Capability-Based Authorization and Resource Control

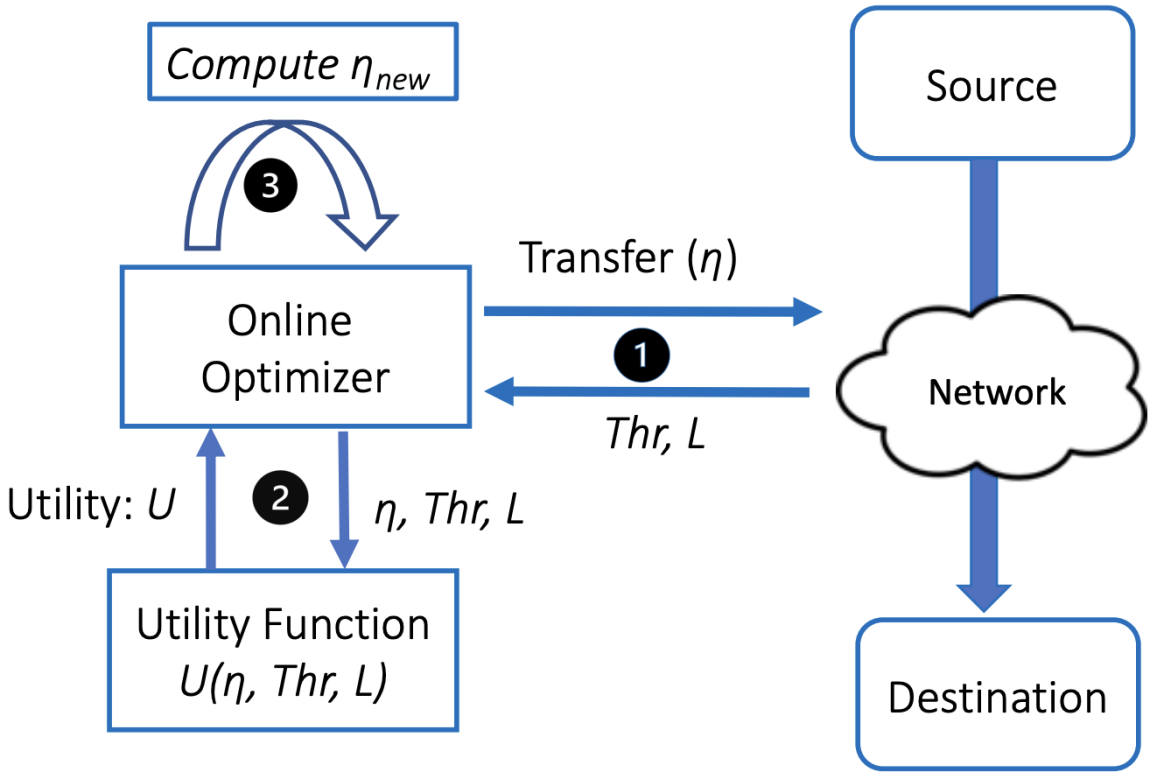
Md Arifuzzaman
arif@nevada.unr.edu
Computer Science & Engineering
University of Nevada, Reno

Introduction

Data transfers in high-performance networks require network and I/O parallelism to reach high resource utilization, known as concurrency. To optimize performance, data transfer agents might increase concurrency, which often overwhelms the end systems and networks. Systems administrators might limit the maximum concurrency value per user to avoid this. They may also cap bandwidth usage per user to maintain fair resource sharing among them. Also, users have filesystem read-write restrictions attached to their profiles. Transfer agents must be aware of these user-specific scopes to properly transfer data from source to destination. Currently, data transfer applications use the certificate-based authentication model, which is too permissive and do not contain well-defined scopes as they can only verify user identities. Which raises potential security and resource abuse concerns. The Scitokens project tries to address these concerns by replacing certificates with Json Web Tokens (JWT). In addition to verifying user identities, JWT can also contain scopes for the users for target audiences, which can facilitate data transfer operations more securely and conveniently.

Falcon - Online Data Transfer Optimization

Falcon aims to maximize high-speed data transfers performance via online blackbox optimization. Falcon agents are responsible for transferring data from source to destination, while the Falcon-server collects metrics from agents and performs online optimizations. At the beginning, the optimizer begins with random/minimal initial configurations η, agents report back throughput and packet loss for the respective η in a predefined interval. Using these observations the optimizer calculates new η and sends it back to agents. This process is repeated until the transfer is completed.



Combining Scitokens with Falcon

Falcon needs properly defined user scopes to securely transfer data among HPC sites and Scitokens solves this problem effectively. Scitokens library provides interoperable and capability-based Json Web Token. In addition to authenticating the user, the issued tokens can have minimal scopes required to perform the data transfer tasks, thus preventing bad actors from security pitfalls and resource consumption malpractices in addition to conveniently providing allowed scopes information to Falcon agents.

Project Architecture

We envision three different types of entities, a centralized Token server, a centralized Falcon Server, and many Falcon agents for different HPC institutes. The centralized token issuance server will issue tokens for specific audiences (data source and destination). The central Falcon server will be responsible for scheduling transfers. Users can login into the Falcon server via federated login and create transfer tasks. Then Falcon server can request tokens from the Token server for the specific user and audiences, and pass tokens to the target falcon agents. Finally, the Falcon agents running on each site are responsible for the data transfer between source and

destination. They subscribed to their respective message queue channel for new tokens, verify and parse scopes from it using the Token server public key, and begins data transfer.

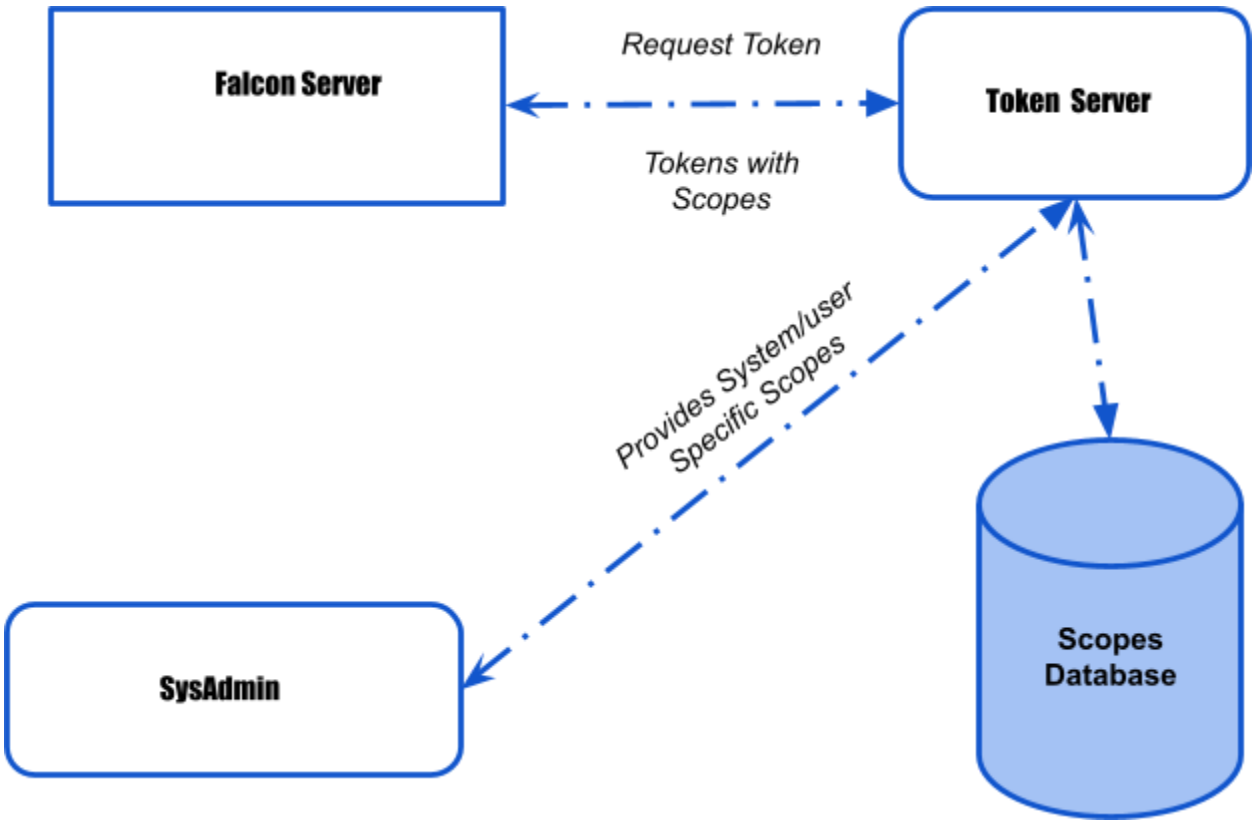
Defining Transfer-Specific Token Scopes

1. Maximum Concurrency:
    a. Could be a single value specifying read, write and transfer threads.
        i. system-wide: *concurrency:/5*
        ii. user-specific: *concurrency:/marifuzzaman/5*
    b. Or it could be more fine-grained. Separate restrictions for different types of resources.
        i. network-specific: *concurrency.connection:/5*
        ii. I/O-specific: *concurrency.read:/5, concurrency.write:/5*

2. Maximum Bandwidth:
    a. Could be the ratio of total bandwidth such as *30% or 0.3*. But fixed values are preferable. As the total available bandwidth value is pretty ambiguous, the restriction enforcement for ratio-based limits might be tricky.
        i. *bandwidth.bps: 10000000, bandwidth.bps: NA*

3. Direct I/O:
    a. A boolean flag to indicate if direct I/O is permitted. As direct I/O has an impact on file caching, systems admins can choose to disable it.
        i. For example:
            1. *falcon.directio:/false,*
            2. *falcon.directio:/marifuzzaman/true*

4. Storage Level:
    a. Put read permission for the source token
        i. *read:/source_directory*
    b. Put write permission for the destination token
        i. *write:/destination_directory*



Token Server

1. Maintain a scopes database for participating institutes. Each institute can provide systems-wide or user-specific scopes.

2. Scopes will have predefined namespaces/formats for convenience and wider adaptability.

3. If the scopes entry of the user for the target audience does not exist, then the server will fetch system-wide generic scopes.

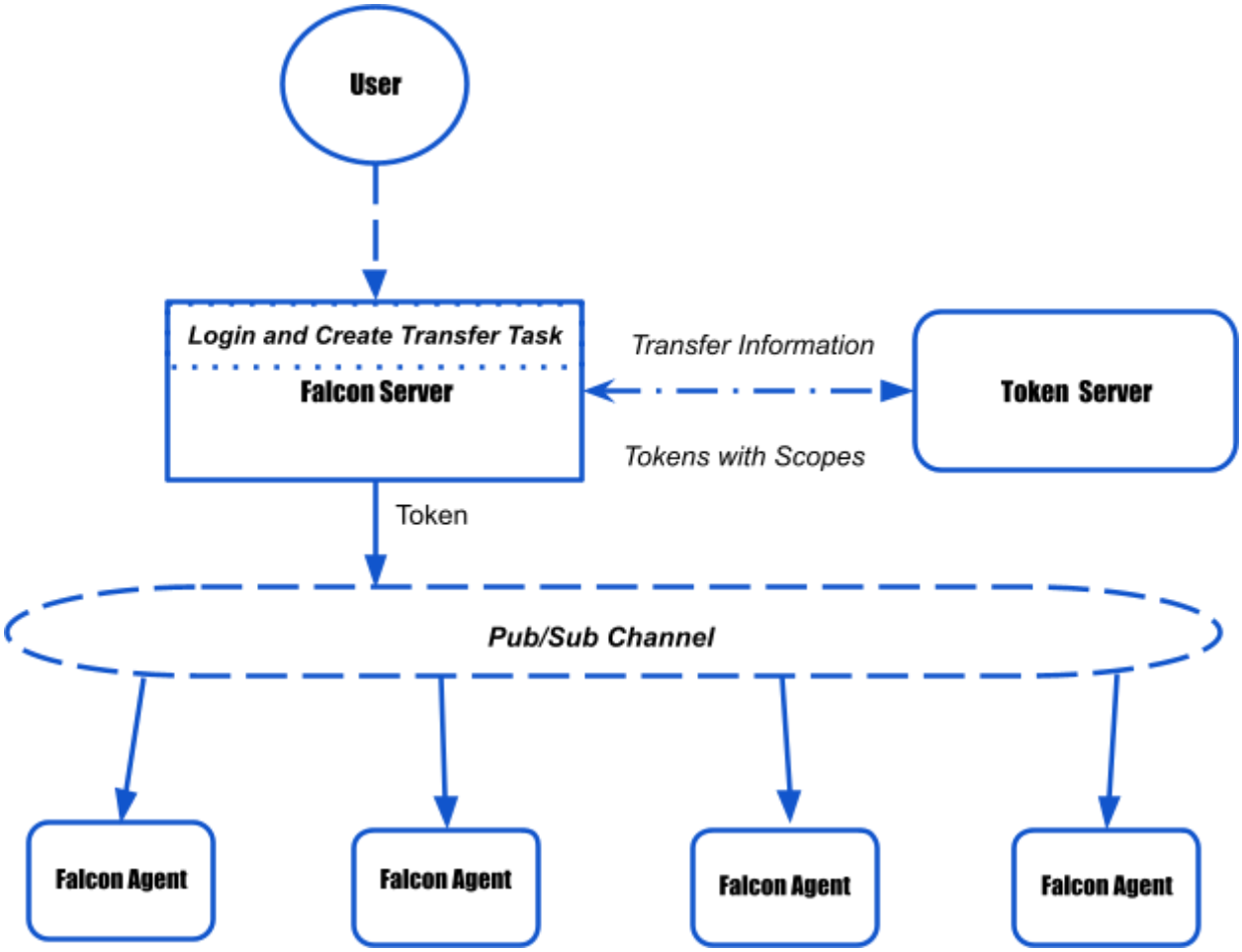
4. Issues separate tokens for source and destination
    a. Source Token: put source as the audience, fetch and append source scopes.
    b. Destination Token: put destination as the audience, fetch and append destination scopes.

5. Example:
    a. Serialized Token (Encryption Algorithm: *RS256*):

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJ1c2VyIjoiYXJpZiIsImVtYWlsIjoiYXJpZkBuZXZhZGEudW5yLmVkdSIsInNjb3BlIjoiY29uY3VycmVuY3k6L2FyaWYvMyBid19icHM6L2FyaWYvMTAwMDAwMDAwMDAwMCBkaXJlY3RfaW86L2FyaWYvMCByZWFkOi9kYXRhL2FyaWYvIiwiYXVkIjoiZHRuMS5jcy51bnIuZWR1IiwiaXNzIjoiaHR0cHM6Ly9ocGNuLnVuci5lZHUiLCJleHAiOjE2NDk4OTA1OTksImlhdCI6MTY0OTg4OTk5OSwibmJmIjoxNjQ5ODg5LCJqdGkiOiJhOTgwYjA0OS0yY2M2LTQ3NzQtOTdmMy0xNDY5ZjMxNjc3OTMifQ.KrDkTmYW0NaBeJtV0JmhkvFjGDU8tp2y_2SlzzxdpiQTBZ1gNPGVQVNWTAZww46SO2kptz1v2bQ0Y6uYmg8fs5kZsDZj1Wn83gKvGPoNEezkhAd73bDhQsCkrVtTLr1TPB08LGf4v7nbsJ19tXKgxuxM-TndrTsfCg3Hxtlx0M3Mr9-13Pw7zoDcro8cD145eOH1ebrNKhPG_094gmRfr6cEhdK9NcOinvTFps13nJLjZsy1bBMPORw94eZ1gr1CvaQLdBvWTDjYuO42zG5MLufXZihK3T0eQ1vFfFkylJHyOzwb9ycJKSKgdZNAa3LlxCPH5M2eTTAk5n6kmhHlcA

    b. Deserialized Token:


```
{
  "user": "arif",
  "email": "arif@nevada.unr.edu",
  "scope": "concurrency:/arif/3 bw_bps:/arif/1000000000
direct_io:/arif/0 read:/data/arif/",
  "aud": "dtn1.cs.unr.edu",
  "iss": "https://hpcn.unr.edu",
  "exp": 1649890599,
  "iat": 1649889999,
  "nbf": 1649889999,
  "jti": "a980b049-2cc6-4774-97f3-1469f3167793"
}
```



Falcon Server

1. Users will log in to the Falcon server via federated login.

2. The User can create transfer tasks by providing source and destination sites and their respective file directives.

3. Falcon-server provides user identity and the transfer source and destination hosts to the Scitoken server.

4. Falcon servers get two tokens from the token server respective to the source and destination hosts. Token has clearly defined scopes for respective source and destination.

5. Finally, the Server publishes tokens to the respective agents' communication channels via a message queue.

Falcon Agent

1. Each HPC site will have a falcon agent for performing the tasks of verifying tokens, parsing scopes, sending, and receiving files.

2. Agents will maintain communication with the Falcon server via predefined communication channels. (Implemented via Redis-stream)

3. Agents subscribed to their channel to receive new tokens. Then it verifies its authenticity via the issuer public key. Then parse claims/scopes from the token.

4. Then agents proceed to transfer data by maintaining the limit of the scope.

Conclusion

Capability-based JWT provided by the Scitokens library simplifies authorization and provides mechanisms for efficient resource control. Properly defined scopes make sure no specific user overuse the system, consequently, Falcon agents can fairly allocate resources among all users. Additionally, predefined rules for setting scope values make the token interoperable among many sites. Finally, in future studies, we aim to integrate additional scopes into the token to automate as many aspects of resource control as possible.