Revisiting OAuth 2.0 Compliance: A Two-Year Follow-Up Study

Pieter Philippaerts
imec-DistriNet, KU Leuven
Leuven, Belgium
Pieter.Philippaerts@kuleuven.be

Davy Preuveneers
imec-DistriNet, KU Leuven
Leuven, Belgium
Davy.Preuveneers@kuleuven.be

Wouter Joosen
imec-DistriNet, KU Leuven
Leuven, Belgium
Wouter.Joosen@kuleuven.be

***Abstract*—OAuth 2.0 is a widely used authorization protocol that allows third-party access to an authorization service on behalf of a user. Like any security protocol, it requires careful implementation to ensure security. Previous research has thoroughly analyzed the security of the OAuth protocol, but popular deployments remain vulnerable due to incorrect or limited implementation of the standards. In our previous work, we introduced a tool called OAUCH to measure and improve compliance with the OAuth standards. We used the tool to measure the compliance of 100 OAuth implementations and created a unique overview of the state of practice within the OAuth ecosystem. This paper revisits these prior results and updates our measurements. We compare the latest results to the original baseline and identify changes in the ecosystem. Our analysis shows that IdPs have become more compliant in the past two years, but a substantial number still lack fundamental countermeasures.**

1. Introduction

OAuth 2.0 is a widely used authorization protocol that allows a third party, called the relying party (RP), to access an authorization service on behalf of a user. The interactions between the authorization service, hereinafter referred to as the identity provider (IdP)[1], and the RPs are precisely defined in a set of standards that make up the OAuth protocol.

Compared to its OAuth 1.0[2], OAuth 2.0 is less complex and easier to understand. However, like any security protocol, it requires careful implementation. Security specifications are scattered throughout the main OAuth standard, and additional ones are outlined in documents such as the OAuth Threat Model [12] and the OAuth Security Best Current Practices [1].

Although the security of OAuth has been thoroughly analyzed, including a comprehensive formal analysis by Fett et al. [4], [5], popular OAuth deployments remain vulnerable due to a limited or incorrect implementation of the standards.

In our previous work [13], we set out to improve the overall security of the OAuth landscape. We introduced a tool, called OAUCH, to measure how well OAuth IdP deployments adhere to the standards and to provide detailed and targeted feedback to operators to improve their services' compliance. We then used OAUCH to measure the compliance of 100 IdPs and created a unique overview of the state of practice within the OAuth ecosystem.

In this paper, we revisit our results and update our measurements of the OAuth ecosystem. Our goal is to investigate whether there have been any changes in the level of compliance of the OAuth providers over time. We compare the latest results to the original baseline and identify changes in the ecosystem. Our analysis shows that IdPs have become more compliant in the past two years, resulting in fewer unmitigated threats. Yet, despite the advancements that have been made, the rate of progress has been slow, and a substantial number of IdPs still lack fundamental countermeasures.

2. Background

Web APIs allow different software systems and applications to communicate and share data with each other seamlessly, enabling the development of complex and interconnected software ecosystems. To avoid abuse and to enable user-specific services, most of these APIs require some form of authentication and/or authorization.

When the traditional client-server authentication model is used, an app authenticates with the server by presenting the user's credentials. This behavior is not desirable, because it requires third parties to process and store user credentials. Furthermore, it reduces the service provider's ability to implement alternative authentication mechanisms (for example, two-factor authentication), to easily revoke access, and to limit the resources the third-party app can access. The popular OAuth 2.0 protocol solves many of these problems.

This section briefly introduces OAuth and the related OpenID Connect protocol. It also introduces OAUCH, a tool that has been built to measure the compliance of OAuth authorization servers with respect to the security-related specifications in the OAuth standards and best practices.

OAuth 2.0

The OAuth 2.0 protocol [6] is a widely used as an authorization framework that enables third-party applications to access a service with limited permissions. It separates the role of the *user* and the *RP*, where the user grants access to a protected resource and the RP requests access to the resource on behalf of the user. The *IdP* issues

1. In the OAuth standard, the relying party (RP), identity provider (IdP), and user are referred to as the client, authorization server, and resource owner, respectively. However, in this paper, we use the more common terminology.

2. In this paper, the focus is solely on OAuth 2.0, and not on the older and substantially different OAuth 1.0 protocol. Therefore, whenever the term *OAuth* is used, it refers to version 2.0 of the protocol.

access tokens when the user successfully authenticates and authorizes the RP to access the resource. These access tokens are linked to a user and a specific *scope*, defining granular permissions for the RP to access data or perform actions. Access tokens can be used on the *resource server* to access the protected resource.

OAuth defines four modes of operation called *grants* or *flows*, with additional flows proposed for other use cases [3], [11]. To receive authorization, an RP must first register with the IdP. During the registration process, the RP receives a *client identifier* that is used to uniquely identify the client app. In most cases, an RP also registers one or more *callback URIs* that identify the valid redirection URIs for the client. If the IdP issues a secret or uses another mechanism to authenticate an RP, the RP is said to be *confidential*. If no authentication credential is issued, the RP is called a *public RP*.

Refresh tokens may be granted with access tokens and can be exchanged for a new access token and refresh token. This allows an access token — which is frequently used and more prone to leakage — to be short-lived. When the token expires, the RP can use the refresh token to request a new access token without involving the user.

OpenID Connect (OIDC) is an authentication protocol and can be used for single sign-on and federated identity functionalities. It is built on top of OAuth 2.0 and extends the OAuth specifications with a very thin identity layer that retains compatibility. OIDC introduces the concept of an *identity token* that contains information about the authenticated user. An identity token has a well-defined structure that can be interpreted by the resource server, and it is cryptographically signed to ensure its integrity.

OAuch OAUCH [13] is an open-source tool that checks OAuth 2.0 IdP implementations for compliance with the *security best practices*. Its primary goal is to reveal potential threats and suggest security improvements.

The tool tests an IdP by executing a large number of test cases to verify compliance with security specifications established in the OAuth 2.0 standard [6], [7], and additional documents that refine the security assumptions and requirements. These documents include the OAuth threat model [12], the Security Best Current Practices [1], and others [2], [3], [8]–[10], [14]. OAUCH also supports OpenID Connect providers [15].

OAUCH leverages the OAuth threat model [12] to provide accurate feedback to users. The test cases identify the countermeasures that are implemented by the IdP, and OAUCH combines this information with the OAuth threat model to determine which threats are mitigated. For each threat, OAUCH compares the list of proposed mitigations from the threat model with the mitigations that have been detected. If all relevant countermeasures are in place, the threat is considered *fully mitigated*. If no relevant countermeasures are detected, the threat is marked as *unmitigated*. In cases where only some countermeasures have been implemented, the threat is categorized as *partially mitigated*.

3. Ecosystem

This section presents the results of the follow-up study on the OAuth ecosystem, which focused on measuring the compliance of OAuth providers with the OAuth security specifications. Our previous work [13] measured the baseline against which the new measurements will be compared. These results were collected in November 2020.

3.1. Selection of OAuth providers

The ProgrammableWeb[3] maintains a catalog of publicly available APIs and has identified 187 websites in the top 10,000 and over 300 additional websites in the top 1,000,000 that host an OAuth 2.0 IdP. In [13], we selected a representative sample of 100 IdPs.

Out of the original 100 tested websites, 92 were retested in March 2023. The remaining eight sites could not be retested due to a variety of reasons. Three sites were offline or otherwise unavailable, one site was bought (and replaced) by another site in the list, one site changed their OAuth implementation which made it incompatible with the standard, another site required payment, and the last two sites did not accept developer registrations anymore.

The sample of 92 IdPs that were tested contained 18 IdPs that support the OpenID Connect protocol. This ratio is consistent with the original dataset, where one out of every five IdPs supported OIDC. There are 68 IdPs among the top 10,000 sites, and this ratio is also in line with the original dataset.

To be able to compare the new results with the original results, the original results have been recalculated to only take into account the 92 sites that were retested. Hence, the results presented in this paper that refer to the initial measurement may slightly differ from the results reported in [13].

Ethical Considerations In accordance with our coordinated disclosure policy, we reached out to all parties involved after the initial measurement in 2020. After our new measurements, we have reached out to all IdPs for which we detected a regression (i.e., a countermeasure that was previously implemented but has since been removed), or when we identified flaws in new functionalities.

3.2. Test Cases

OAUCH implements a total of 113 test cases, each of which is based on a security specification in one of the OAuth standards. A single test case performs a basic check to verify whether the IdP correctly implementated the specification. Hence, each test case can be regarded as a binary question that confirms whether the implementation is compliant with a specific security specification or not. As a consequence, there is a direct mapping between security specifications and test cases. OAUCH automatically determines which test cases are relevant for the IdP and runs them. The number of test cases run for each IdP may vary depending on the features supported by that IdP.

An important metric is the *failure rate* of the test cases. This is computed by dividing the number of failed tests by the total number of executed tests. The failure rates can be computed per IdP, per OAuth standards document, per test case, and can be split per *requirement level*. Each security specification in OAuth is assigned one of three

3. https://www.programmableweb.com/

TABLE 1. THE AVERAGE SECURITY COMPLIANCE SCORES OVER THE ENTIRE ECOSYSTEM, LISTED PER DOCUMENT. THE PERCENTAGES ARE THE FAILURE RATES OF THE TEST CASES (LOWER IS BETTER).

	November 2020 → March 2023			
Normative Document	**Overall**	**Must**	**Should**	**May**
OAuth 2.0 (RFC6749)	24.9% ↘ 24.1%	19.7% ↘ 19.1%	39.7% ↘ 36.9%	51.7% → 51.7%
Bearer Tokens (RFC6750)	7.9% ↘ 7.5%	0.7% ↘ 0.4%	61.5% ↗ 62.2%	
Threat Model (RFC6819)	22.1% ↘ 21.2%	2.6% ↘ 1.8%	22.5% ↘ 21.6%	63.8% ↘ 61.8%
Token Revocation (RFC7009)	8.9% ↘ 6.9%	5.8% ↘ 4.3%	12.5% → 12.5%	25.0% ↘ 12.5%
JWT Grant Type (RFC7523)	30.0% → 30.0%	12.5% → 12.5%		100.0% → 100.0%
PKCE (RFC7636)	19.5% ↘ 16.1%	11.4% ↘ 8.4%		100.0% → 100.0%
Security Best Current Practices	60.3% ↘ 56.1%	38.5% ↘ 31.7%	66.1% ↘ 65.6%	85.0% ↘ 80.7%
OpenID Connect	13.1% ↗ 15.3%	13.8% ↗ 16.2%	5.6% ↘ 5.3%	
All documents combined	32.9% ↘ 31.6%	19.9% ↘ 18.9%	56.1% ↘ 54.5%	80.6% ↘ 78.8%

requirement levels (*must*, *should*, or *may*) to indicate its priority.

Table 1 lists the overall failure rates of the entire ecosystem, listed per document. With some exceptions (marked in red), the results show that the failure rate has decreased and thus compliance has improved. In particular the improved compliance with the Security Best Current Practices (BCP) is of importance, because the BCP is the main source of up-to-date security guidelines. Among the individual test cases, a few stand out based on their rate of improvement.

Require user consent for public RPs The recommended mitigation to always require explicit user consent for public RPs has seen the largest increase in adoption (a 14.7-point increase, from 44.1% to 58.8%). For clients without secrets, this countermeasure protects against impersonation attacks.

Access token rotation Token rotation enhances the security of access tokens by frequently replacing tokens with newly minted tokens. It reduces the risk of exploitation of a stolen token, as the token becomes invalid after a short period of time. According to the measurements, the adoption of this important mandatory countermeasure has increased by 9.8 percentage points to 58.9%.

Proof Key for Code Exchange (PKCE) The adoption of PKCE—a security extension that protects against authorization code injection attacks—has increased by 7.3 percentage points to 31.8%. PKCE prevents attackers from using an intercepted authorization code to obtain access tokens. Although it was initially only intended for public clients to protect against CSRF attacks, it is now mandatory to implement for all client types because of its effectiveness against authorization code injection attacks.

The regression in the results of the OpenID Connect security requirements is largely due to one site that started supporting OpenID identity tokens, but has implemented it in a way that is non-compliant. The identity tokens are missing several required security attributes and are not digitally signed, opening them up to abuse. The site's failure to pass a significant number of test cases has a magnified effect on the reported failure rates due to the limited number of sites that support OpenID.

3.3. Threats

The OAuth threat model outlines the possible attack vectors that an attacker may try to abuse and the corresponding security measures that can be implemented to mitigate them. OAUCH combines the information from the threat model with the measurements provided by the test cases to give relevant insights into the weaknesses of an IdP implementation.

OAUCH evaluates 42 threats and calculates the appropriate classification. When a threat has been completely mitigated, it is labeled as *fully mitigated*. Conversely, when there are no countermeasures in place, the threat is categorized as *unmitigated*. Threats may also be *partially mitigated* if only some of the relevant security measures have been implemented. Finally, threats can be *not relevant* if the threat does not apply to the IdP's configuration.

Figure 1 shows how the number unmitigated threats has evolved over time. The initial measurement showed that on average, each IdP had 3.9 unmitigated threats. This has slightly improved to an average of 3.6 unmitigated threats per IdP. However, the average number of partially mitigated threats has made the reverse movement from an average of 6.9 to 7.1 partially mitigated threats per provider.

Table 2 shows how the top threats have evolved over time. In general, the trend is positive, as most of the threats

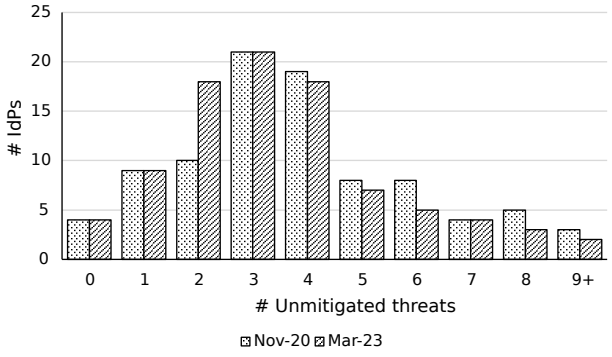

Figure 1. The histograms of the number of threats that are not mitigated. An unmitigated threat is a threat for which none of the countermeasures that are suggested in the OAuth threat model are implemented.

TABLE 2. AN OVERVIEW OF THE TOP UNMITIGATED THREATS IN THE ECOSYSTEM AND THE CORRESPONDING FAILURE RATES (LOWER IS BETTER).

Threat	Nov 2020 → Mar 2023
Auth. code injection	75.3% ↘ 68.2%
Obtaining access tokens	72.5% ↗ 73.6%
Obtaining client secrets	55.9% ↘ 39.4%
PKCE downgrade attack	42.9% ↘ 37.0%
Token leakage	39.5% ↘ 35.7%

have more mitigations in place.

Authorization code injection The uptake of the PKCE countermeasure, as described in Section 3.2, directly translates into an improved protection against the authorization code injection threat. Yet, despite the improvement, only 31.8% of the IdPs support this required countermeasure.

Obtaining access tokens There are different ways that access tokens may be revealed to an attacker. Using short-lived access tokens is a relatively simple but effective way to mitigate such risks. Although the OAuth standard does not specify the lifetime of a short-lived token, OAUCH adopts a commonly used maximum lifetime of one hour. The token lifetime has remained relatively constant over time, with one IdP even extending the lifetime of its tokens. This is surprising, because more IdPs have added support for token rotation, which allows them to reduce the lifetime of their tokens.

Obtaining client secrets Public OAuth clients do not have a client secret, which makes them vulnerable to impersonation attacks. An important mitigation for this threat is to always explicitly ask the user for consent when a public client requests authorization, even if such consent has already been given in the past. Almost a third of all IdPs that were not compliant in the initial measurement have corrected their behavior.

PKCE downgrade attack The PKCE downgrade threat's failure rate has decreased, which can be attributed to IdPs who were previously non-compliant during the initial assessment but have since corrected their implementation. In this subgroup of IdPs, the failure rate has dropped to 28.6%. However, implementing PKCE correctly to avoid the downgrade attack remains difficult. Of the IdPs that have implemented PKCE after our initial measurements, only one in three has implemented it correctly.

Token leakage Access tokens are usually sent to the API endpoint via the AUTHORIZATION HTTP header. However, some API endpoints also accept access tokens that are sent via URI query parameters. This behavior is not recommended, because these parameters may leak to log files and the HTTP referrer.

3.4. Deprecated Features

The OAuth protocol has evolved over time to improve security and functionality. Some features are now deemed outdated or no longer secure and have been deprecated. While IdPs may maintain support for deprecated features to ensure backward compatibility, certain features could potentially pose a security risk. Table 3 gives an overview of the deprecated features in OAuth and lists the number of IdPs that offer support for these features.

TABLE 3. AN OVERVIEW OF THE NUMBER OF IDPS IN THE ECOSYSTEM THAT SUPPORT VARIOUS DEPRECATED FEATURES (LOWER IS BETTER).

Deprecated Feature	Nov 2020 → Mar 2023
Old versions of TLS	46 ↘ 23
Implicit flow	35 → 35
Password flow	2 → 2
OIDC *id_token token* flow	5 ↘ 4
OIDC *code token* flow	4 ↘ 3
OIDC *code id_token token* flow	4 ↘ 3
Plain PKCE	12 ↗ 17

Old versions of TLS TLS 1.0 and 1.1 have been officially deprecated in March 2021 by the IETF, but already half of the tested sites in the initial measurement had removed support for these older versions. After the official deprecation, another 25% of IdPs removed support, leaving just one quarter of all IdPs that support the older protocols.

Deprecated flows OAuth and OpenID Connect have deprecated certain flows that were initially supported to enable certain use cases, but are now deemed insecure. Most IdPs continue to support the same flows, with the implicit flow being particularly popular. We have not observed significant changes in this regard.

Plain PKCE The PKCE standard defines two variations of the countermeasure: hashed and plain. The hashed version is mandatory-to-implement and has better security properties. However, the standard also defined a simpler (and less secure) version that avoids hashing. Although plain PKCE is not recommended for use, many IdPs implement it because it is part of the standard. The increase in support for plain PKCE compared to the initial measurement, as shown in Table 3, is a testament to the improved adoption of the PKCE standard.

4. Conclusion

In this paper, we presented an update to the security compliance analysis of the OAuth 2.0 ecosystem that was originally published in [13]. We used the OAUCH tool to repeat the analysis of 92 IdPs and compared the results with the original baseline

The analysis shows that IdPs have become more compliant in the past two years. The test case failure rates are consistently lower over all the requirement levels and the various OAuth standard documents, with only a few exceptions. This indicates that IdPs implement more countermeasures and support fewer deprecated features. This improved compliance has a positive effect on the number of (partially) mitigated threats. We did not find significant differences between high-ranked and low-ranked websites in terms of improved compliance.

Although progress has been made, we find that there are still problems with many implementations. A considerable number of IdPs still lack essential security measures and have multiple threats that are left completely unmitigated.

Acknowledgements

This research is partially funded by the Research Fund KU Leuven, the APISEC project, and by the Flemish Research Program Cybersecurity.

References

[1] John Bradley, Andrey Labunets, and Daniel Fett. OAuth 2.0 security best current practice. https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics, October 2020. [Online; accessed May 20, 2021].

[2] Brian Campbell, John Bradley, Nat Sakimura, and Torsten Lodderstedt. OAuth 2.0 mutual-TLS client authentication and certificate-bound access tokens. https://datatracker.ietf.org/doc/html/rfc8705, February 2020. [Online; accessed May 20, 2021].

[3] William Denniss, John Bradley, Michael Jones, and Hannes Tschofenig. OAuth 2.0 device authorization grant. https://datatracker.ietf.org/doc/html/rfc8628, August 2019. [Online; accessed May 20, 2021].

[4] Daniel Fett, Ralf Küsters, and Guido Schmitz. A comprehensive formal security analysis of OAuth 2.0. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*. Association for Computing Machinery, 2016.

[5] Daniel Fett, Ralf Küsters, and Guido Schmitz. The web SSO standard OpenID Connect: In-depth formal security analysis and security guidelines. In *Proceedings of the IEEE 30th Computer Security Foundations Symposium (CSF'17)*, pages 189–202, 2017.

[6] Dick Hardt. The OAuth 2.0 authorization framework. https://datatracker.ietf.org/doc/html/rfc6749, October 2012. [Online; accessed May 20, 2021].

[7] Dick Hardt and Michael Jones. The OAuth 2.0 authorization framework: Bearer token usage. https://datatracker.ietf.org/doc/html/rfc6750, October 2012. [Online; accessed May 20, 2021].

[8] Michael Jones and Brian Campbell. OAuth 2.0 form post response mode. https://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html, August 2015. [Online; accessed May 20, 2021].

[9] Michael Jones, Brian Campbell, and Chuck Mortimore. JSON Web Token (JWT) profile for OAuth 2.0 client authentication and authorization grants. https://datatracker.ietf.org/doc/html/rfc7523, May 2015. [Online; accessed May 20, 2021].

[10] Torsten Lodderstedt, Stefanie Dronia, and Marius Scurtescu. OAuth 2.0 token revocation. https://datatracker.ietf.org/doc/html/rfc7009, August 2013. [Online; accessed May 20, 2021].

[11] Maciej Machulak and Justin Richer. User-managed access (UMA) 2.0 grant for OAuth 2.0 authorization. https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html, January 2018. [Online; accessed May 20, 2021].

[12] Mark McGloin and Phil Hunt. OAuth 2.0 threat model and security considerations. https://datatracker.ietf.org/doc/html/rfc6819, January 2013. [Online; accessed May 20, 2021].

[13] Pieter Philippaerts, Davy Preuveneers, and Wouter Joosen. OAuch: Exploring security compliance in the OAuth 2.0 ecosystem. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses (RAID'22)*, RAID 2022, pages 460–481. Association for Computing Machinery, 2022.

[14] Nat Sakimura, John Bradley, and Naveen Agarwal. Proof key for code exchange by OAuth public clients. https://datatracker.ietf.org/doc/html/rfc7636, September 2015. [Online; accessed May 20, 2021].

[15] Nat Sakimura, John Bradley, Michael B. Jones, Breno de Medeiros, and Chuck Mortimore. OpenID Connect. https://openid.net/specs/openid-connect-core-1_0.html, November 2014. [Online; accessed May 20, 2021].