

Secrets Sharing Between MFE Artifacts

Executive Summary

Modern web applications are progressively adopting Micro Frontend (MFE) architectures to create scalable and maintainable systems. A significant challenge within this architecture is the secure sharing of authentication tokens and sensitive information among various MFEs while preserving user sessions. This document introduces a centralized secret management approach utilizing YAML configuration files and deployment pipelines to eliminate secret duplication, minimize security risks, and simplify maintenance across MFE artifacts.

Introduction

Micro Frontend Environments (MFEs) have emerged as the standard method for constructing extensive web applications, enabling teams to create, deploy, and manage separate frontend modules that integrate into a unified user experience. As companies embrace this architecture, ensuring consistent user authentication and session management throughout all MFEs becomes crucial. This requires the secure and efficient transfer of sensitive data, such as API keys, authentication tokens, and configuration secrets, among various frontend components.

Problem Definition

Current MFE implementations face several critical challenges in secret management:

- **Secret Duplication:** Storing identical secrets across multiple MFE repositories creates maintenance overhead and increases security risks
- **Synchronization Issues:** Manual secret updates across multiple artifacts lead to version mismatches and potential service disruptions
- **Security Vulnerabilities:** Multiple storage locations increase the attack surface and complicate access control
- **Operational Complexity:** Managing secret rotation and updates becomes exponentially more difficult as the number of MFEs grows
- **Compliance Risks:** Inconsistent secret handling across artifacts may violate security policies and regulatory requirements

Solution

The proposed solution implements a centralized secret sharing mechanism using YAML configuration files and automated deployment pipelines. This approach establishes a parent-child relationship between artifacts, allowing secrets to be defined once and shared programmatically.

Implementation Steps

Step 1: Create Secret Configuration File. In the parent artifact repository, create a secret-`<env>`.yaml file for each target environment:

secrets:

pipeline_relationships:

- type: static

children:

- deployment_guid: <child_mfe_deployment_guid>

description: <descriptive_name>

secret: <secret_name>

access:

- private-key

Step 2: Configure Deployment Manifest: Add the following task to your deploy.yml file:

```
jobs:
  deploy:
    name: Secrets SPI
    if: ${{ hashFiles(steps.manifest-names.outputs.SECRETS_MANIFEST_FILENAME) !=
" }}"
    uses: actions
    with:
      name: secrets
      forge-file: ${{ steps.manifest-names.outputs.SECRETS_MANIFEST_FILENAME }}
      env: ${{ inputs.environment }}
```

Step 3: After adding this task, create a new branch from the development branch, commit, and push your changes to the parent artifact repository.

Step 4: Create a Pull Request to development/main/master branches after approval.

Step 5: Deploy the updates first into the non-production environment

Step 6: If the deployments succeeded, you could proceed further with production deployments

Validation

1. **Parent Artifact Validation:** Navigate to the Secrets tab in the parent artifact dashboard to confirm child artifacts and their respective environments are listed for shared secrets

2. **Child Artifact Validation:** Check the Secrets tab in each child artifact to verify that shared secrets from the parent are visible and accessible
3. **Functional Testing:** Conduct end-to-end testing to ensure MFEs can successfully authenticate and maintain user sessions using shared secrets

Benefits

Operational Advantages

- **Centralized Management:** A Single source of truth for secret definitions reduces complexity
- **Automated Synchronization:** Pipeline-based distribution ensures consistency across all MFEs
- **Reduced Maintenance:** Simplified secret rotation affects only the parent artifact
- **Enhanced Security:** Fewer storage locations minimize potential security vulnerabilities
- **Improved Compliance:** Standardized secret handling supports regulatory requirements

Conclusion

The centralized secret sharing solution addresses critical security and operational challenges in MFE architectures. By establishing parent-child relationships through YAML configuration and automated pipelines, organizations can maintain secure, consistent secret management while reducing operational overhead. This approach scales effectively with growing MFE ecosystems and provides a foundation for robust, maintainable applications.

Call to Action

1. **Immediate Steps:** Review current MFE secret management practices and identify duplication issues
2. **Implementation Planning:** Begin pilot implementation with non-critical environments
3. **Team Training:** Educate development teams on the new secret sharing methodology
4. **Monitoring Setup:** Establish monitoring and alerting for secret sharing pipeline health
5. **Security Review:** Conduct a security assessment of the centralized approach with your information security team

Author's Statement

This white paper is based primarily on the author's direct professional experience in enterprise software development and solution delivery across the Aerospace, Banking, and Healthcare industries. The insights, methods, and recommendations presented here are drawn from real-

world project implementations, production challenges, and hands-on technical leadership—rather than from secondary or purely academic sources.

The purpose of this paper is to share applied knowledge, best practices, and proven approaches that have evolved through years of delivering large-scale, mission-critical systems. These insights aim to help technology leaders, architects, and developers address similar technical and operational challenges in their own environments.