Creating Secure Software

Benefits from Cloud Thinking

Daniel Sawano

Kraków, 9-11 May 2018

Daniel Sawano

AVANZA

Security benefits from cloud thinking?

Cloud concepts

Twelve-factor app

- **Codebase**
  One codebase tracked in revision control, many deploys
- **Dependencies**
  Explicitly declare and isolate dependencies
- **Config**
  Store configuration in the environment
- **Backing services**
  Treat backing services as attached resources
- **Build, release, run**
  Strictly separate build and run stages
- **Processes**
  Execute the app as one or more stateless processes
- **Port binding**
  Export services via port binding
- **Concurrency**
  Scale out via the process model
- **Disposability**
  Maximize robustness with fast startup and graceful shutdown
- **Dev/prod parity**
  Keep development, staging, and production as similar as possible
- **Logs**
  Treat logs as event streams
- **Admin processes**
  Run admin/management tasks as one-off processes

https://12factor.net

Cloud-native

*A **cloud-native** application is an application that has been **designed** and implemented to run on a **Platform-as-a-Service** installation and to embrace **horizontal elastic scaling**.*

Kevin Hoffman, Beyond the Twelve-Factor App

What we’ll cover today

- Configuration
- Separate processes
- Logging
- The three R’s of enterprise security

Configuration

“Store configuration in the environment”

Configuration

Configuration in code

Configuration

Configuration in code

```
public class DatabaseConnection {

    private static final int PORT_NUMBER = 1521;
    private static final Duration CONNECTION_TIMEOUT = ofSeconds(5);

    // ...
}
```

Configuration

Configuration in code

```
public class DatabaseConnection {

    private static final int PORT_NUMBER = 1521;
    private static final Duration CONNECTION_TIMEOUT = ofSeconds(5);
    private static final String USERNAME = "service-A";
    private static final String PASSWORD = "yC6@SX50";

    // ...
}
```

Configuration

Configuration in code — challenges

- Anyone with access to the code can read the secrets
- No audit trail

Configuration

Configuration in resource files

```
environments:
    dev:
      database:
          port: 1521
          connection-timeout: 5000
          username: dev-service-A
          password: spring2019
    prod:
      database:
          port: 1521
          connection-timeout: 1000
          username: service-A
          password: yC6@SX5O
```

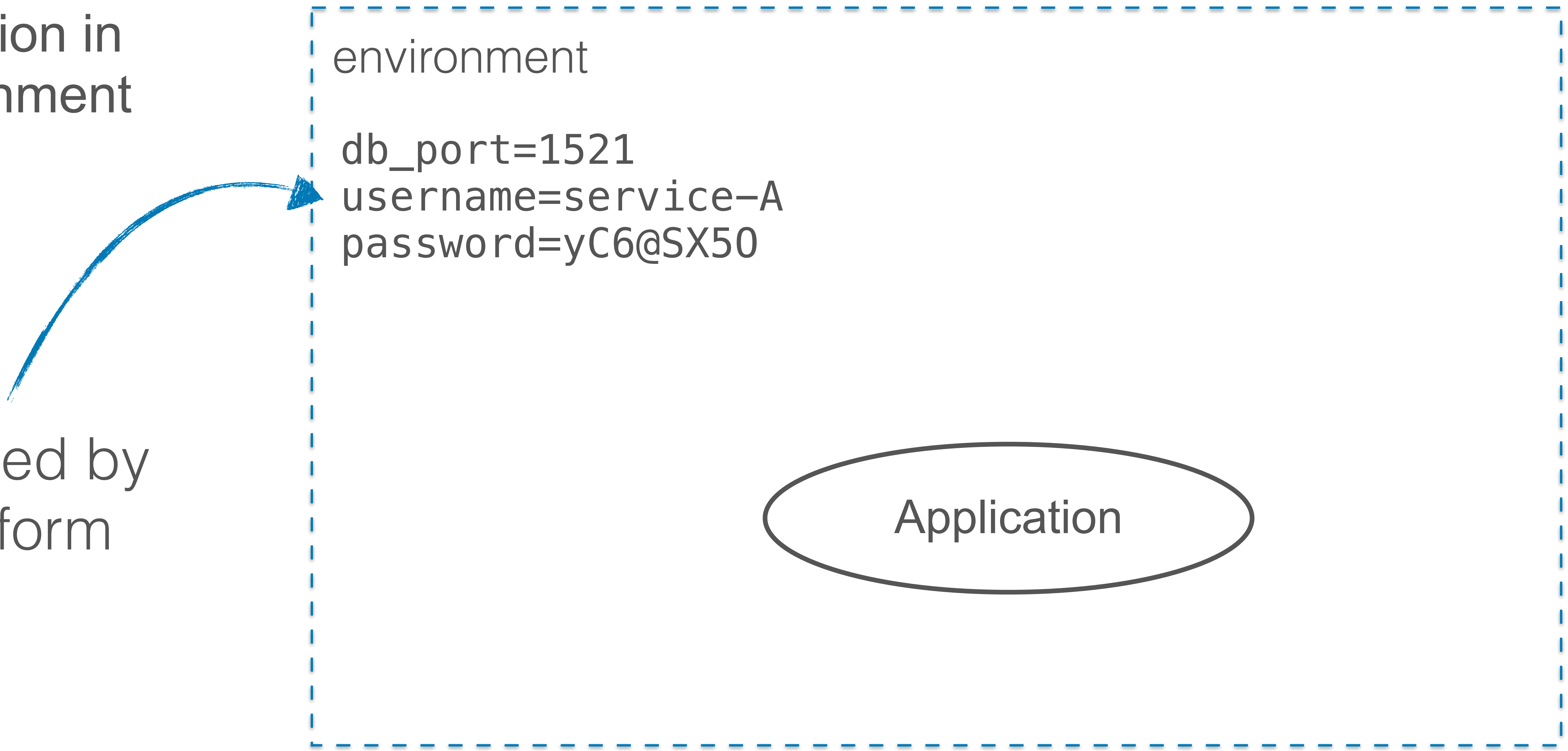
Configuration

Configuration in resource files — challenges

- Anyone with access to the conf can read the secrets
- No, or very limited, audit trail
- Encrypting values creates new problems

Configuration

Configuration in the environment



```
db_port=1521
username=service-A
password=yC6@SX50
```

AVANZA

Configuration

Configuration in the environment - solved security challenges

- Audit trail
  Responsibility put on the platform. Some aspects can be solved with IAM.

- Sharing secrets
  Minimized. Only managed by platform admins.

- Encryption
  Not completely solved. Can be solved with ephemeral secrets.

What we’ll cover today

- ✓ Configuration
- Separate processes
- Logging
- The three R’s of enterprise security

Separate processes

Run apps as separate stateless processes

Separate processes

- Run the app as multiple **stateless processes**
- **Separate** the **deployment** and **running** of the application
- Only **communicate via backing services**

Separate processes

Run the app as multiple stateless processes

- Security benefit: increased **availability** and **integrity**

CIA

- **Confidentiality** — data must only be disclosed to authorized users
- **Integrity** — data modification is only allowed in an authorized manner
- **Availability** — data must be available when needed

Separate processes

Run the app as multiple stateless processes

- Security benefit: increased **availability** and **integrity**

Separate processes

Separate the deployment and running of the application

- Security benefit: **principle of least privilege**

Separate processes

Only communicate via backing services

- Security benefit: improves **availability** and **integrity** by allowing apps to be **stateless**

What we'll cover today

- ✓ Configuration
- ✓ Separate processes
- Logging
- The three R's of enterprise security

Logging

Use logging as a service

Logging

Logging to disk - challenges

- Confidentiality
  - May contain sensitive information
  - Hard to control access
  - Hard to get a good audit trail
  - Hard prevent illegal access

Logging

Logging to disk - challenges

- Integrity
  - Maintaining integrity often overlooked
  - Write access to log files usually not restricted or audited

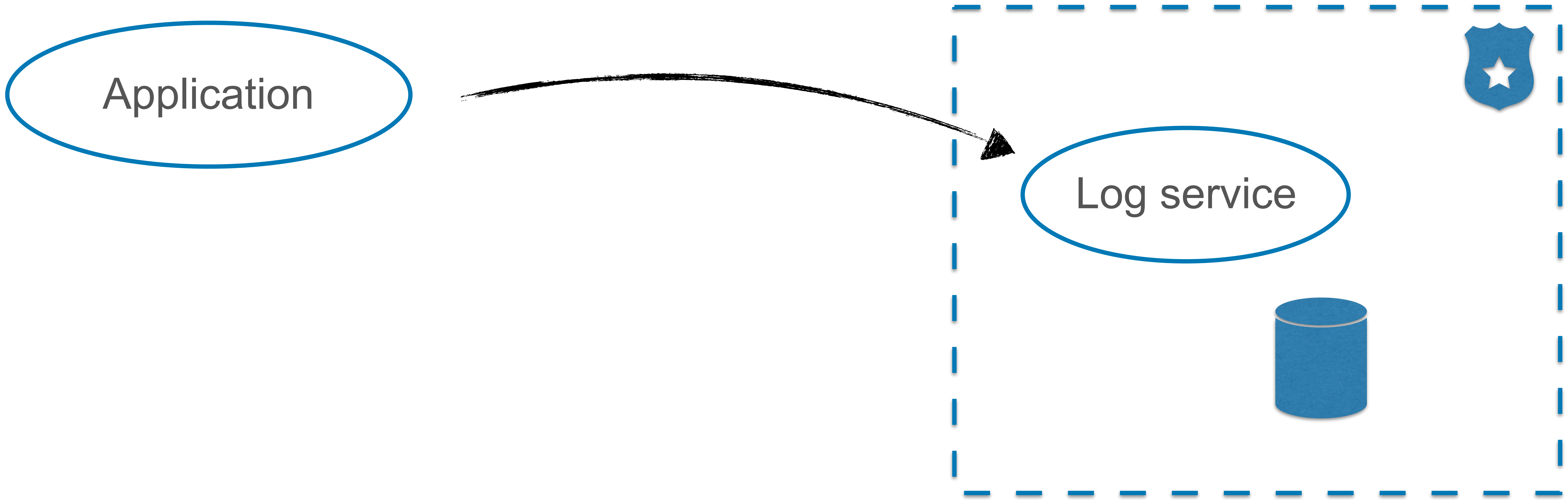
Logging

Logging to disk - challenges

- Availability
  - Log files are lost when servers are replaced
  - Disk space runs out

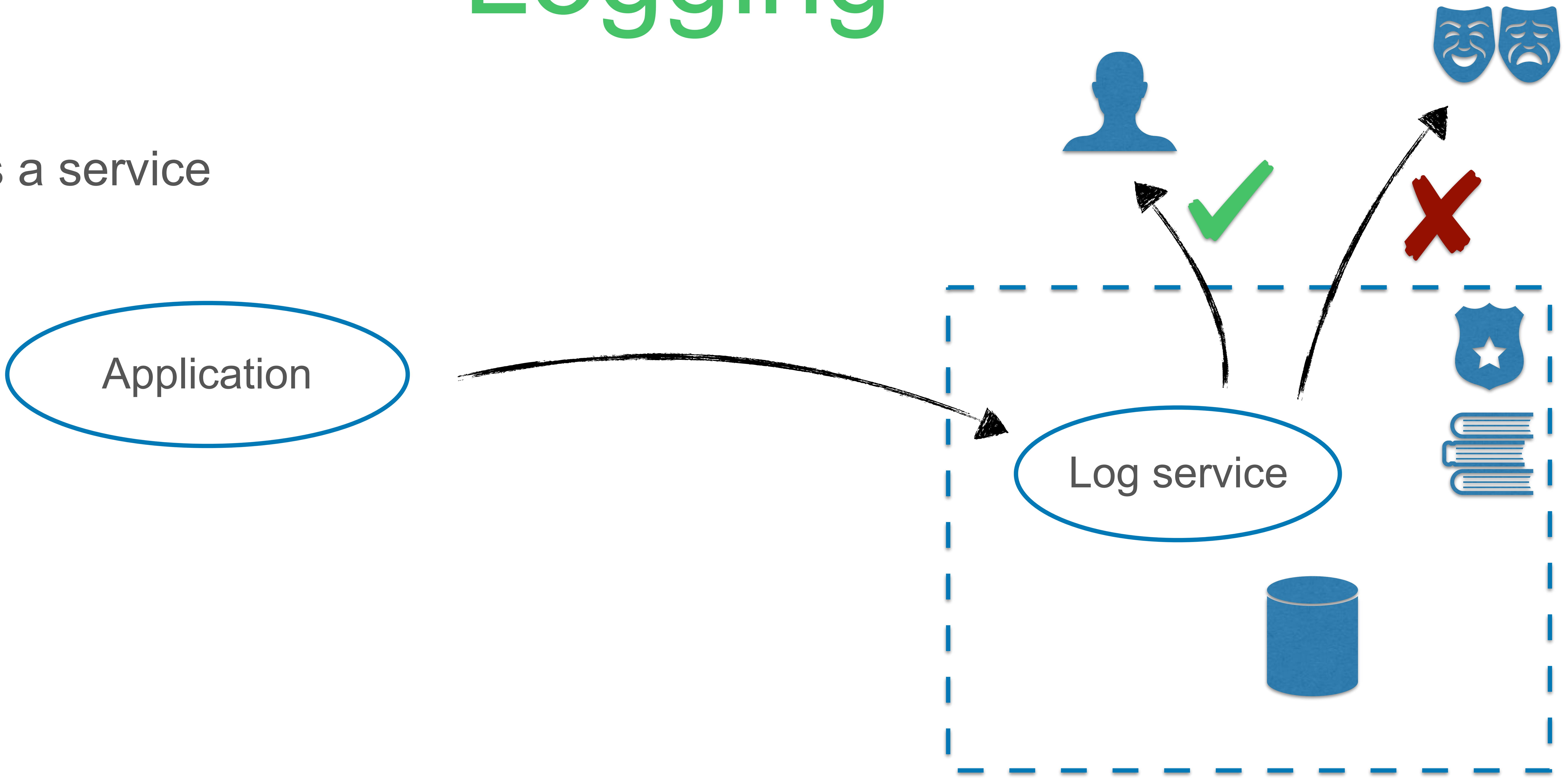
Logging

Logging as a service



Logging

Logging as a service



Logging

Logging as a service - solved security challenges

- Confidentiality
  Easy to restrict access and prevent illegal access.
  Audit trail.

- Integrity
  Mutating operations not exposed/implemented.
  Can even digitally sign log events

- Availability
  Log storage is handled explicitly so no log files can go missing
  Storage is a primary concern so no accidental shortage of disk space.

What we'll cover today

- ✓ Configuration
- ✓ Separate processes
- ✓ Logging
- The three R's of enterprise security

The three R’s

The three R’s of enterprise security

Justin Smith, 2016

The three R’s

The three R’s of enterprise security

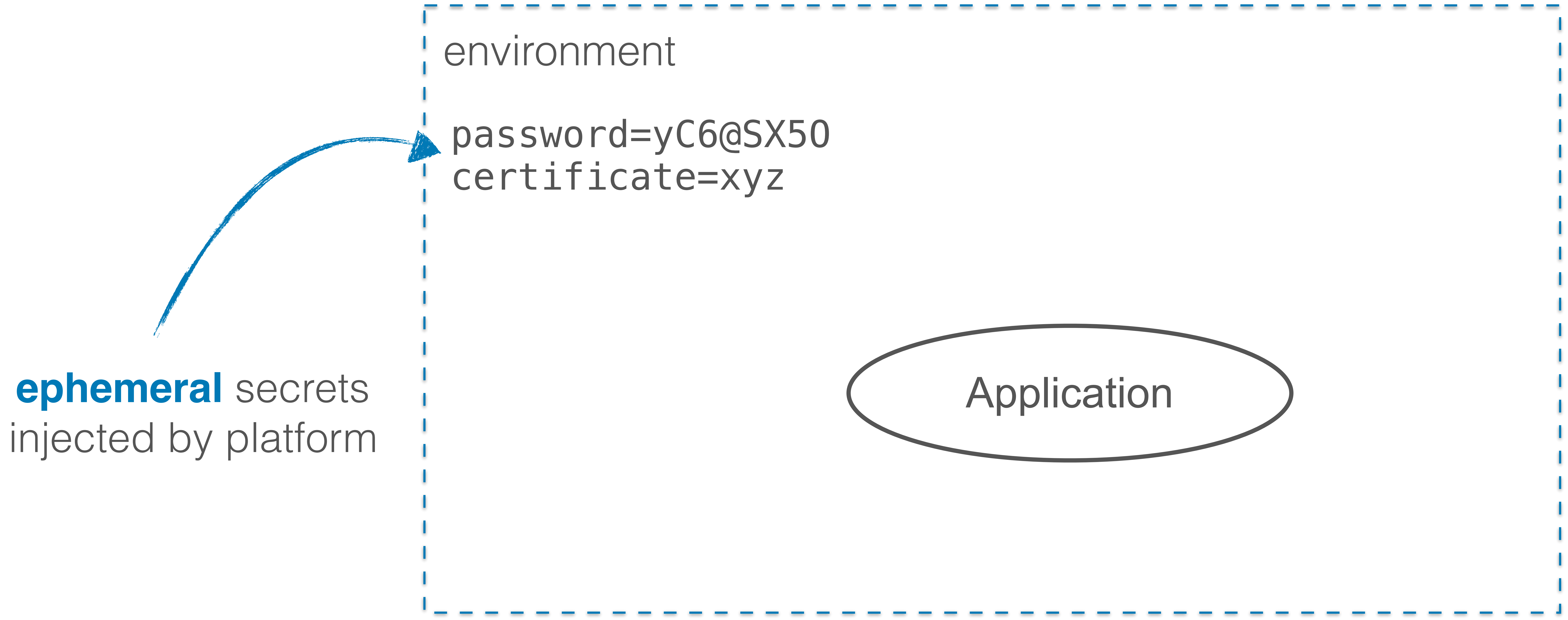
- Rotate
  Rotate secrets every few minutes or hours
- Repave
  Repave servers and applications every few hours
- Repair
  Repair vulnerable software a few hours after patch is available

The three R’s

Increase change to reduce risk

The three R’s

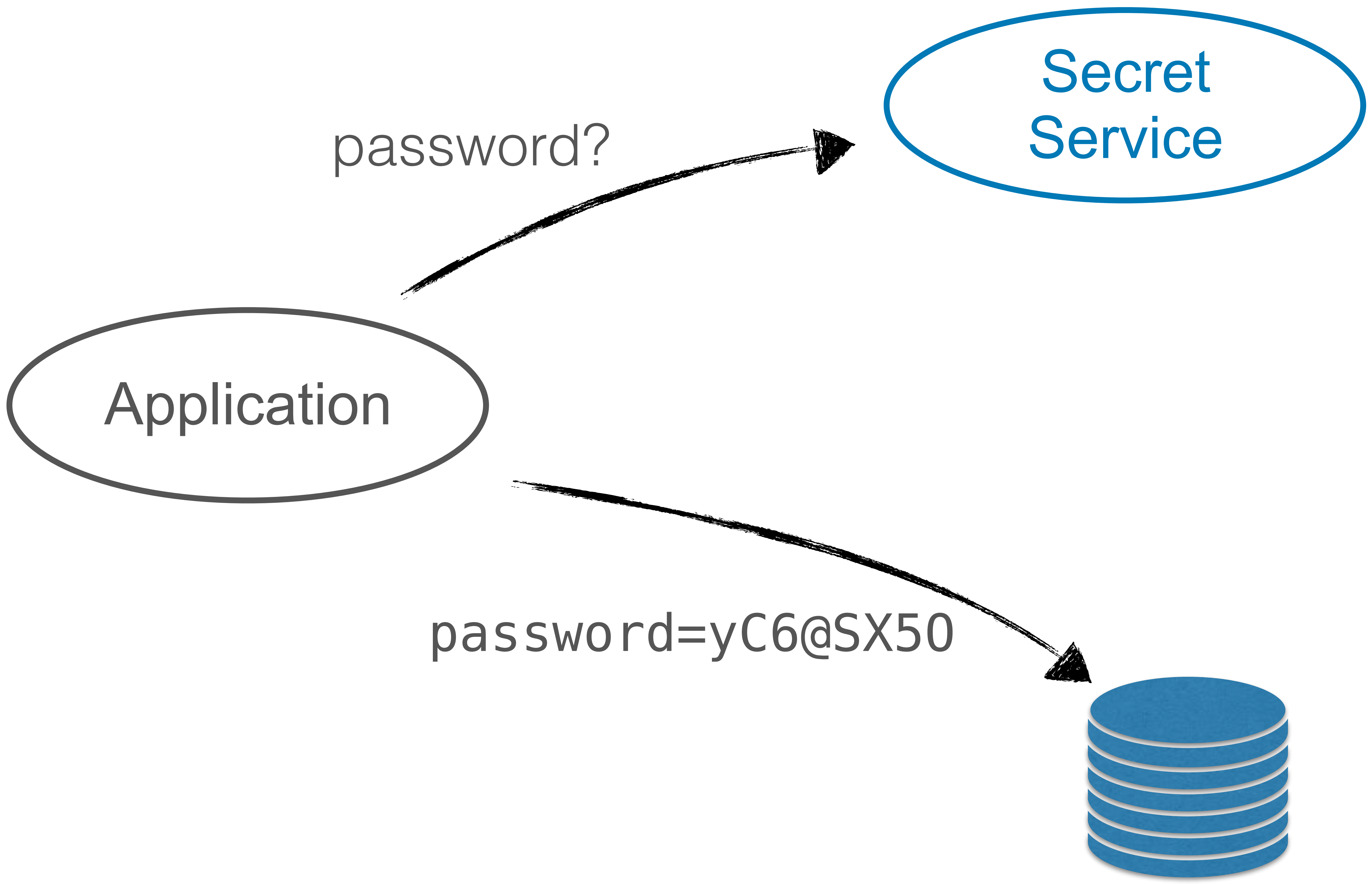
Rotate secrets every few minutes or hours



AVANZA

The three R's

Rotate secrets every few minutes or hours



AVANZA

The three R's

Rotate secrets every few minutes or hours

- Passwords
- Certificates
- Access tokens
- …

The three R’s

Repave servers and applications every few hours

- Recreate servers and apps from a know good state
- Use rolling deployments to eliminate downtime
- Burn old instances to the ground
- If running containers, consider also repaving the host

The three R’s

Repair vulnerable software a few hours after patch is available

- Applies to both operating systems and applications
- No incremental updates, repave instead

The three R’s

Repair vulnerable software a few hours after patch is available

The three R's

Repair vulnerable software a few hours after patch is available

Patch available → New known good state → Repave

The three R's

Repair vulnerable software a few hours after patch is available

- Applies to both operating systems and your own applications
- No incremental updates, repave instead
- CI/CD enables you to repair your own applications
- Don't forget 3rd party dependencies

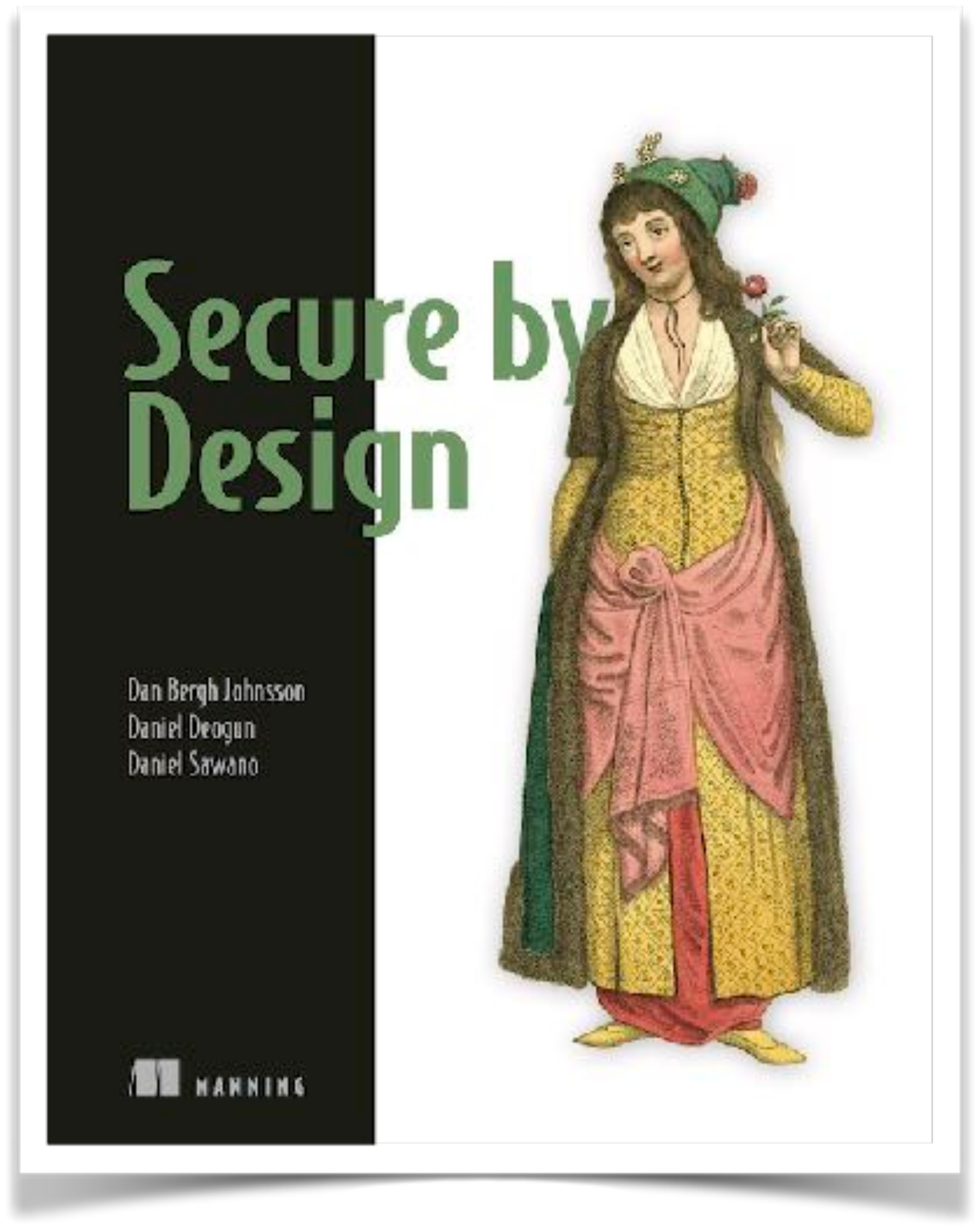
The three R’s

Ever-changing software is the nemesis of persistent threats

Summary

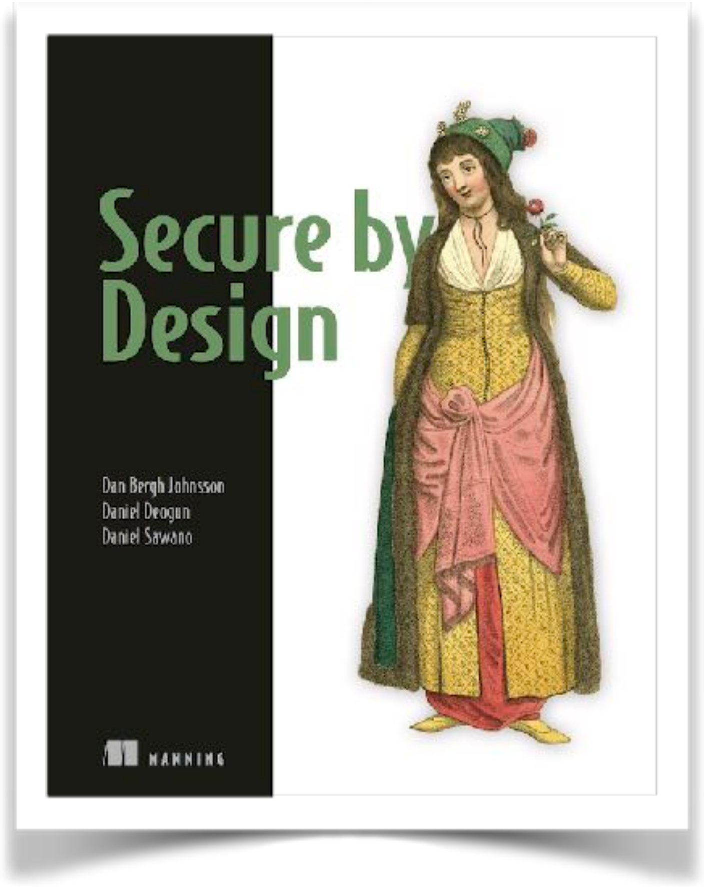
- ✓ Configuration
- ✓ Separate processes
- ✓ Logging
- ✓ The three R’s of enterprise security

Manning Publication



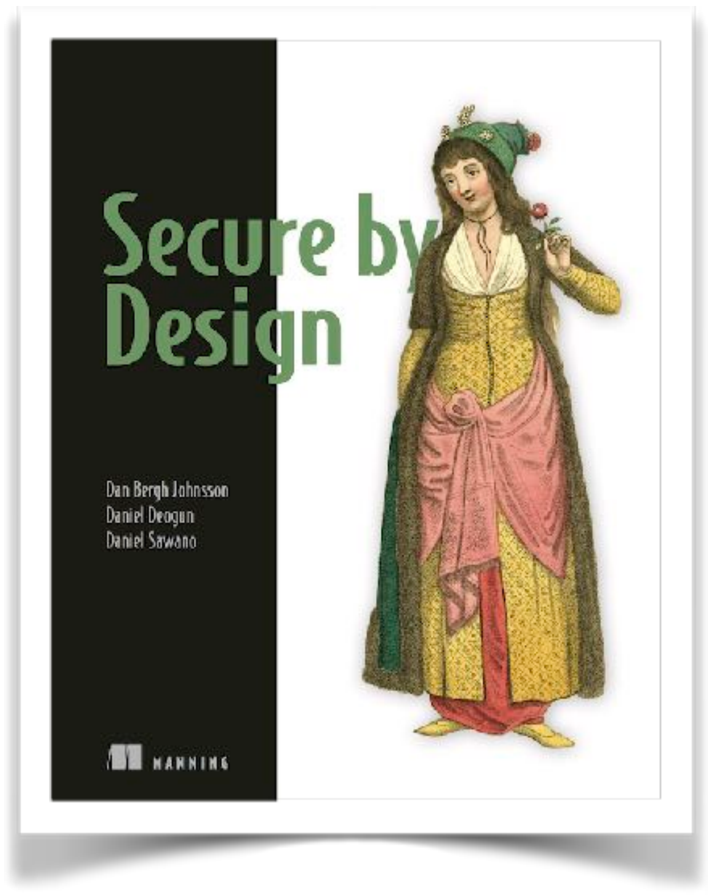
bit.ly/secure-by-design

Q&A


Secure by Design
Dan Bergh Johnsson
Daniel Deogun
Daniel Sawano
MANNING

[2]

AVANZA



Thanks!