Performance Evaluation of a Novel JWT Revocation Algorithm

Laszlo Janoky[1] Janos Levendovszky[2] Peter Ekler[3]
Department of Automation and Applied Informatics[1,3]
Department of Networked Systems and Services[2]
Budapest University of Technology and Economics
{laszlo.janoky[1],peter.ekler[3]}@aut.bme.hu, levendov@hit.bme.hu[2]

Abstract. JSON Web Tokens (JWT) are a technology that is widely used for securing distributed web applications. The main advantage of such tokens is that they can be verified without accessing a central authority. On the other hand, the lack of centralized access means that revoking a JWT is not a trivial task. In our previous works in the field, we enumerated the possible revocation approaches and introduced our novel solution, which improved on them. In this work, we aim to evaluate the performance of different solutions more in-depth, backed by a common mathematical framework.

Keywords: JSON Web Tokens, JWT, Token revocation, Performance modeling

1 Introduction

Our previous work in the field included the examination of different revocation mechanisms for JSON Web Tokens [1] and the introduction of our novel approach [2]. This paper further elaborates on the performance aspect of the later. We provide the general basis of comparison and examine each method accordingly.

The paper is structured as follows: in this first section (I), a quick overview of the current revocation schemes are enumerated, their main characteristics presented. The second section (II) is dedicated to the description of our novel approach. In the third section (III), we introduce a mathematical model for dealing with the performance comparison of different methods. In the fourth section (IV), we wrap the discussion by drawing conclusions and outlining future developments.

1.1 About JSON Web Tokens

A JWT called access token is typically used to authenticate and authorize user access to a protected resource. To ensure the validity of token, a cryptographic scheme is usually employed, such as digital signatures [3]. In a general way, we simply refer to the signing key or the public key as the secret. In typical

scenarios, a refresh token is usually employed along with the access token. This second token is used by the clients to acquire new access tokens.

When a client wants to log out from the system, the refresh token is destroyed, and existing JWT tokens should be revoked. The revocation is not trivial, as the validity of a token is only determined by the cryptographic assurance; there is no easy way to invalidate it.

1.2 Common revocation methods

In the industry, there are three standard methods of revocation that are currently used [4].

1.2.1 Short-lived tokens

In the most common case, each token has a limited - usually very short - lifespan. In this method, a token is never directly revoked. Instead, the means of a client acquiring a new token is revoked. Therefore, when the access token's lifetime runs out, no new token can be acquired, so the client's access to the system is revoked.

1.2.2 Blacklist

In the blacklist scheme, revoked tokens are stored in a shared location (usually a database), where the consuming services can lookup invalidated tokens. The downside of this method is that it requires checking the shared storage for each token validation. Even if a token is valid (not on the blacklist), this checking must be done. Ultimately, this means that the validity of the token cannot be determined in itself. Therefore this method loses one of the main benefits of using the JWT scheme.

1.2.3 Secret change

The secret change method, while applicable in theory, is rarely used in a real-world application. The basis of the method is the changing of the cryptographic secret on client logout. Changing this secret means that every other access tokens are also revoked. While incurring a significant performance hit, using the refresh tokens, the still logged in users can request new ones for staying in the system.

2 Our novel revocation algorithm

The revocation strategy we came up with is based on the secret change method. The basic idea is to minimize the impact of token revocations on non logged out clients while keeping the main architectural benefits of the JWT scheme intact. In the following section, we provide an overview of our novel approach.

2.1 Basic principle

When the secret is changed, all issued tokens are revoked. This means that if a client logs out, every other active user must also request a new token.

In our method, we arranged the users in groups to overcome the performance hit of unnecessary token revocations. The grouping can be done on any arbitrary method, such as the hash of usernames. Each group has a different secret. Therefore a revocation only means changing the secret in each group, limiting the amount of total token request.

Log-outs events can be modeled statistically; this means that optional group sizes can be calculated to minimize the total performance impact on the system, while still maintaining a manageable number of groups and secrets.

2.2 Secret change event propagation

With our method, revocation is instantaneous, and the main architectural advantages of the JWT scheme remain intact, and tokens are valid without the need for a central authority.

A requirement of our solution is the availability of a channel for propagating the information about the secret change event. The channel must be available between the token consuming services and the token issuer.

This channel may be unavailable in certain cases; to overcome this limitation, we came up with an alternate solution for propagating secret change events. The basis of the solution is that the secret is generated by a deterministic pseudo-random number generator [5]. Each token consuming service is initiated with the same seed, keeping track of the current active secret. When a token in the group is revoked, and the secret is changed, the next random number is taken, and the new tokens are issued with this new secret. When a consuming service still uses the old secret and receives a token signed with a new secret (the next value from the random number generator), it will update its stored secret accordingly.

This alternate approach provides eventual consistency for the system in the long term. As a trade-off for the relaxed consistency guarantees, the need for the secret change propagation channel is dropped. This also means that instantaneous revocation is no longer possible, an invalidated token is only revoked after the new secret is assigned (another token, using this new secret is received by the service).

3 Performance model

To accurately model the performance of different revocation methods, a common mathematical framework must be set up. This section deals with the common model for performance comparison and the evaluation of the different methods using this framework.

3.1 Cost model

As the basis of the performance comparison framework, we identified a set of basic operations, which make up the performance cost of each method. Each operation has an implementation dependant cost, which can be considered as a system characteristic, a parameter of the model. These main parameters are the following:

- C_c **(Communication cost):** the cost of system components communicating with each other.
- C_d **(Data access cost):** the cost of accessing data stored in a shared storage.
- C_i **(Issue cost):** the cost of issuing a new token.
- C_v **(Validation cost):** the cost of checking the validity of a token.

To predict the overall performance impact of a revocation scheme on the system, the next step is to determine the number of individual operations occurring in a given time. The main performance cost of a system is determined by the clients consuming its services. By measuring and characterizing the metrics of client sessions, it is possible to come up with predictions of their impact on the system [6]. The following client metrics have to be measured to determine the performance impact.

- N**:** number of clients in the system.
- $f_i(t)$**:** probability distribution of client session lengths.
- r**:** average number of resource access / client / second.

As demonstrated in our previous work on the topic, one can calculate the average time between token revocations in a group of clients from these metrics. This value is denoted as T_{rvk}. Knowing the occurrences of typical operations in the system, it is possible to determine the overall cost function.

To predict the performance characteristics of a system, one must determine the typical cost associated with the operations defined previously and measure the characteristics of the client population. With these metrics known, one can choose an optimal revocation algorithm.

Each revocation algorithm has its own cost function. This function uses the client pool behavior to determine the number of operations. These operations can then be used to calculate the overall performance cost of a given method.

Some revocation algorithms have parameters, which can be used to optimize their performance.

3.2 Short-lived tokens

The short-lived method has a single parameter, T_{life}, which denotes the lifetime of a token. To maximize the performance of this approach, this T_{life} should be chosen to the maximum tolerable time for token revocation after logout.

The overall cost function is made up of two main factors. The first one is the cost of validating the tokens sent along with the request, the second one being the cost of issuing new tokens to replace the expiring ones.

$$C = (N * r * C_v) + (N * \frac{1}{T_{life}} * C_i)$$

3.3 Blacklist

In the blacklist scheme, the main factor of the cost comes from checking each token received in a request against the shared blacklist. This extra step on each request makes this approach the worst in terms of scaling with the number of requests. The cost function of this method is the following:

$$C = N * r * C_v * C_d$$

3.4 Secret change

In the case of the secret change approach, the same performance cost of authorizing incoming requests can still be found. On top of this load, the task of generating new tokens for each client still in the system is added.

$$C = (N * r * C_v) + (N * \frac{1}{T_{rvk}} * C_i)$$

Notice that the formula for secret change is similar to the one associated with the short-lived method. This can be attributed to the fact that in both cases, the number of token revocations depends on the average lifetime of a token. In the short-lived case, it is determined by the age of token, while in the secret change case, client logout events trigger it.

3.5 Our novel algorithm

Because our method is built on the secret change event, the cost function can be constructed in a similar way too. The main difference is the introduction of parameter K, which denotes the number of client groups. With different groups, the T_{rvk} - which was calculated for the whole client population in the secret change method - should be calculated for the number of clients in a group, denoted as $\frac{N}{K}$. This group-level revocation time is denoted by T_{rvk}. As K increases, T_{rvk} monotonously increasingly approaches the mean value of $f_i(t)$.

$$C = (N * r * C_v) + (K * \frac{1}{T'_{rvk}})(\frac{N}{K} * C_i + C_c)$$

As for the application of this model, in our previous work in the field, we prove that a system using our approach can be parametrized in a way, that our solution provides the best performance of the different options.

4 Overview

In this paper, we gave an overview of the JWT revocation problem and introduced the current solutions. After that, we introduced our novel solution, with the intention to improve on the performance of current solutions, while keeping the architectural gains of the JWT auth scheme.

We described a common mathematical model for comparing the performance of different solutions in different systems. This model works by determining the basic operations of a system while dealing with a revocation scheme. The cost of these common operations can then be measured and used in actual comparisons. We also outlined the necessary characteristics to measure in a client population of such a system. Using the cost operations and client model, we determined the cost function of each solution.

With the mathematical framework we provided, one can determine the optimal revocation method for any system where JWT revocation is necessary. In schemes, where the cost function has additional parameters, traditional minimizing approaches can be used to find the optimal solution.

Ultimately, we hope that our work will aid the capacity planning and system design of distributed systems using the JWT auth scheme.

Acknowledgements

The research reported in this paper was supported by the BME Artificial Intelligence TKP2020 IE grant of NKFIH Hungary (BME IE-MI-SC TKP2020) and the Janos Bolyai Research Fellowship of the Hungarian Academy of Sciences.

References

[1] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, RFC Editor, May 2015. Published: Internet Requests for Comments.

[2] L. V. Janoky, J. Levendovszky, and P. Ekler, "An analysis on the revoking mechanisms for JSON Web Tokens," *International Journal of Distributed Sensor Networks*, vol. 14, p. 1550147718801535, Sept. 2018. Publisher: SAGE Publications.

[3] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Signature (JWS)," RFC 7515, RFC Editor, May 2015. Published: Internet Requests for Comments.

[4] dWTV, "Learn how to revoke JSON Web Tokens," July 2017.

[5] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudorandom bits," *SIAM journal on Computing*, vol. 13, no. 4, pp. 850–864, 1984.

[6] M. Arlitt, "Characterizing web user sessions," *ACM SIGMETRICS Performance Evaluation Review*, vol. 28, no. 2, pp. 50–63, 2000.