Security Potentials & Challenges of Micro-Service Architectures

Ralph Janke
About eSentire

» CyberSecurity Service Provider

» Managed Detection Response

» Integrated “grey-matter” / silicon solution

» Protecting SMBs
Motivation

» Service Oriented Architecture are often implemented through Microservices

» Architecture of scalable solutions

» Management of scalable solutions

» Arising aspects for security
Introduction

» Microservices
  - Example Architectures

» Areas of Security Posture
  - Services
  - Environment
Terminology

» SOA – Service Oriented Architecture
  - Self-contained services for other components

» Containers
  - Linux Containers like Docker®, Rocket®

» Orchestration
  - Deployment of Containers
Goals

» Building a flexible & scalable production environment

» Being able to deploy fast (automated)
  - Continuous Deployment (DevOps)

» Using the right tool for the problem

» Mastering security challenges
Micro-Services
Micro-services

» Ideal implementation of SOA

» Each micro-service is independent service

» Each service provides responses to request either from outside the services or another service
Micro-services

» Mostly 2 different architectures
  - Synchronous (API) services
  - Asynchronous data pipeline
Synchronous / API Services

» Mostly 3 parts

- Front end receiving requests
  • i.e. AWS API Gateway

- Messaging Broker
  • i.e. 0MQ, AMPQ Implementation

- Workers serving requests
  • i.e. AWS Lambda services

» Bi-directional communication

» Good for interactions (Uis)

» Can have performance adverse properties
Synchronous / API Services
Asynchronous / Data Pipeline

» Mostly chain of services

- Messaging Broker connecting chain links
  - i.e. Kafka, 0MQ, AMPQ Implementation
  - AWS Kinesis, Google PubSub

- Workers serving requests
  - i.e. AWS Lambda services

» Uni-directional communication

» Good for performance

» Does not support interactivity
Asynchronous / Data Pipeline

Source

Worker A

Sink/Source

...

Sink/Source

Worker Z

Sink
Security
Security Footprint Services

» Decoupling of functionality can reduce exploitable footprint

» Each service is isolated in its own container using minimalistic approach -> less exploits
  - Only packages needed for particular service are included in container

» Breach of one container does not allow intrusion into other containers
Security Footprint Services

» Decoupling creates barriers to gain direct access to persistences (databases)
  - Worker is isolated from API Gateway
    • Access to database only through workers
    - Databases have no external exposure

» Containers need to be sandbox so no exposure to orchestration tools

» Additional considerations of environment
Security Posture

» Control of sources

» Control of build system

» Control of container images

» Control of Infrastructure

» Control of orchestration
Security - Sources

» Nice way to breach services is to taint sources – sources need to be protected

» Secure Source Control System
  - Accessibility (Multi-factor authentication)
  - Multi-tenancy vs Self-hosted
  - Cloud vs On-Prem

» Code Reviews increase Security

» Merge / Pull Request Gateways

» Third Party Components
Nice way to breach services is to taint build system – build needs to be protected

Secure Build Control System

- Accessibility (Multi-factor authentication)
- Multi-tenancy vs Self-hosted
- Cloud vs On-Prem

CVE Scanners

- Claire allows scanning of container images
  - https://github.com/coreos/clair

Merge / Pull Request Gateways
Security – Infrastructure

» Secure Infrastructure
  - Accessibility (Multi-factor authentication)
  - Multi-tenancy vs Self-hosted
  - Cloud vs On-Prem

» Reviewable Infrastructure Definitions
  - Example: Terraform from Hashicorp
  - Templating of best practices:
    ● IP base firewall rules
    ● Secure Storage for certificates, tokens etc.
  - Example Vault from Hashicorp
Security – Orchestration

» Secure Orchestration
  - Accessibility (Multi-factor authentication)
  - Multi-tenancy vs Self-hosted
  - Cloud vs On-Prem

» Sandboxing can reduce vulnerabilities
  - Example: Kubernetes vs. Docker compose
  - Docker Volume Mount Vulnerability
General Security Considerations

» Generally good lock down of services (only open up what is needed)
  - ACLs
  - Containers
  - IP-based restrictions
  - No default password !!!!!!!

» Always use encrypted communication

» Public-Apis:
  - Authentication through Tokens
Authentication

» Periodic Renewal of Tokens, Certificates, Secrets

» MFA (Multi-factor authentication)
Conclusion

» Micro-services fit very well the scalable and fast-pace nature needed for security

» Sandboxing through containers can reduce exposure

» Continuous Deployment allows faster reaction after detection of vulnerabilities

» Main parts of security management:
  - Source / Build / Container Images
  - Infrastructure
  - Orchestration
  - Services

» Good security practices
Conclusion

» Micro-services fit very well the scalable and fast-pace nature needed for security
» Sandboxing through containers can reduce exposure

» Continuous Deployment allows faster reaction after detection of vulnerabilities

» Main parts of security management:
  - Source / Build / Container Images
  - Infrastructure
  - Orchestration
  - Services

» Good security practices
Recruiting

» We are always looking for smart and passionate people that want to make a difference in the security of our networks

» Talk to us !!!!