



Keeping Secrets Secret with Conjur

OWASP Cincinnati
June 2022

Who am I?

Shlomo Zalman Heigh

Senior Software Engineer, CyberArk



Developer

Conjur

- Open source projects
- Community engagement
- Long held interest in security

Cincinnatian

Go Cincy!

- Regular OWASP attendee

Hobbies

- 3D printing (see my prints on [Thingiverse!](#))
- Woodworking
- Small electronics
- RC planes

Social

Find me on...

- LinkedIn: linkedin.com/in/szheigh
- GitHub: github.com/szh





Story Time!

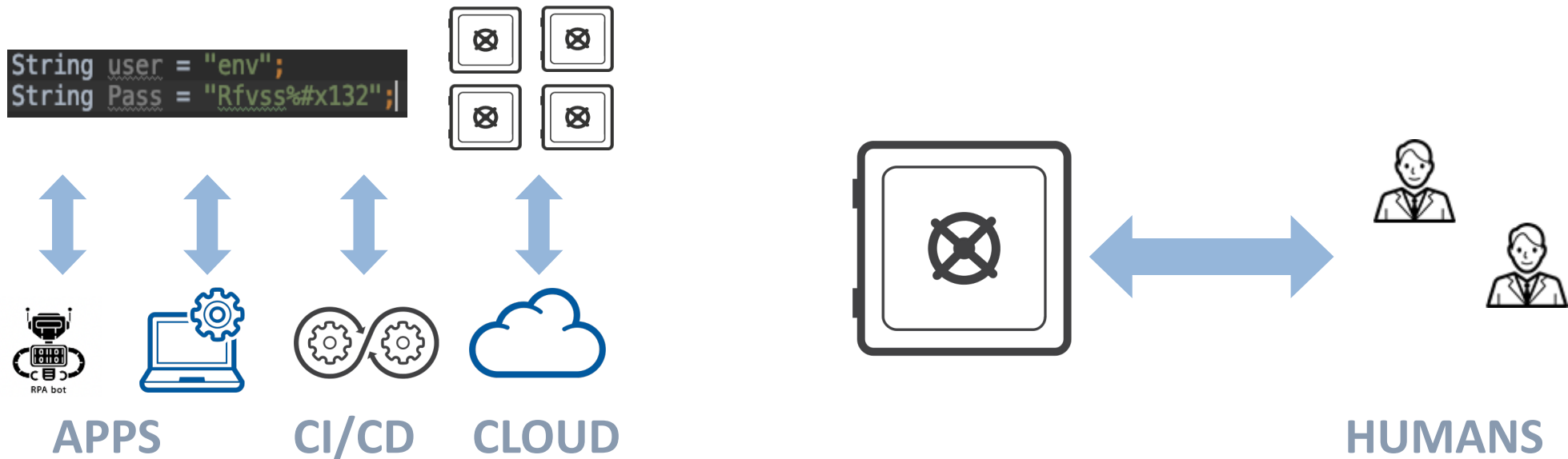


1

The Problem with Secrets



Organizations today



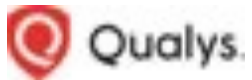
No Centralized Enterprise Secrets Management

- ✗ No rotation or security governance
- ✗ No audit or logs
- ✗ No centralized control
- ✗ No visibility to security team
- ✗ Secrets stored in multiple vaults (some not secure)
- ✗ Hard coded credentials in code or config files



Common application platforms

RPA / COTS



Homegrown Apps



CI/CD Automation



Cloud native apps



Cloud workloads

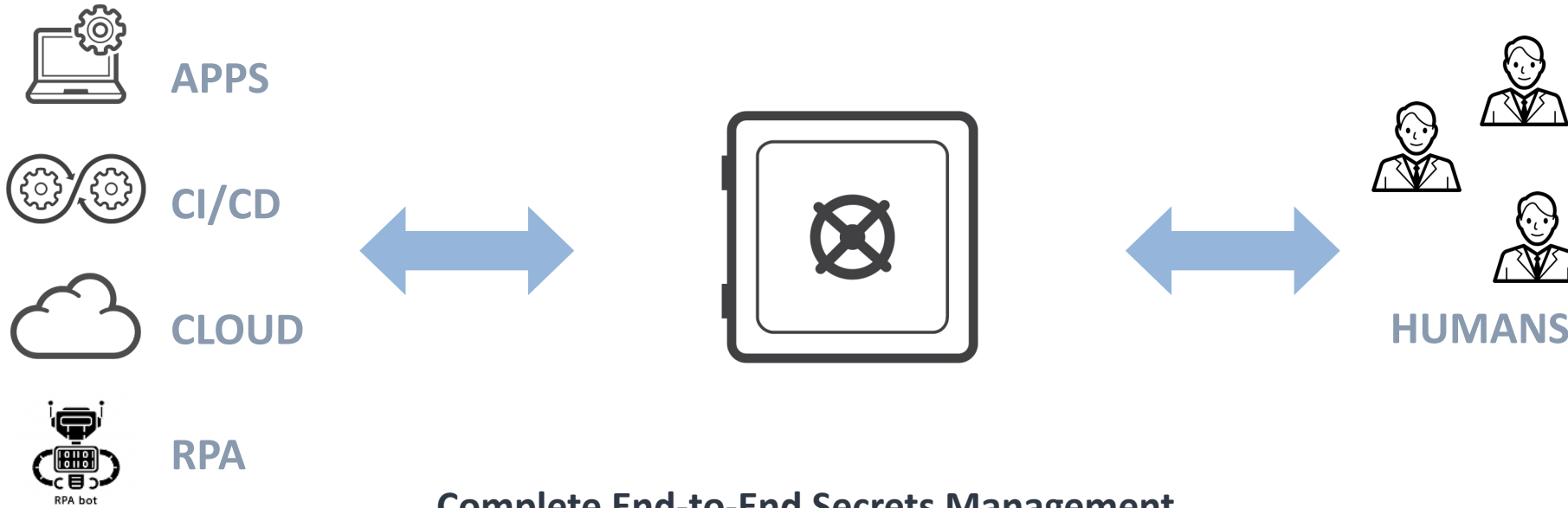


2

The Solution



What is secrets management?



Complete End-to-End Secrets Management

- ✓ One centralized source of truth serves both humans and apps
- ✓ Secures all application types, everywhere
- ✓ Strong authentication and authorization – apply least privilege
- ✓ Automated secrets rotation
- ✓ Fully audited and controlled by security team
- ✓ No hard-coded credentials



SECRET DELIVERY/CONSUMPTION OPTIONS

Ease of use



APIs

Use REST API or SDKs to retrieve secrets

Key Advantages:

Available SDKs for Java, Ruby, Go and more.

Supports rotations

Other Considerations:

Requires code change in the application



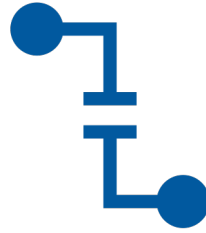
C# SDK Example

```
Conjur.Client client = new Conjur.Client("https://localhost:8443", "myConjurAccount");  
client.Credential = new System.Net.NetworkCredential("host/demo-aspnet-app/demo-aspnet-app", "83w54e257aw0c  
string secret = client.Variable("demo-aspnet-app/mysecret").GetValue();
```



SECRET DELIVERY/CONSUMPTION OPTIONS

Ease of use



Summon

Fetches secrets and makes them available to the application as environment variables

Key Advantages:

No code change required

Other Considerations:

Rotations are not supported – requires a pod restart when password changes

Deployments requires more steps



Summon Example

```
$ brew tap cyberark/tools  
$ brew install summon  
$ brew install summon-conjur
```

```
$ conjur variable values add "my_vault/my_secret" "mySuperSecretValue"
```

```
$ cat > secrets.yml <<EOF  
MY_SECRET: !var my_vault/my_secret  
EOF
```

```
$ summon /bin/bash -ec 'echo $MY_SECRET'  
mySuperSecretValue
```



SECRET DELIVERY/CONSUMPTION OPTIONS

Ease of use



Secretless Broker

Brokers the connection to the target resource

Key Advantages:

No Secrets delivered to the application

No code changes required

Supports rotations

Other Considerations:

Requires a service connector to the target (select from list of available connectors)



Secretless Broker Example

```
$ docker container run --rm -p 5432:5432 -p 5454:5454 cyberark/secretless-broker-quickstart
```

```
$ psql "host=localhost port=5432 user=secretless dbname=quickstart sslmode=disable" \  
-c 'select * from counties;'
```

Password for user secretless:

psql: FATAL: password authentication failed for user "secretless"

```
$ psql "host=localhost port=5454 user=secretless dbname=quickstart sslmode=disable" \  
-c 'select * from counties;'
```

id	name
1	Middlesex
2	Worcester
3	Essex
4	Suffolk
5	Norfolk



SECRET DELIVERY/CONSUMPTION OPTIONS

Ease of use



Secrets Provider

(Kubernetes Secrets or File)

Uses init or sidecar container to fetch secrets and push them into Kubernetes Secrets or a shared volume

Key Advantages:

Easier deployment using HELM

Native experience for developers that already use Kubernetes Secrets

Small footprint

Other Considerations:

Uses Kubernetes RBAC and Audit functions

Secrets can be stored externally to the pod in Kubernetes Secrets.



Kubernetes Manifest Example

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test-app-secrets-provider-rotation
  template:
    metadata:
      labels:
        app: test-app-secrets-provider-rotation
      annotations:
        conjur.org/container-mode: "sidecar"
        conjur.org/secrets-refresh-enabled: "true"
        conjur.org/secrets-refresh-interval: "10s"
        conjur.org/authn-identity: "myLogin"
        conjur.org/secrets-destination: "file"
        conjur.org/conjur-secrets.rotation-app: |
          - test-secrets-provider-rotation-app-db/url
          - test-secrets-provider-rotation-app-db/username
          - test-secrets-provider-rotation-app-db/password
        conjur.org/secret-file-path.rotation-app: "./application.yaml"
        conjur.org/secret-file-format.rotation-app: template
        conjur.org/secret-file-template.rotation-app: |
          spring:
            datasource:
              platform: postgres
              url: jdbc:{{ printf `{{ secret "url" }}` }}
              username: {{ printf `{{ secret "username" }}` }}
              password: {{ printf `{{ secret "password" }}` }}
            jpa:
              generate-ddl: true
              hibernate:
                ddl-auto: update
```


SECRET DELIVERY/CONSUMPTION OPTIONS

Ease of use



Secretless Broker

Brokers the connection to the target resource

Key Advantages:

- No Secrets delivered to the application
- No code changes required
- Supports rotations

Other Considerations:

Requires a service connector to the target (select from list of available connectors)



APIs

Uses API calls to retrieve secrets

Key Advantages:

- Available APIs for Java, Ruby, Go.
- Supports rotations

Other Considerations:

Requires code change in the application



Summon

Fetches secrets and makes them available to the application as environment variables

Key Advantages:

- No code change required

Other Considerations:

- Rotations are not supported – requires a pod restart when password changes
- Deployments requires more steps



Secrets Provider (Kubernetes Secrets or File)

Uses init or sidecar container to fetch secrets and push them into Kubernetes Secrets or a shared volume

Key Advantages:

- Easier deployment using HELM
- Native experience for developers that already use Kubernetes Secrets
- Small footprint

Other Considerations:

- Uses Kubernetes RBAC and Audit functions
- Secrets can be stored externally to the pod in Kubernetes Secrets.



Summary



Why not keep secrets in code?

- Hard to manage
 - Can't easily rotate
 - No audit trail
- Violates "least privilege"
- Developers shouldn't have access to production credentials

Why Conjur?

- Open source
 - Can be self hosted near the application (in K8s cluster)
- Enterprise ready
 - Scalable, can sync with CyberArk Vault
- Easy integrations
 - Native K8s integrations, REST API, SDKs
- Trusted
 - Used by Fortune 500 companies
 - Maintained by a trusted security company



QUESTIONS



Resources

- Blog: Remove Secrets from your Codebase
<https://www.conjur.org/blog/remove-secrets-from-your-codebase/>
- Conjur OSS Quickstart
<https://www.conjur.org/get-started/quick-start/oss-environment/>
- CyberArk Commons Community (Discourse)
<https://discuss.cyberarkcommons.org/>
- OWASP Secrets Management Cheat Sheet
https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_CheatSheet.html
- OWASP WrongSecrets
<https://owasp.org/www-project-wrongsecrets/>
- Secure Deployment: 10 Pointers on Secrets Management
<https://dev.to/commjoen/secure-deployment-10-pointers-on-secrets-management-187j>



Thank You

