SecureVault Setup and Installation

Document Number: PROD-SV-001 **Product:** SecureVault **Version:** 1.x **Last Updated:** November 1, 2023 **Owner:** SecureVault Product Team

Introduction

SecureVault is NovaTech's enterprise secrets management solution. It provides secure storage, access control, and rotation for sensitive data like API keys, database credentials, certificates, and encryption keys.

What is SecureVault?

SecureVault provides: - **Secure storage:** AES-256 encryption at rest, TLS in transit - **Access control:** Fine-grained permissions and policies - **Secret rotation:** Automated credential rotation - **Audit logging:** Complete audit trail of all access - **Dynamic secrets:** Generate credentials on-demand - **Integration:** Native support for CloudForge, DevPipeline, and popular tools

Deployment Options

SecureVault Cloud (Recommended)

Fully managed service hosted by NovaTech: - No infrastructure to manage - Automatic updates and backups - SLA-backed availability - Start at app.securevault.io

SecureVault Self-Hosted

Deploy in your own infrastructure: - Full control over data location - Air-gapped environment support - Available for enterprise customers - Contact sales@novatech.com for licensing

Getting Started with SecureVault Cloud

Step 1: Create an Organization

1. Go to app.securevault.io
2. Sign in with your NovaTech account
3. Click **"Create Organization"**
4. Enter organization name
5. Select your primary region

Step 2: Install the CLI

macOS:

```
brew tap novatech/securevault
brew install securevault
```

Linux:

```
curl -sSL https://get.securevault.io | bash
```

Windows:

```
iwr https://get.securevault.io/win | iex
```

Step 3: Authenticate

```
sv login
```

This opens your browser to authenticate. After authentication:

```
sv status
# Output: Authenticated as user@novatech.com (org: my-org)
```

Step 4: Create Your First Secret

```
sv secret set production/database/url "postgres://user:pass@host:5432/db"
```

Step 5: Retrieve a Secret

```
sv secret get production/database/url
# Output: postgres://user:pass@host:5432/db
```

Core Concepts

Secrets

A **secret** is any sensitive data: - API keys - Database credentials - TLS certificates - SSH keys - Encryption keys - Tokens

Secrets are stored at **paths** (e.g., `production/database/password`).

Secret Engines

Secret engines define how secrets are stored and managed:

Engine	Description
`kv`	Key-value storage (default)
`database`	Dynamic database credentials
`aws`	Dynamic AWS credentials
`pki`	X.509 certificates
`transit`	Encryption as a service

Policies

Policies define who can access what:

```
# Example policy
path "production/*" {
  capabilities = ["read"]
}

path "production/database/*" {
  capabilities = ["read", "create", "update"]
}
```

Authentication Methods

Auth methods define how users and applications authenticate:

Method	Use Case
`novatech`	NovaTech SSO users
`token`	Service accounts, scripts
`kubernetes`	Kubernetes pods
`aws`	AWS IAM roles
`cloudforge`	CloudForge services

Organizing Secrets

Recommended Path Structure

```
{environment}/{application}/{secret-type}
```

```
Examples:
production/webapp/database-url
staging/api/stripe-key
shared/certificates/wildcard
```

Environment Separation

Organize by environment:

```
production/
  webapp/
     database-url
     api-key
  worker/
     queue-credentials
  shared/
      encryption-key

staging/
  webapp/
     database-url
...
```

CLI Reference

Authentication

```
# Login interactively
sv login

# Login with token (for scripts)
sv login --token $SECUREVAULT_TOKEN

# Check authentication status
sv status

# Logout
sv logout
```

Managing Secrets

```
# Create/update a secret
sv secret set path/to/secret "value"
```

```
# Create from file
sv secret set path/to/cert @certificate.pem

# Get a secret
sv secret get path/to/secret

# Get as JSON
sv secret get path/to/secret --format json

# List secrets at path
sv secret list production/

# Delete a secret
sv secret delete path/to/secret
```

Secret Metadata

```
# Get secret metadata (without value)
sv secret metadata path/to/secret

# Set metadata
sv secret set path/to/secret "value" \
  --metadata owner=team-platform \
  --metadata rotate=90d
```

Version History

```
# List versions
sv secret versions path/to/secret

# Get specific version
sv secret get path/to/secret --version 3

# Rollback to version
sv secret rollback path/to/secret --version 2
```

Application Integration

Environment Variables

```
# Export secret to environment variable
export DATABASE_URL=$(sv secret get production/database/url)
```

CloudForge Integration

```yaml
# cloudforge.yaml
services:
  web:
    secrets:
      - path: production/webapp
        env_prefix: ""
```

CloudForge automatically injects secrets as environment variables.

DevPipeline Integration

```yaml
# .devpipeline.yaml
steps:
  - name: Fetch secrets
    uses: securevault/fetch@v1
    with:
      secrets: |
        production/database/url => DATABASE_URL
        production/api/key => API_KEY
```

Kubernetes Integration

```yaml
# SecretProviderClass
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: vault-secrets
spec:
  provider: securevault
  parameters:
    vaultAddress: "https://vault.securevault.io"
    objects: |
      - objectName: "production/webapp/database-url"
        secretKey: "DATABASE_URL"
```

SDK Usage

Node.js:

```javascript
const SecureVault = require('@novatech/securevault');
const client = new SecureVault({
  address: process.env.SECUREVAULT_ADDR,
  token: process.env.SECUREVAULT_TOKEN
});

const secret = await client.read('production/database/url');
console.log(secret.data.value);
```

Python:

```
import securevault

client = securevault.Client()
secret = client.secrets.get("production/database/url")
print(secret.value)
```

Security Best Practices

Access Control

- Use least-privilege policies
- Separate environments (dev, staging, prod)
- Regularly audit access

Secret Hygiene

- Rotate secrets regularly
- Never log secret values
- Don't commit secrets to code

Authentication

- Use app-specific auth where possible (CloudForge, Kubernetes)
- Rotate tokens periodically
- Use short-lived tokens when possible

Troubleshooting

"Permission denied"

- Check your policy grants the required capability
- Verify you're authenticated (`sv status`)
- Ensure you're accessing the correct path

“Secret not found”

- Check the path is correct (case-sensitive)
- Verify the secret exists (`sv secret list`)
- Check if secret was deleted

“Connection refused”

- Check network connectivity
- Verify SECUREVAULT_ADDR is correct
- Check firewall rules

Getting Help

- **Documentation:** docs.securevault.io
- **Support:** support@securevault.io
- **Internal NovaTech:** #securevault-help on Slack

Next Steps

1. Access Policies - Configure fine-grained access
2. Secret Rotation - Automate credential rotation
3. Audit Logging - Monitor access
4. Dynamic Secrets - Generate credentials on-demand
5. Disaster Recovery - Backup and recovery

Related Documents: Access Policies (PROD-SV-010), CloudForge Integration (PROD-CF-025), DevPipeline Integration (PROD-DP-045)