



Transaction Tokens

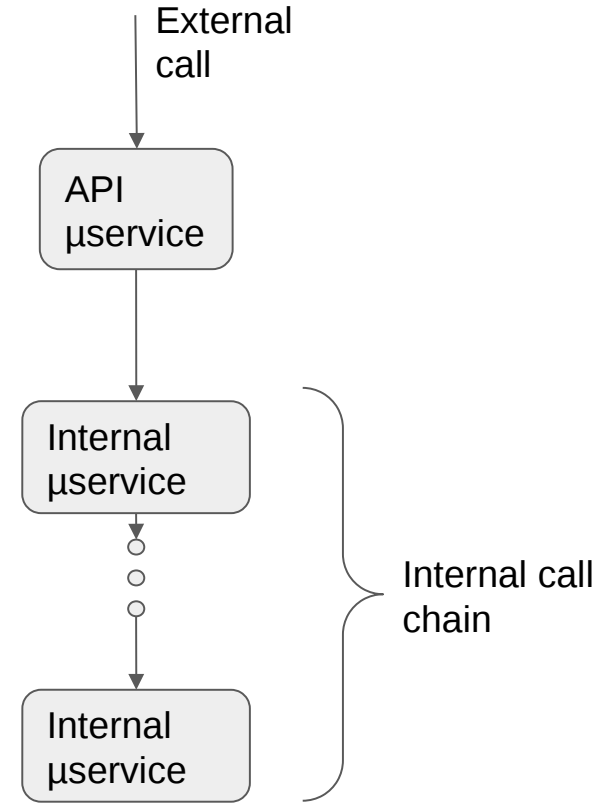
Atul Tulshibagwale
SGNL

George Fletcher
Capital One

Pieter Kasselmann
Microsoft

Why Transaction Tokens?

- Microservices architectures result in call chains to service a single external call
- API microservice logically encapsulates any network infrastructure layers
- Calls are short-lived
- Batch jobs can be thought of as a sequence of external calls made by the robotic principal of the batch job. Each such call is short-lived



Why Transaction Tokens?

- Individual microservices in a VPC may be compromised
 - Software supply chain attacks
 - VPC compromise
 - Other attacks
- Attacks can result in
 - User impersonation
 - Arbitrary method invocation

Why Transaction Tokens?

- Transaction Tokens limit damage by
 - Preserving context immutably throughout a call chain
 - Services can assert that they have processed a call to downstream services
- Context may include
 - Identity of user or robotic principal initiating the external call
 - Parameters provided by the external caller
 - Other data that needs to be preserved in the call chain (including environmental properties, e.g. device identifier, or computed values, e.g. “customer category”)

Why Transaction Tokens?

- Transaction Tokens limit internal calls to
 - Identity provided by external caller
 - Using pre-created context
 - Cannot impersonate intermediate services in a call chain
- With Transaction Tokens, fewer services need higher security controls
 - Enables more agility / development velocity for most services

What is a Transaction Token

- Short-lived OAuth token
- Uniquely identifies a call chain
- Asserts context that needs to be preserved in a call chain
 - User identity
 - Transaction identifier
 - Other context

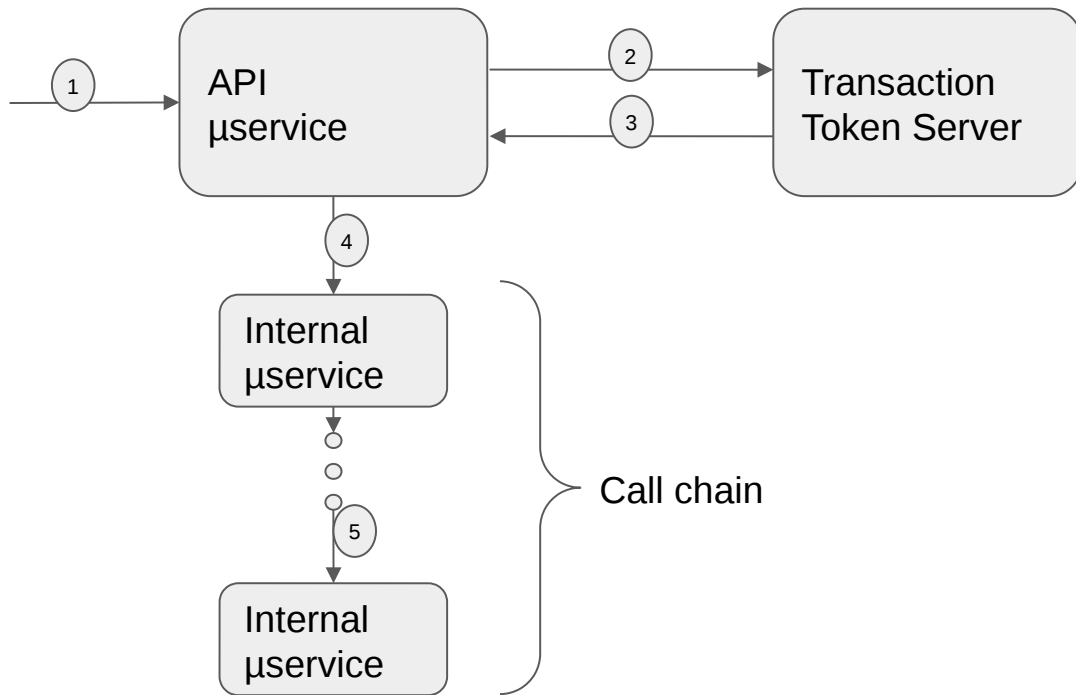
What a Transaction Token Looks Like

```
{
  "iss": "https://trust-domain.example/tx-token-service",
  "iat": "1686536226000",
  "exp": "1686536526000",
  "tid": "97053963-771d-49cc-a4e3-20aad399c312",
  "sub_id": {
    "format": "email",
    "email": "user@trust-domain.example"
  },
  "azc": {
    "action": "BUY", // parameter of external call
    "ticker": "MSFT", // parameter of external call
    "quantity": "100", // parameter of external call
    "user_ip": "69.151.72.123", // env context of external call
    "user_level": "vip" // computed value not present in external call
  }
}
```

Comparison with Access Token Format

- Common Fields:
 - “iss”, “iat”, “aud”, “exp”, “jti”
- Missing in Txn-Tokens
 - “sub”
- Unique to Txn-Tokens
 - “tid”, “azc”, “sub_id”
- No Refresh Tokens for Transaction Tokens

Creating Transaction Tokens



1. User invokes external endpoint in API microservice
2. External microservice obtains transaction token from “Transaction Token Server”
3. Transaction Token Server verifies requesting service using SPIFFE and issues Transaction Token
4. API service uses Transaction Token to call internal service
5. Subsequent services in call chain use Transaction Token to invoke downstream services

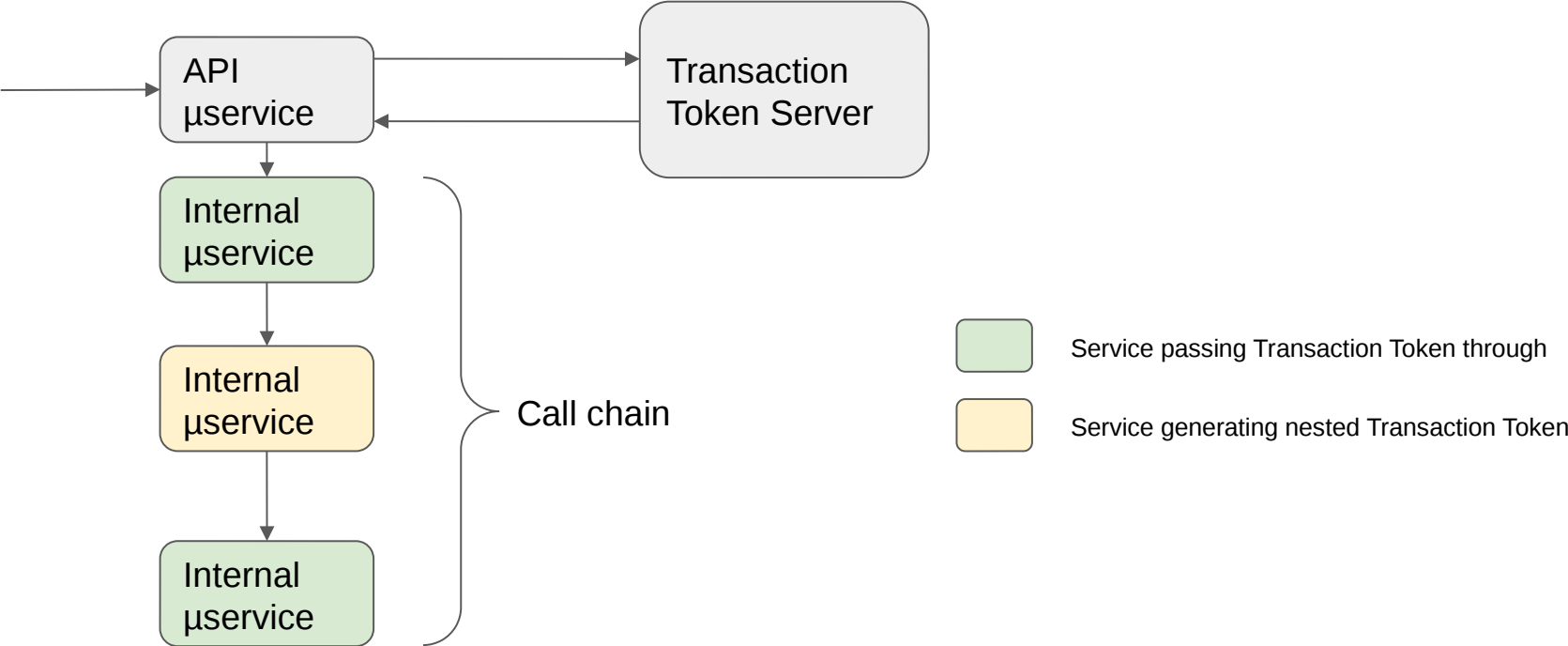
Creating Transaction Tokens

- Transaction Token Request is a OAuth 2.0 Token Exchange Request
 - **subject_token** is external token
 - **subject_token_type** is external token type
 - **azc**: Additional parameter in request contains information required to generate Transaction Token
- Transaction Token Response
 - Issued **token_type** is “**txn_token**”
 - Transaction Token is contained in **access_token** value
 - Response MUST NOT contain:
 - **expires_in**, **refresh_token**, and **scope**

Nested Transaction Tokens

- Self-signed JWT Embedded Token, contains Transaction Token
 - May be nested recursively
- Benefits:
 - Downstream service can verify that signing service was in the call chain
- Drawbacks:
 - Token bloat
 - More trusted services

Creating Nested Transaction Tokens



Replacing a Transaction Token

- An intermediate service may replace an existing Transaction Token
- Benefits
 - Reduce token bloat
 - Updated context for downstream services
- Drawbacks
 - Can completely negate value of Transaction Tokens
 - Increased traffic to Transaction Token Server

Creating a Replacement Transaction Token

- Similar to creating Transaction Tokens
 - **subject_token** is the Transaction Token to be replaced (may be a nested Txn-Token)
 - **subject_token_type** is “**txn_token**”
- Transaction Token Server Responsibilities
 - Should not enable arbitrary modifications to previously asserted values
 - May reduce scope of asserted values
 - May add to asserted values
 - Remove any nesting
 - May include call chain information of services that had created nested Txn-Tokens

Creating Replacement Transaction Tokens

