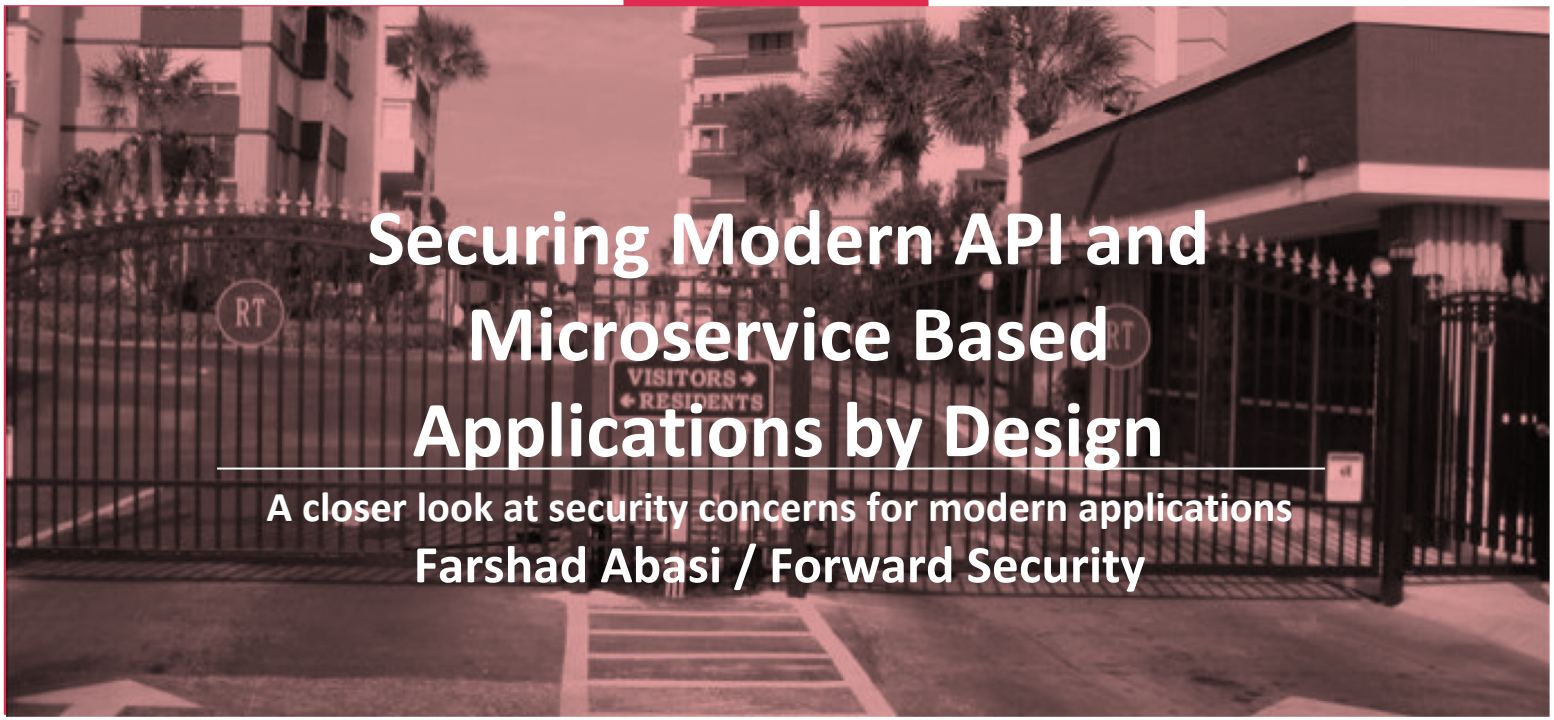


Securing Modern API and Microservice Based Applications by Design

A closer look at security concerns for modern applications

Farshad Abasi / Forward Security

FWDSEC»

Farshad Abasi

- Based in: Vancouver, BC, Canada
- UBC alumnus: Biology, Computer Science
- CSO/Founder: Forward Security
- Instructor: BCIT
- News correspondent: CFAX AM1070 (Victoria)
- Board member: BSides Vancouver / MARS
- Chapter Lead: OWASP Vancouver
- Avid music fan and food lover!

FWDSEC

This presentation provides an overview of the subject matter, and architectural take-aways from a security perspective.

FWDSEC»

- What are services and microservices?
- What about APIs?
- The API Gateway and the Post-monolithic World
- Importance of user-level security context and E2E trust
- The Need for AuthZ
- What AuthN and AuthZ Protocols Can I use?
- Invocation by external applications and services
- What other security issues should I care about?
- Take-away considerations

FWDSEC

What Are Microservices?

FWDSEC

*"**Microservices** are a software development technique - a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight."*

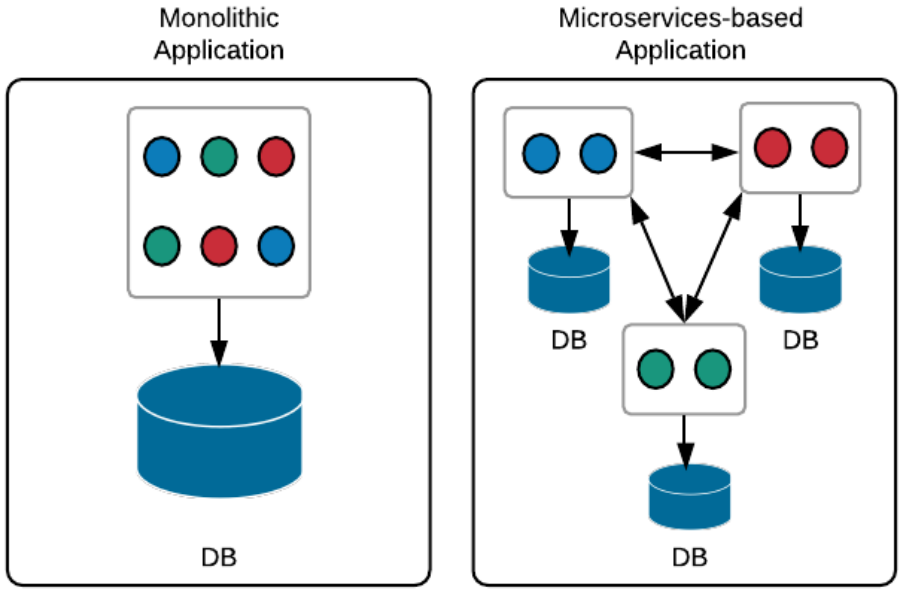
- Wikipedia

Monolithic Applications

- Functions tightly coupled together
- Do not scale well
  - Specially when different components had different resource requirements
- Often large and too complex to be understood by one developer
- Slow down development and deployment
- Do not protect components from other components' issues
- Difficult to rewrite if you need to adopt new frameworks

FWDSEC

- General move towards re-architecting traditional ”monolithic” services and applications
- Applications and services decomposed into smaller “microservices”



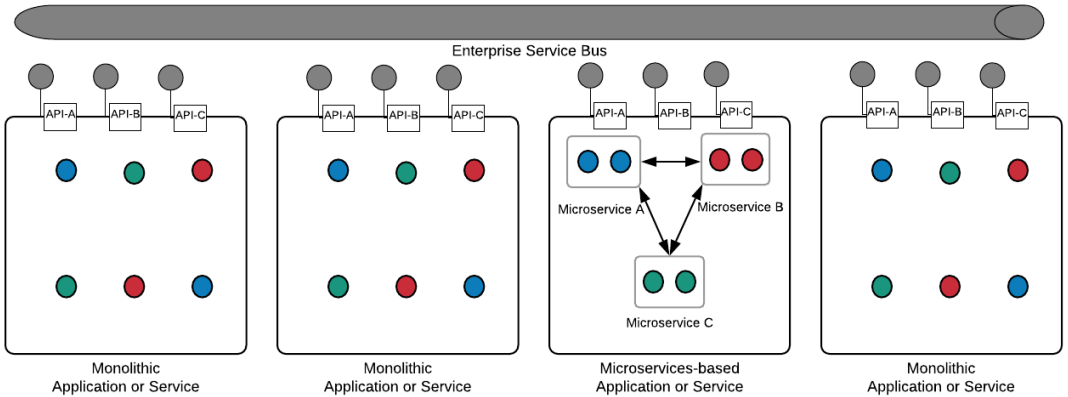
FWDSEC

Service Oriented Architecture (SOA)

- Loosely coupled components, each delivering a specific service
- Services communicate using an Enterprise Message Bus and SOAP + XML

Microservices Architecture

- Also made up of loosely coupled component
- Each MS delivers a specific function (more fine-grained than SOA)
- Typically communicate using a gateway and HTTP



FWDSEC

Microservices

- Hosted inside containers
  - Lightweight, less overhead
- Typically exposed via APIs
  - Directly or via gateway
  - Light-weight protocols used (HTTP + JSON)
  - Not all MS are exposed externally
- Each one is independent
  - Has its own database
  - Allows for decoupling from others
  - Well suited for Agile and CI/CD

- New set of challenges
  - Complexities of distributed systems development
  - Maintenance of data consistency across the microservices
  - Deployment
  - Increased memory consumption

FWDSEC

What About APIs?

FWDSEC

*"(**API**) In general terms, is a set of clearly defined methods of communication among various components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer."*

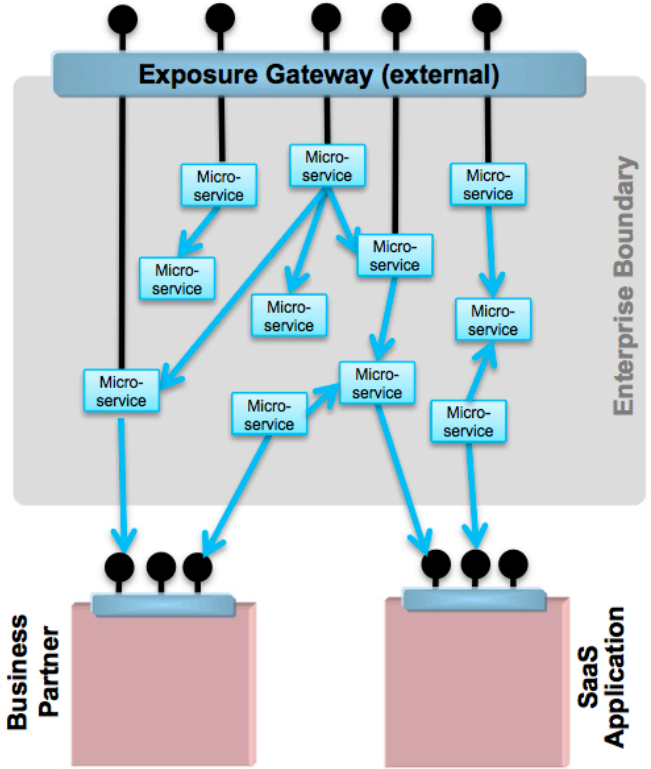
- Wikipedia

Application Programming Interface (API)

- Used by software components to communicate
- Been around for a long time
- Can be used to build components that work together
  - Inner working knowledge of other components is not required
  - Only need to know what functions are exposed (good separation)

FWDSEC

- Many modern applications are web-based
  - Expose Web APIs using HTTP and JSON
- Web APIs typically RESTful & follow ROA
- Used to communicate to:
  - Internal components/functions (intra-application)
  - External components/functions (inter-application)



FWDSEC

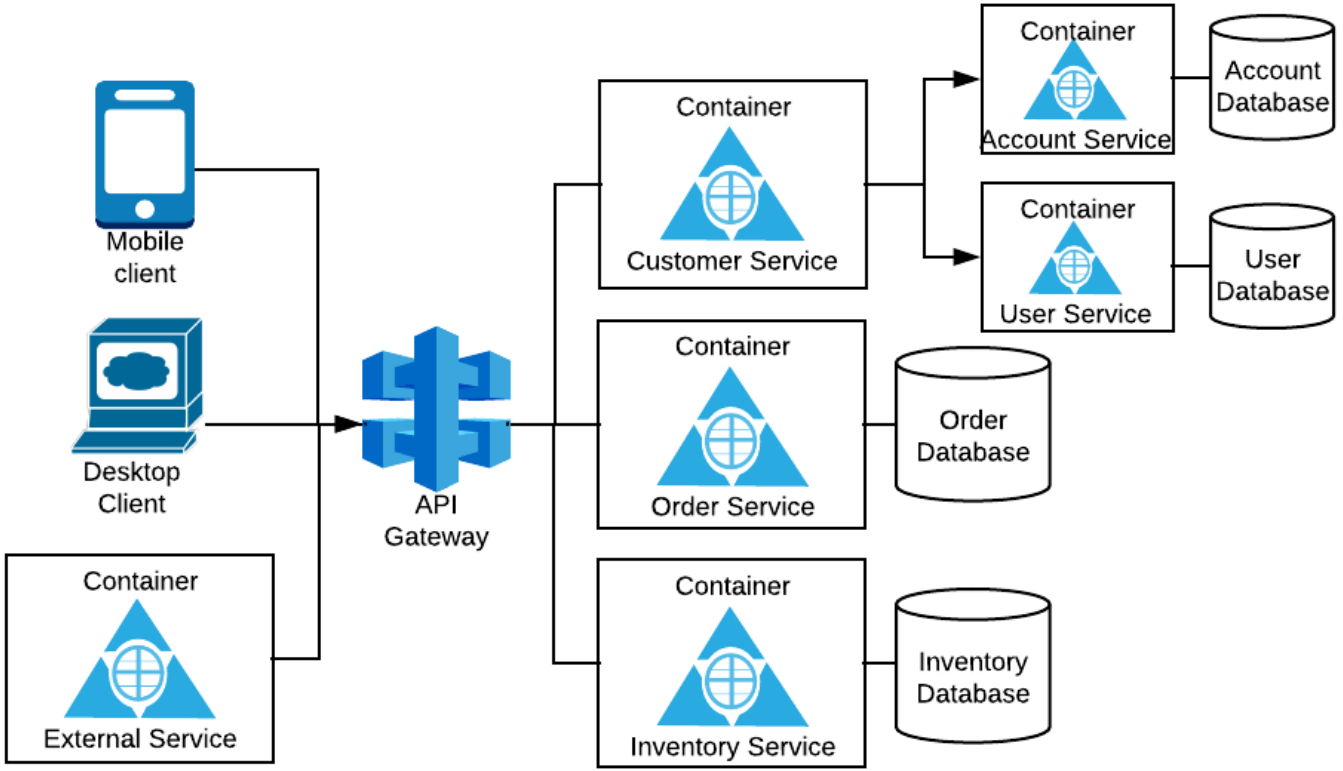
The API Gateway and the Post-Monolithic World

FWDSEC

- Monolithic applications
  - All functions inside the same “walled garden”
  - Protected by single point of entry
  - User-journey does not typically require further AuthN
  - Functions inside cannot be invoked by outsiders
  - AuthZ can happen at gate or functional level
- Microservices
  - No more “wall” with single point of entry
  - Each function/MS could be accessed via its API
  - Need to ensure user is authentic without full re-AuthN

FWDSEC

The API Gateway and the Post-monolithic World



FWDSEC

API Gateway - security benefits

- Acts as the single point of entry
  - Central enforcement of security policies (AuthN, AuthZ, etc.)
  - Guards the APIs/microservices
- Can issue tokens once security policies are met
  - ”Heavy” AuthN takes place at the gateway
    - Should use an IAM to properly handle AuthN + AuthZ
  - Tokens used for “light” AuthN & AuthZ when accessing microservices
    - E.g. ID token and Access Token in OpenID Connect
    - Lowers the exposure of “long term” credentials

FWDSEC

API Gateway - non-security benefits

- Can expose different APIs for each client
- Routes requests to different microservices
  - e.g. desktop vs. mobile
- Can create “mash-ups” using multiple microservices
  - Microservices may be too granular to access individually
- Abstracts service instances and their location
- Hides service partitioning from clients
  - Can change over time
- Provides protocol translation

FWDSEC

Importance of User-level Security Context and E2E trust

FWDSEC

- User-level security context
  - knowing who the user is and what level of authN they passed
- In traditional systems, users-level security context is not carried throughout the call chain
  - Downstream components perform service level authN (e.g. logic layer to data layer)
- Problem: access to all data provided to a service account
  - Too permissive and against principle of least privilege
  - No comprehensive auditing of specific actions related to a user

FWDSEC

Importance of user-level security context and E2E trust

- All services should take user-level security context into account before allowing access
- A security token can be used to determine user's security context before servicing the request
  - Tokens must be signed (also encrypted if there is a confidentiality concern)
- API invocation may involve calls across several downstream microservices
  - Need to authenticated user's security context
  - = User-level End-to-end (E2E) trust

FWDSEC

- User-level E2E Trust
  - Communicate the authenticated user's security context to all parties across the call chain
  - Each party can take appropriate action based on user security context
  - Can use a token-exchange end-point or E2E trust tokens
- Service-level trust should also be in place
  - Prevents compromised trust tokens from playback by others
  - Use overlay networking, mutual TLS, service account + TLS

FWDSEC

Token-exchange end-point

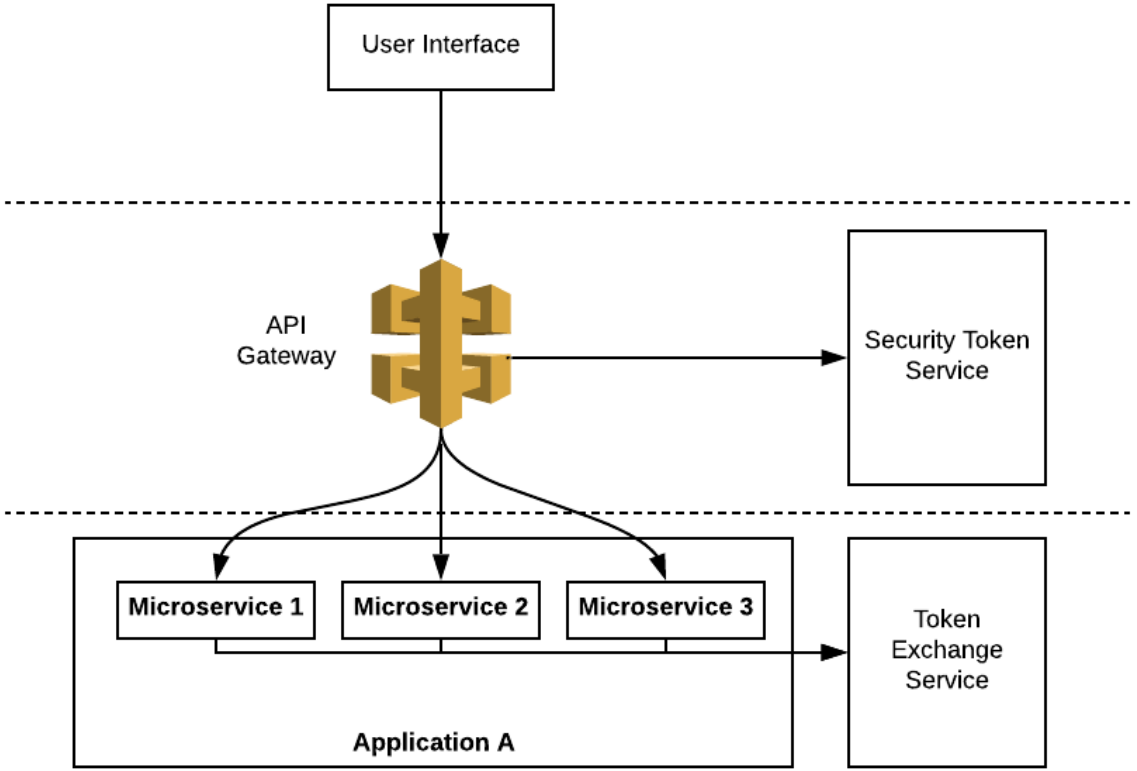
- Services can exchange a token for one with the required scope + protocol for the next service
- Limits the amount of access granted and allows for heterogenous protocol use
- May be handled by the same end-point that issues the initial security tokens
- Each microservice verifies the token and provides access based on context, claims, scope

E2E trust tokens

- Generated at the gateway after initial set of policies are enforced
- All microservices must have the same scope + protocol, or policy information

FWDSEC

Importance of user-level security context and E2E trust



FWDSEC

- User-level E2E trust is established among microservices of the same application or service
  - All in the same application trust-zone (same IdP, same policies)
- What about access to microservices in another trust-zone?
  - Need to go through API-gateway for that microservice
  - Can accept the security token if there is a trust relationship
  - E2E trust tokens need to be re-issued

FWDSEC

The Need for AuthZ

FWDSEC

- Each microservice may have specific authZ requirements
  - E.g. Bank account microservice to only allow read ops to a chequing account for a specific user with basic authN, but allow read/write with 2FA
- Security token service can issue a token based on:
  - Resource requested
  - Scope or type of request
  - User's security context
- Token's claims can include:
  - End-user and issuer's identity
  - Identities of specific consumers
  - Expiration time

FWDSEC

- Each microservice verifies the authenticity of the token first
- Access is provided based on valid authorized claims and scope
- Using a single E2E trust token, each microservice handles authZ
  - Requires policy information at each microservice
- Token-exchange service can be used to obtain a new token for downstream calls
  - Allows more flexibility and better access control

FWDSEC

What AuthN and AuthZ Protocols Can I Use?

FWDSEC

What AuthN and AuthZ Protocols Can I Use?

- SAML
  - Used for AuthN and AuthZ; XML based
- Open ID (Now replaced by OpenID Connect)
  - Used for AuthN only
- OAuth (latest V. 2.0)
  - Provides AuthZ only (access token), Used for access delegation
  - Can use JSON Web Tokens (JWT)
- OpenID Connect (latest V. 1.0)
  - Used for AuthN + AuthZ; Extension of OAuth 2.0; Uses JSON
  - Supports ID + access tokens

FWDSEC

Invocation by External Applications and Services

FWDSEC

- APIs may be invoked by services outside trust boundary
- Establish trust between the two domains
  - Verify authenticity of user's security context presented by the token
- API gateway can then issue a new security trust token
  - To be used downstream
  - Security policies may be different, so should not accept security tokens from another application or service
  - Should only be issued if appropriate security policies pass
- Without central handling, each microservice would need a trust relationship with external consumers!

FWDSEC

What Other Security Issues Should I Care About?

FWDSEC

- In addition to AuthN and AuthZ, consider:
  - Rate-limiting
  - JSON threat protection
  - XML threat protection
  - Custom policies
    - Centrally address other application security attacks or specific needs
- Provide pre-built policies to developers
  - Eases the development process
  - Avoids learning curve and mistakes

FWDSEC

- Grouping objects and applying policies is not new
- Helps in consistent application of policies
- This principle should be applied to APIs as well
  - Group APIs that address a particular business need
  - Provide the group to Devs along with a service plan and set of policies
- Security trust token issued valid only for APIs in that group
  - Account Servicing application (internal, XML based, uses AD)
    - Policies: AD AuthN/AuthZ, XML threat protection, rate-limiting for internal
  - Shopping site (external, JSON based, uses LDAP)
    - Policies: LDAP AuthN/AuthZ, JSON threat protection, rate-limiting for external

FWDSEC

- Detective controls are important
  - E.g. logging and monitoring
- APIs should log important events including security
  - Logs should be centrally aggregated, correlated and monitored
  - Success or failure of policy checks should be logged
  - Other info useful in DoS detection and better profiling of the system
    - E.g. number of API invocations/sec, response time, etc.
- API logs can help with fraud prevention
  - E.g. by providing info about consuming device and location

FWDSEC

Take-away Considerations

FWDSEC

- Maintain user-level E2E trust across the entire journey
- Enforce AuthZ at the right place & right level of granularity
- Group APIs to apply configurable policies consistently
- Use an API gateway to centrally enforce policies
- Don’t forget to log, monitor and detect
- Follow a defense-in-depth strategy
- Build security into the service architecture and design

FWDSEC

Q&A

Contact: Farshad Abasi
f.abasi@fwdsec.com / @FWD_SEC

https://developer.ibm.com/series/securing-modern-apps-api-microservices/

FWDSEC»