# Auditable Authorization

Igor Zboran
izboran@gmail.com

***Abstract*—Bearer tokens are vulnerable at rest and in transit when an attacker is able to intercept a token to gain illegal access to private information. In order to mitigate some of the risks associated with bearer tokens, an authorization audit trail may be used alongside the bearer tokens during the authorization process. The authorization audit trail (hereinafter simply referred to as "audit trail") is the intrinsically recorded transaction flow of the authorization process. It comprises cryptographically chained blocks of data bearing chronological tamper-resistant records of all their possessors and the changes that have been made by them. A nested, chained MAC construction (e.g., HMAC) is used to ensure the authenticity and integrity of the audit trail. All sensitive information is encrypted to preserve confidentiality. The access control relies on a real-time auditability of the audit trail by the authorization server. The audit trail concept is compatible with existing OAuth 2.0 and UMA protocols.**

## I. Introduction

Bearer tokens are easy to use and easy to integrate with any client, server, and mobile device. Once they have been minted on the authorization server, they don't require additional processing on the client, and token validation on the resource server is trivial. While existing technology standards are acceptable for many use cases, they have been harder adopted in the healthcare industry and financial sector, where a higher degree of authenticity and integrity of the authorization flow is required. To overcome these restraints, additional data alongside the access token should be provided. The auditable authorization mechanism aims to provide these data by recording detailed information during the authorization flow. The resulting audit trail increases the level of security, while ensures compliance with financial and legal regulations by documenting all actions and events of the authorization flow.

## II. Concept

The auditable authorization concept is based on a verifiable tamper-resistant audit trail created by authenticated participants (authorization server, client, resource server) during the respective stages of the authorization process. The audit trail carries information in the form of a sequence of records organized into blocks. Each block comes from an individual participant—a block possessor. Records and blocks are chained using the MAC value of the previous record.

### *A. Record Chaining*

To create a chain of records, we use the HMAC chaining construct $MAC = \text{HMAC}(K, \text{HMAC}(MAC, m))$, broken down into individual MACs,

$MAC = \text{HMAC}(MAC, m)$
$MAC = \text{HMAC}(K, MAC)$

which forms the basis of the record chaining mechanism.

To simplify notation, we use the Double HMAC construct—a nested HMAC function, denoted by DHMAC, that takes 3 inputs ($K$, $MAC$, $m$) and outputs a message authentication code

$$MAC = \text{DHMAC}(K, MAC, m) = \text{HMAC}(K, \text{HMAC}(MAC, m))$$

where $K$ is the secret key, $MAC$ is the input message authentication code, and $m$ is the message to be authenticated.

### *B. Block Chaining*

The block possessor must be registered at the authorization server (public clients can use dynamic registration to become confidential clients).

Each block contains four mandatory records:

- The random NONCE to prevent replay attack.
- The timestamp of when the block was created.
- The URI that identifies who created the block.
- The MAC value of the last record copied from the previous block.

Additional groups of optional records can be added at any time until the block is sent to the next possessor.

### *C. Verification*

The audit trail is reviewed in real-time by the authorization server that acts as an auditor. Given that, the resource server uses the token introspection endpoint of the authorization server to verify the audit trail and look up relevant information about the authorization process. The verification system of the authorization server is able to recreate MACs of the individual blocks of the audit trail using the shared secrets of the authorization flow participants. Thus, if any block of the audit trail was altered, then the audit trail tampering can be detected by comparing the recreated MACs with the audit trail MACs.

## III. Conclusion

By utilizing simple cryptographic techniques, we can build the audit trail that, when used alongside the bearer tokens, mitigates the risk that an adversary could gain illegal access to private information by stealing a token. As part of the broader applicability of the audit trail, the recorded data can later be used by government agencies and businesses for forensic analysis and compliance verification.

## IV. Future Work

As a part of future work, we plan to explore the possibility of replacing the bearer tokens with an audit trail and storing JWT claims as a set of records directly into chained blocks.

## Acknowledgment

This work has benefited from the valuable WG-UMA discussions on Audit in UMA [1], [2], [3], and from the Audit in OAuth 2.0 IETF Draft [4].

[1] WG-UMA, Audit in UMA, Wed Jul 16 2014, https://kantarainitiative.org/pipermail/wg-uma/2014-July/002889.html.
[2] WG-UMA, Audit in UMA, Mon Aug 18 2014, https://kantarainitiative.org/pipermail/wg-uma/2014-August/002966.html.
[3] WG-UMA, Audit in UMA, Mon Aug 18 2014, https://kantarainitiative.org/pipermail/wg-uma/2014-August/002967.html.
[4] Zh. Tsitkov, “Audit in OAuth 2.0”, IETF Draft, 2015, https://datatracker.ietf.org/doc/html/draft-tsitkov-oauth-audit-02.