, "FPT" for a fingerprint, etc.

Regarding the rest of aforementioned parameters, all of them enable a more secure management of sessions and cookies. For example, the *state* parameter of the Authentication Request is used to maintain a state between the request and the callback at the RP, it can be cryptographically binded with a browser cookie at the User Agent to securely manage the current session. The *nonce* parameter in the Authentication Request and the token is again a string value, it must be unguessable to attackers. In this case, it is used to associate a session of a specific end user with it specific ID token. This value is present, unmodified, from the beginning of the IAAA flow, passing, if everything is correct, from the Authentication Request to the ID Token.

OpenID Connect specification enables other Claims within the ID tokens. Given the threats identified in the previous section of this paper, including identifiers in flows' messages and checking these values, may be a good idea because this information can help in preventing an adversary to impersonate an agent of this flow (mainly, the EU or the RP). As it has been demonstrated, the use of bearer tokens implies that any party in possession of a token is able to gain access to the protected resource, application or service. We propose two different options to mitigate the related threats:

1. Add to these Claims two new parameters *sub_IP* and *aud_IP*, indicating the IP address of the end user and of the Relying Party respectively. This measure should be

used when relying on http to connect the end user and the RP.
2. Add to these Claims a new parameter, *public_key*, the public key of the TLS channel established between the end user and the RP. Therefore this improvement, binding the token to a specific TLS channel (the RP checks this parameter against the corresponding private key), can only be applied when using https to connect the end user and the RP.

4.2.4. *Specification: standardized session management*

The correct utilization of *state* and *nonce* parameters is tightly related to session management mechanisms, currently out of the scope of the OpenID Connect specification. This is the reason why application developers at RPs implement multiple options to perform session management and current SDKs and APIs provided by IdPs are not able to cover all these session management tools and mechanisms. As a result, developers cannot rely on official SDKs and APIs to perform certain verifications and validations. They usually skip these steps or try to develop custom routines and functions to perform them (with all the security issues this brings).

We suggest to introduce in the OpenID core specification some guidelines about session management based on cookies. This standardization will increase current levels of security in two ways. The first, providing unambiguous rules to manage RP sessions through cookies. The second, allowing IdPs to improve their SDKs and APIs knowing how developers are going to manage sessions and providing specific routines and functions compatible with this reduced set of management mechanisms.

For example, cookies must be always set with the secure attribute or flag, ensuring that the cookie value is only transmitted over HTTPS. Otherwise, an adversary with network access could read cookie values by eavesdropping on non-HTTPS connections to RPs. Other example, what happens when an ID token expires? The cookie in the User Agent should be invalidated until a new token is received, i.e. the expiration of tokens and cookies should be rigorously binded. Furthermore, including a universal revocation mechanism in the core specification of OpenID Connect (the single sign-off concept proposed in Ghasemisharif et al., 2018, for example) may be also very interesting when trying to mitigate the IdP account compromise threat.

4.2.5. *Specification: fragmented keys*

ID tokens should be signed with a multi-key or fragmented key instead of using the standard JSON Web Signature (JWS IETF). One part of the key is saved by the IdP, the other part is exclusively stored within the token itself (as a claim, for example). With this improvement, a compromise at the IdP would not compromise active tokens and sessions.

4.2.6. *Implementation: improved SDKs, APIs and official resources provided by IdPs*

Official and public SDKs provided by different IdPs do not include functions or routines for application developers at the RPs to properly perform some validations, verifications or exceptions management. We propose a thorough review of SDKs and APIs in order to provide all missing routines and functions to achieve compliance with the specification at any time. These routines and functions regarding token validation and verification are specially critical knowing that at many RPs only a very basic validation is performed (verifying one or two of the required parameters, allowing the use of parameters not included in the specification, wrongly performing cryptographic validations trying to save resources, etc.). This allows adversaries to successfully rely on some of the attack patterns identified in this work to materialize their threats.

A correct and complete token validation and verification must include all these steps:

- Token parsing: The received tokens have to be parsed into data objects so that they can be processed further. All required parameters must be present, if any of them is missing an error message must be produced. Any parameters that are not understood must be ignored following the specification.
- Origin verification: The RP receiving a token must validate the unique identity of the IdP (*iss* parameter) and check that the corresponding shared secrets, keys, certificates, etc. are available and updated to perform further cryptographic verifications.
- Audience verification: A token is intended for a single RP, it should be checked that the *aud* parameter has the correct value (the *client_id* at the IdP).
- Freshness validation: Parameters such as *exp* and *iat* enables this validation, essential to avoid replay threats detecting expired tokens.
- Cryptographic validation: This is usually the more time and resource consuming task, involving the verification of signatures, etc. It is essential to use the cryptographic material (key, cypher, etc.) belonging to the legitimate IdP.
- Session validation: The RP receiving a token must validate the *state* and *nonce* parameters.
- Additional claims verification: For example, if using the proposed security improvements, the RP must check that the IdP has authenticated (or re-authenticated) the end user with the required level and authentication methods. IP addresses involved in the IAAA flow or public keys of TLS connections should be checked too.

If any of these steps, even if only one, is missing or wrongly executed an adversary may be able to take advantage of it. In addition, it has to be pointed that timing attacks may leak useful information to potential adversaries, because there are significant differences in the code paths taken by successful and unsuccessful validation processes (specially regarding the cryptographic validation IETF). We suggest to avoid terminating these processes and sending an error message as soon as an error is found. All responses, successful or unsuccessful should take similar amounts of time.

Furthermore, templates and/or toy implementations of web applications, mobile apps and native application are very useful for developers and allow them to avoid many implementation mistakes. IdPs should work on providing this kind of support to their clients.

4.3. *Privacy-focused mitigations*

4.3.1. *Specification: encryption of PII*

PII must be hashed and encrypted by end users before sharing it with identity providers. This measure provides control over the data stored at IdPs, protecting confidentiality and integrity. Therefore, when an end user is registering with an IdP she has to hash and to encrypt with a private key her sensitive PII (email, telephone number, picture, address, credit card numbers, social security numbers, etc.), not essential for IdPs operation. This procedure must be performed by some kind of agent running on end users' side (a browser plugin, an app: some component at the User Agent). This plugin or app can rely on cryptographic software primitives or on some kind of trusted hardware (following the aforementioned FIDO Alliance standards or relying on Trusted Platform Modules) to guarantee isolation from other applications running on the same device and better protect the private key. These aspects may be out of the scope of the new specification, the essential recommendation here is to use cryptographic mechanisms regardless of how they are implemented.

Furthermore, when the step 5 of Fig. 1 takes place, the end user not only performs her authentication or gives her consent, she also decides, explicitly, which attributes of her PII can be revealed to this specific RP. To avoid this kind of decisions to the end user, she could be asked during her registration at the IdP what subset of PII attributes she would like to reveal to all RPs or to all RPs within certain categories (for example, by sector, by domain name, etc.).

If any of these attributes is encrypted at the IdP, the end user decrypts it and encrypts it again with the public key of the RP involved in this specific flow. This public key is provided by the IdP and it must be gathered during the RP registration at the IdP (the JSON Web Key specification is recommended to represent the cryptographic keys IETF). In this way, only the RP, using her private key, will be able to recover this sensitive PII.

There are two different scenarios here. The first one is when the PII is shared with the RP using the Standard Claims specified by OpenID Connect for the ID token. These claims may include PII such as *name, given_name, family_name, middle_name, nickname, preferred_username, profile, picture, website, email, gender, birthdate, zoneinfo* (time zone), *phone_number* or *address*. If the Implicit Flow is being used, the end user can substitute in the ID token the PII attributes encrypted with her secret key by the same PII attributes encrypted with the public key of the RP (via the aforementioned plugin or app) between steps 6 and 7 of Fig. 1 (during the redirection). If the Authorization Code Flow is being used, the IAAA flow must be modified (see Fig. 2) to properly prepare the Claims of the ID token for this specific RP (steps 5a, 5b and 5c).

The second scenario is when the PII is shared during the optional steps in Fig. 1, involving the RP asking to the UserInfo endpoint at the IdP (for example, a credit card number is not in the Standard Claims of the ID token). If this is the case, the end user must prepare the information shared with the RP, again decrypting the PII attributes with her private key and encrypting them with the public key of the RP (steps 5a, 5b and 5c in Fig. 2 again).

4.3.2. *Specification: flow ID*

To mitigate threats regarding profiling and tracking, we propose to change the meaning of the *sub* parameter of the ID token, instead of being a unique identifier of the end user at the IdP it should be a unique identifier of a specific IAAA flow. The Flow ID can be generated at the IdP after receiving an Authentication Request from the RP, based on the RP identifier, the end user identifier and a salt. It has to be pointed that IAAA flows do not change, only the meaning of this parameter.

4.3.3. *Specification and implementation: Privacy Arbiter*

The encryption of PII proposed before may imply a drawback when working with the OpenID Connect specification: the current version allows the IdP to verify the email and the phone number of end users (*email_verified* and *phone_number_verified* in the Standard Claims of the ID token). But if an end user, with our proposal, decides to encrypt this information, the email and the phone number, the IdP will not be able to perform any verification. RPs will have to verify this information by their selves or rely on a trusted third party to perform this validation (for example, in other context (Weingärtner and Westphall, 2017) proposes Validation Services to perform this kind of tasks).

Deepening in this concept, the need for a third-party responsible for checking and validating certain users' attributes, it could be interesting to extend this responsibility improving current levels of privacy when using the OpenID Connect specification (clearly not focused on protecting user's privacy).

The main idea is to have a third-party, different from the Identity Provider, allowing the end user to know how her attributes are being shared, to have an effective and real-time control over her data deciding to who, when and how are forwarded, to easily use metrics such as reputation, trust or risk to make decisions, to use pseudonyms, etc. To avoid significant modifications in the current specification of OpenID Connect, we propose this mitigation, mainly, as an implementation improvement, extending the IdP standard functionalities with an additional service provider, the Privacy Arbiter or PA (instead of the aforementioned Validation Service, or actually, extending it). This new provider should not have, ever, the aforementioned dual-role (RP and PA at the same time), avoiding therefore the threat of amplification that arises from account compromises, etc.

Fig. 3 shows the Authorization Code flow with OpenID Connect when a Privacy Arbiter is used (it can be also used with the Implicit Flow, it is only an example). As in previous mitigations, some kind of agent running on end users' side (a browser plugin, an app: some component at the User Agent) is required.

Again, when the step 5 of Fig. 1 (and Fig. 3) takes place, the end user not only performs her authentication or gives her consent, she also decides, explicitly, which attributes of her PII can be revealed to the RP when the PII is shared with the RP using the Standard Claims of the ID token. This would be the first modification affecting the current OpenID Connect specification.

The second modification is related to the UserInfo Endpoint, no longer required since the sharing of additional PII with RPs will be performed, if necessary, through the Privacy Arbiter. When the RP presents the Access Token at the IdP, the

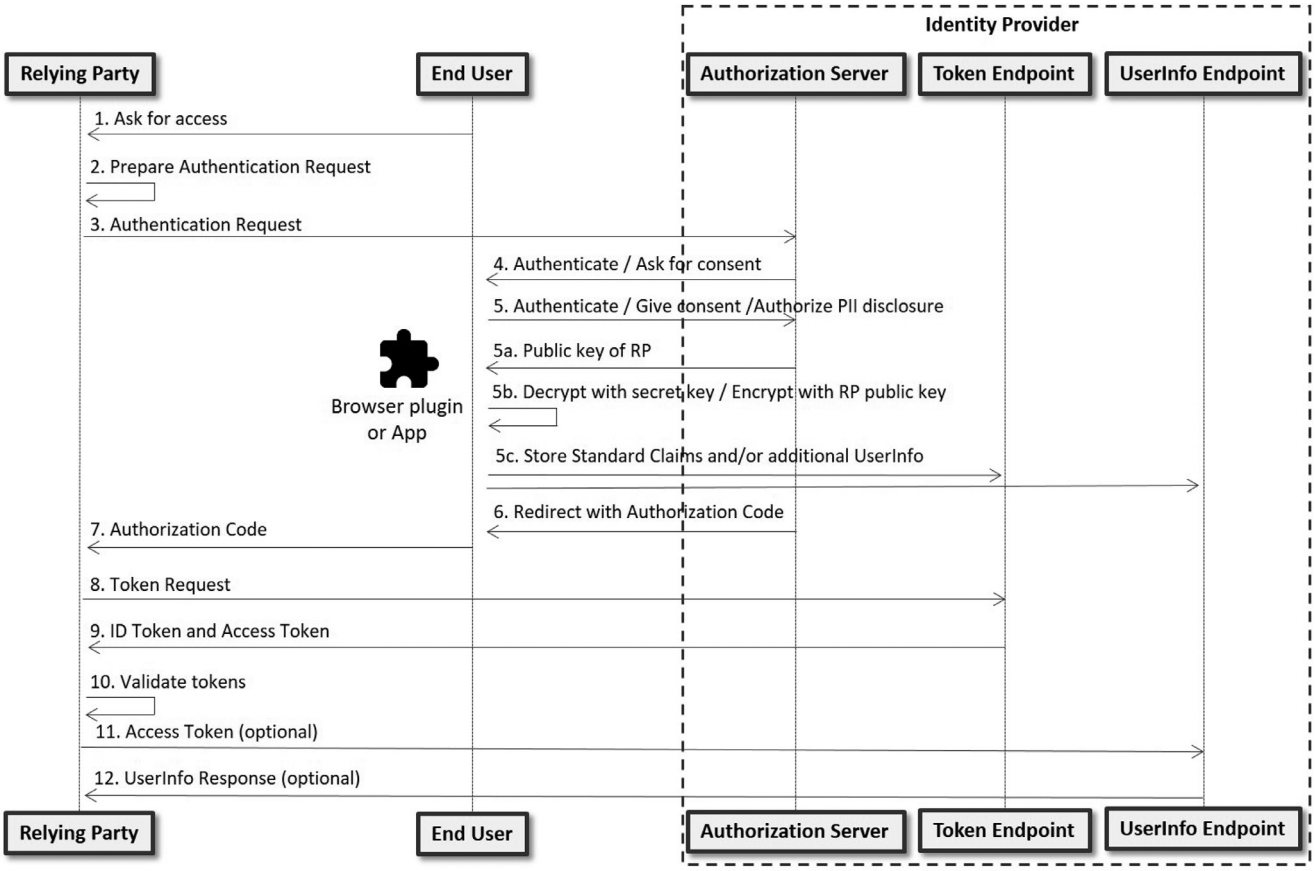

Fig. 2 – Authorization code flow with encrypted PII.

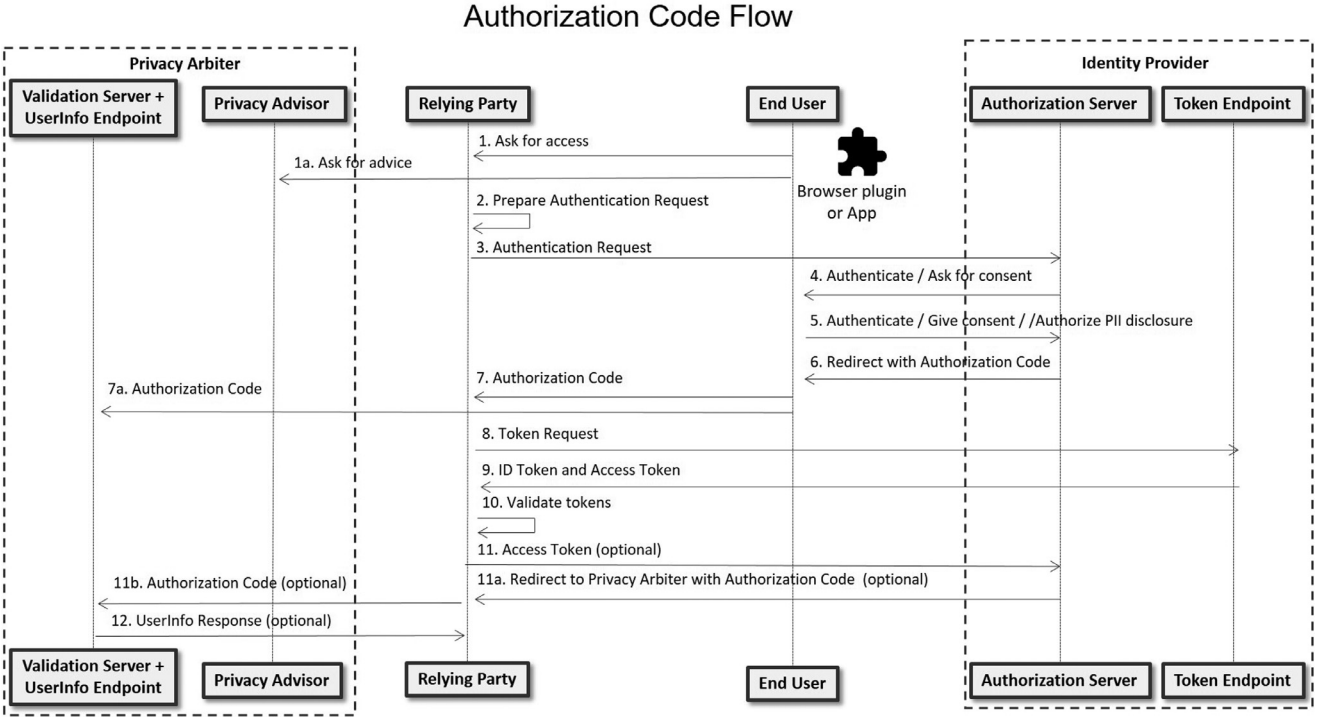

Fig. 3 – Authorization code flow with a Privacy Arbiter.

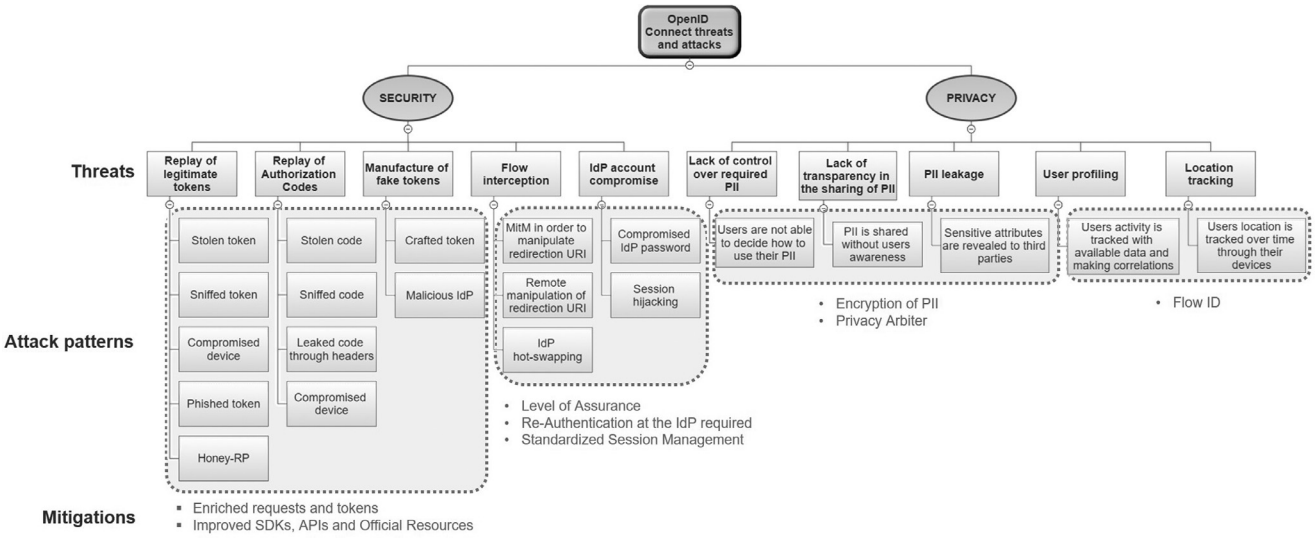

Fig. 4 – Summary of proposed mitigations in the threat and attack tree.

IdP has to redirect this information request to the Privacy Arbiter specified by the End User during her registration process.

The rest of proposed improvements involve implementation issues at the Privacy Arbiter. This research proposes two different elements for this new agent: a Privacy Advisor and a Validation Service combined with the traditional UserInfo endpoint (at the IdP in the current specification).

The Privacy Advisor communicates with the browser plugin or the app within the User Agent for each end user's request to access to a new resource, service or application (step 1a in Fig. 3). Knowing the RP involved in the IAAA flow and end user's preferences and requirements regarding privacy, the Privacy Advisor acts as a recommendation system in order to support end user's decisions, mainly, which pseudonym to use (with a specific related account and attributes at an IdP) and what information should be shared with the RP. Different advisors may rely on different mechanisms to make these recommendations such as static or dynamic policies (Orawiwattanakul et al., 2010), reputation systems (Sanchez et al., 2012) or risk modelling.

These recommendations can be implemented in all cases, to avoid disturbing the end user with decisions that can imply a complex trade off. Or, in the case of aware users, they can be treated as such, recommendations, leaving the end user the responsibility of the final decision. Although it is not represented in Fig. 3, the Privacy Advisor has other significant function, it enables disclosure awareness and accounting, allowing end users to know which data have been shared with which RPs in the past.

The Validation Service/UserInfo endpoint is responsible for performing the aforementioned validation when PII is encrypted and the IdP is not able to perform it any longer, and for sharing with RPs additional information about end users (attributes not present within the Claims of the ID token). To allow the Privacy Arbiter to identify a specific IAAA flow when a Request with an Access Token is received (step 11b in Fig. 3), the Authorization Code can be used (if an Authorization Code Flow is being used, step 7a in Fig. 3), or the Flow ID of the previous proposed improvement or a similar code or identifier. It has to be pointed that all sensitive information sent by the end user to the Privacy Arbiter is encrypted with the PA public key.

This mitigation should be used in combination with the two previously proposed, Encryption of PII and Flow ID, but all of them are independent from each other and they can be used individually.

4.4. *Discussion*

All proposed mitigations have been validated and evaluated in the controlled lab described in Section 3.1. The implementation of the proposed security and privacy improvements at the IdP infrastructure (we have developed a first prototype using the ForgeRock Identity Platform ForgeRock) is expensive in terms of software engineering because it implies important changes in the current specification, specially in the Authorization Server and Token endpoint and in its IAAA flows. But we have demonstrated that the potential benefits are worth the effort, assessing that these modifications allows us to mitigate all identified threats significantly decreasing the attack surface introduced by the use of OpenID Connect within a website or app. It has to be pointed that the costs of incorporating proposed mitigations comes from required changes in current software projects, not from a significant increase in the computational complexity of using OpenID Connect (not in terms or resource – CPU, memory – consumption). This complexity is increased only by using cryptographic mechanisms and in this case all the burden falls on the IdP (Facebook or Google in production environments) and on the Privacy Arbiter, therefore, on the agents of the flow in possession of large and powerful resources.

To reduce these costs regarding the modification of large software frameworks implementing the current version of the OpenID Connect specification a huge opportunity could be to implement proposed measures and best practices as a wrapping improving current solutions without changing core

implementations, adding as a new layer additional security and privacy capabilities.

The development of the plugin/app for the end user and of a complete Privacy Arbiter are out of the scope of this research, in our experiments for validation and evaluation their behaviour has been emulated with toy prototypes executing only their essential functionalities.

As a proof of concept, to know how easy or complicated it would be for RPs to adapt to the proposed improvements when working with the most popular IdPs, Facebook and Google, we have extended the Facebook ID PHP SDK Facebook and the Google's OAuth 2.0 APIs in Python Google to implement all the specification modifications and best practices proposed to mitigate security and privacy threats and it only required a little more than 100 code lines.

Fig. 4 summarizes the proposed mitigations, mapping each specific measure with mitigated threats and/or attacks in the threat & attack tree. It has to be pointed that all the proposed mitigations are independent of each other: only a subset of the proposed measures and practices could be included in future versions of specifications and/or implemented, although we recommended to consider them all as a whole. As it can be observed, almost all proposed mitigations imply changes in the OpenID Connect core specification, but many of them trying to add certain aspects that until now has been considered out of its scope (authentication at the IdP, session management, use of encryption for PII) or slightly modifying the meaning of certain parameters, claims, etc. Only the proposal of the Privacy Arbiter can be considered a significant modification to the current specification approach. But we think that it is conveniently justified given the introduced threat model and that our proposal minimizes required modifications in the specification.

Last, it must be considered that a new specification introducing all the recommended modifications should be analysed again in order to find new security and privacy threats. For example, if relying on the Level of Assurance improvement, we have seen during our validation experiments that downgrading attacks could be performed. Or if including the public key of a TLS connection within the ID token claims, new alternatives for user profiling may arise. As it has been mentioned before, our threat model will probably be extended with new attack patterns in the future although we hope that threat categories will remain the same.

5. Conclusions

OpenID Connect is an emerging specification built on OpenID and OAuth that enables technology providers to act as identity providers, allowing relying parties to delegate IAAA responsibilities in an interoperable and REST-like manner. This is a promising standard due to its ease of use (from end users' point of view) and to its ability to work with different client resources, applications and services running on different platforms (from relying parties' point of view).

This paper has presented a threat model for OpenID Connect, considering both, security and privacy threats. This model can be considered the basis to derive guidelines and good practices for current users and developers, and to propose new APIs and tools in order to detect and to avoid the most common implementation mistakes. But also to make motivated recommendations in order to improve future versions of the OpenID Connect specification or to implement security or privacy extensions able to mitigate or to avoid identified security and privacy threats in different contexts. Substantial improvements can be made if we take into account aspects that until now have been considered out of the scope of the specification such as the end user authentication at the Identity Provider, the session management at the RP or the treatment of Personally Identifiable Information.

It is a very interesting line for future research how to formally specify IAAA flows in future versions of OpenID Connect (or similar solutions) in a way such automatic enforcement and/or automatic testing of these flows become possible when they are implemented.

Acknowledgments

This research has been partially supported by the Government of Spain (Grant Ref. TIN2015-69542-C2-1-R and RTC-2017-6253-1) and by the Ericsson-URJC Chair ("Data Science applied to 5G").

REFERENCES

Al-Ghamdi B, Al-Anizy M, Sultan O, Al-harby F. Selfie authentication: using handheld devices through iris recognition. Int J Comput Sci Netw Secur 2017;17(10):67–80.

Alliance F. UAF and U2F specifications. https://fidoalliance.org/download/.

Bansal C, Bhargavan K, Delignat-Lavaud A, Maffeis S. Discovering concrete attacks on website authorization by formal analysis. J Comput Secur 2014;22(4):601–57.

Birrell E, Schneider FB. Federated identity management systems: a privacy-based characterization. IEEE Secur Priv 2013;11:36–48.

Chari S, Jutla CS, Roy A. Universally composable security analysis of OAuth v2.0. Proceedings of the IACR cryptology ePrint archive, 2011.

Chen EY, Pei Y, Chen S, Tian Y, Kotcher R, Tague P. OAuth demystified for mobile application developers. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security; 2014. p. 892–903.

CNDS. Professos: Fully automated evaluation-as-a-service for SSO. https://github.com/RUB-NDS/PrOfESSOS.

Dmitrienko CRA, Liebchen C, Sadeghi A. Security analysis of mobile two-factor authentication schemes. Intel Technol J 2014;18(4).

Facebook. Facebook login. https://developers.facebook.com/docs/facebook-login/.

Fett D, Küsters R, Schmitz G. A comprehensive formal security analysis of OAuth 2.0. In: Proceedings of the ACM conference on computer and communications security; 2016. p. 1204–15.

Fett D, Küsters R, Schmitz G. The web SSO standard OpenID Connect: in-depth formal security analysis and security guidelines. In: Proceedings of the IEEE thirieth computer security foundations symposium; 2017. p. 189–202.

ForgeRock. Identity platform. https://www.forgerock.com/platform/identity-management.

Ghasemisharif M, Ramesh A, Checkoway S, Kanich C, Polakis J. O single sign-off, where art thou? An empirical analysis of

single sign-on account hijacking and session management on the web. In: Proceedings of the 2twenty-seventh USENIX security symposium; 2018. p. 1475–92.
Google. Google identity platform. https://github.com/googleapis/google-api-python-client.
GSMA. MobileConnect. https://github.com/GSMA-OneAPI/Mobile-Connect.
Halpin H. NEXTLEAP: Decentralizing identity with privacy for secure messaging. Proceedings of the twelfth international conference on availability, reliability and security, 2017.
Hu P, Yang R, Li Y, Lau WC. Application impersonation: problems of OAuth and API design in online social networks. In: Proceedings of the second ACM conference on online social networks; 2014. p. 271–8.
IETF. The javascript Object Notation data interchange format (JSON). https://tools.ietf.org/html/rfc7159.
IETF. JSON web key (JWK). https://tools.ietf.org/html/rfc7517.
IETF. JSON web signature (JWS). https://tools.ietf.org/html/rfc7515.
IETF. The OAuth 2.0 authorization framework. http://tools.ietf.org/html/rfc6749.
IETF. OAuth 2.0 Threat Model and security considerations. https://tools.ietf.org/html/rfc6819.
ISO/IEC. ISO/IEC 29115 standard. https://www.iso.org/standard/45138.html.
Li W, Mitchell CJ. Security issues in OAuth 2.0 SSO implementations. In: Proceedings of the seventeenth international conference on information security; 2014. p. 529–41.
Li W, Mitchell CJ. Analysing the security of Google's implementation of OpenID Connect. In: Proceedings of the thirteenth international conference on detection of intrusions and malware, and vulnerability assessment; 2016. p. 357–76.
Mainka VMC, Schwenk J. Do not trust me: Using malicious IdPs for analyzing and attacking single sign-on. In: Proceedings of the IEEE European symposium on security and privacy; 2016. p. 321–36.
Mainka C, Mladenov V, Schwenk J, Gortz H, Wich T. SoK: Single Sign-On security an evaluation of OpenID Connect. In: Proceedings of the 2017 IEEE European symposium on security and privacy; 2017. p. 189–202.
OIDF. Certified OpenID Connect Implementationsnote = https://openid.net/developers/certified/.
OIDF. OpenID authentication 2.0. https://openid.net/specs/openid-authentication-2_0.html.
OIDF. OpenID Connect 1.0. http://openid.net/connect/.
Orawiwattanakul T, Yamaji K, Nakamura M, Kataoka T, Sonehara N. User-controlled privacy protection with attribute-filter mechanism for a federated SSO environment using Shibboleth. In: Proceedings of the 2010 international conference on P2P, parallel, grid, cloud and internet computing; 2010. p. 243–9.
OWASP. Threat risk modeling. https://www.owasp.org/index.php/Threat_Risk_Modeling.
Polleit P, Spreitzenbarth M. Defeating the secrets of OTP apps. Proceedings of the eleventh international conference on IT security incident management & IT forensics, 2018.
Reimair CKF, Marsalek A. Emulating U2F authenticator devices. In: Proceedings of the IEEE conference on communications and network security; 2016. p. 543–51.
Rosen G.. Facebook security update. https://newsroom.fb.com/news/2018/09/security-update/.
Sanchez R, Almenares F, Cabarcos P, Diaz-Sanchez D, Marin-Lopez A. Enhancing privacy and dynamic federation in IDM for consumer cloud computing. IEEE Trans Consum Electron 2012;58:95–103.
Sciarretta G, Carbone R, Ranise S, Armando A. Anatomy of the Facebook solution for mobile single sign-on: security assessment and improvements. Comput Secur 2017:71–86.
Sun ST, Beznosov K. The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems. In: Proceedings of the 2012 ACM conference on computer and communications security; 2012. p. 378–90.
Teh PS, Zhang N, Teoh ABJ, Chen K. A survey on touch dynamics authentication in mobile devices. Comput Secur 2016;59(4):210–35.
Wang R, Chen S, Wang X. Signing me onto your accounts through Facebook and Google: a traffic-guided security study of commercially deployed single-sign-on web services. In: Proceesing of the IEEE symposium on security and privacy; 2012. p. 365–79.
Weingärtner R, Westphall CM. A design towards personally identifiable information control and awareness in OpenID Connect identity providers. In: Proceedings of the 2017 IEEE international conference on computer and information technology; 2017. p. 37–46.
Werner J, Westphall CM. A model for identity management with privacy in the cloud. In: Proceedings of the symposium on computers and communication; 2016. p. 463–8.
WSO2. Identity Server. https://docs.wso2.com/display/IS541/WSO2+Identity+Server+Documentation.
Yang F, Manoharan S. A security analysis of the oauth protocol. In: Proceedings of the IEEE Pacific Rim conference on communications, computers and signal processing; 2013. p. 271–6.
Yang R, Li G, Lau WC, Zhang K, Hu P. Model-based security testing: an empirical study on OAuth 2.0 implementations. In: Proceedings of the eleventh ACM on asia conference on computer and communications security; 2016. p. 651–62.
Zhou Y, Evans D. SSOScan: automated testing of web applications for single sign-on vulnerabilities. In: Proceedings of the twenty-third USENIX conference on security symposium; 2014. p. 495–510.

Jorge Navas received the master's degree in Computer Engineering from Universidad Rey Juan Carlos of Madrid (Spain) in 2003 and he is currently finishing his doctorate at the Computing Department of the same university. Over the past years he took part in various research projects involving distributed systems and cybersecurity in both, academy and private sector. Besides his Ph.D. studies he is currently an IT engineer at INTA (National Aerospace Technology Institute). His current research interests include Threat Modelling and Identity and Access Management.

Marta Beltrán received the master's degree in Electrical Engineering from Universidad Complutense of Madrid (Spain) in 2001, the master's degree in Industrial Physics from UNED (Spain) in 2003 and the Ph.D. degree from the Department of Computing, Universidad Rey Juan Carlos, Madrid (Spain) in 2005. She is currently working with this department as an Associate Professor. She is the leader of the GAAP research group, co-founder of the Cybersecurity Cluster and she has published extensively in high-quality national and international journals and conference proceedings in the areas of computer architecture and security, and parallel and distributed systems. Her current research interests are Cloud computing, Fog Computing and Internet of Things, specifically, Identity and Access Management and privacy-preserving mechanisms for these paradigms.