What Information About The Application Architecture Needs To Be Collected For AppSec Purposes?

February 25, 2020
sec4dev Conference 2021
Alexander Barabanov

Motivation

SAMM model overview

Governance	Design	Implementation	Verification	Operations
Strategy and Metrics	Threat Assessment	Secure Build	Architecture Assessment	Incident Management
Policy and Compliance	Security Requirements	Secure Deployment	Requirements-driven Testing	Environment Management
Education and Guidance	Security Architecture	Defect Management	Security Testing	Operational Management

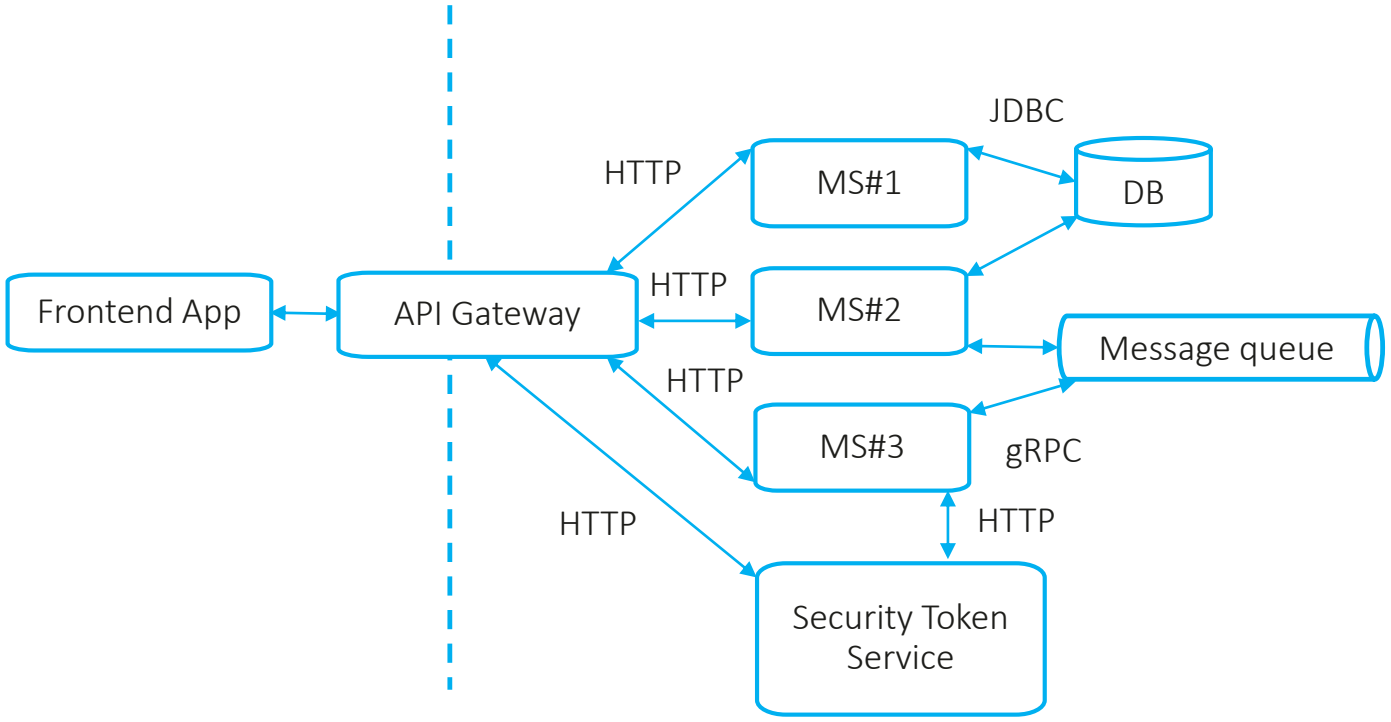
Maturity level		Stream A Application Risk Profile	Stream B Threat Modeling
1	Best-effort identification of high-level threats to the organization and individual projects.	A basic assessment of the application risk is performed to understand likelihood and impact of an attack.	Perform best-effort, risk-based threat modeling using brainstorming and existing diagrams with simple threat checklists.

Where I can find diagrams and design documents suitable for threat modeling?

About me

- Ph.D. in Computer Science
- Currently: Principal Security Engineer at Advanced Security Research Team, Advanced Software Technology Lab, Huawei
- Associate Professor at Bauman MSTU, Information Security Department
- Prior: Application security architect at Baker Hughes/Rosneft, Security expert at NPO Echelon
- CISSP, CSSLP
- OWASP contributor: CS series, microservice security

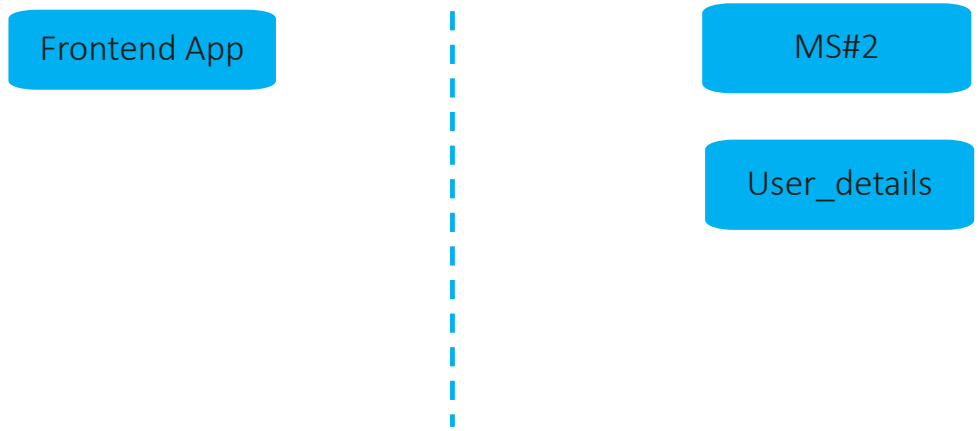
Why to document security architecture?



1. Threat modeling and enforcement of the principle of least privilege:
   - What scopes or API keys does microservice minimally need to access other microservice APIs?
   - What grants does microservice minimally need to access database or message queue?
2. Data leakage analysis:
   - What storages or message queues do contain sensitive data?
   - Does microservice read/write date from/to specific database or message queue?
   - What microservices are invoked by dedicated microservice? What data is passed between microservices?
3. Attack surface analysis:
   - What microservices endpoints need to be tested during security testing?

Collect information on the building blocks: step 1

1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…



Identify and describe application-functionality services

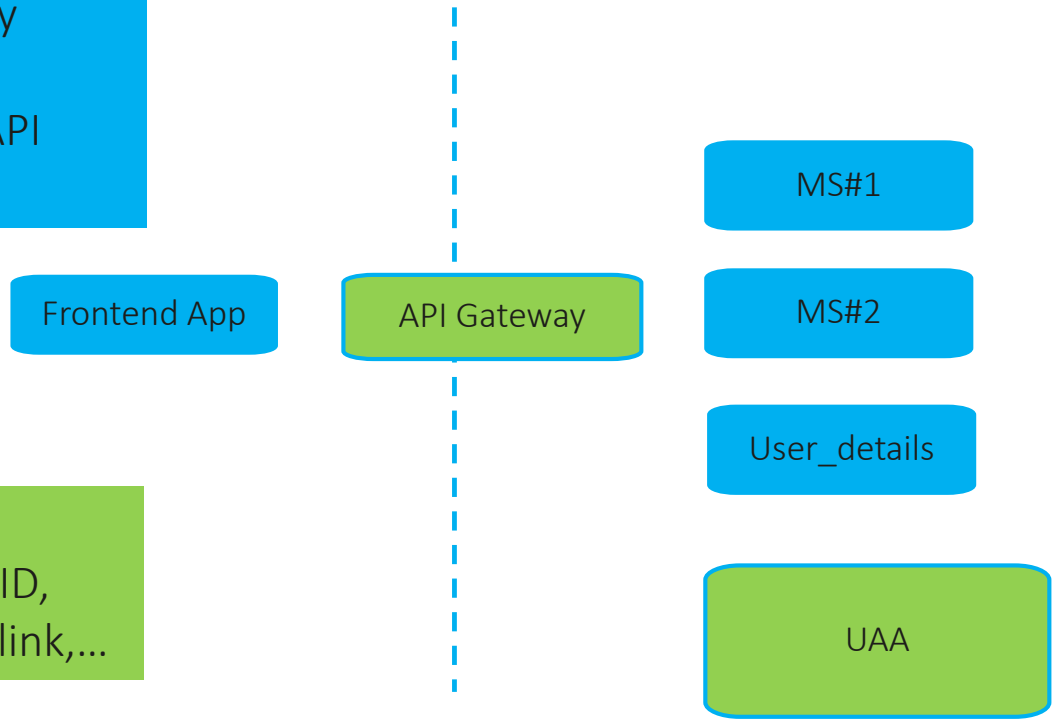
- Testing environment
- Product owner/System architect/Development Lead
- Source code repository
- HLD/LLD artifacts

Parameter name	Description
Service name (ID)	User_details
Short description	Service to store and retrieve user data (ID, roles, attributes)
Link to source code repo	../User_details/tree/master/
Development Team	Alpha-team
API definition	../User_details/specification/openapi.yaml
The microservice architecture description	../User_details/LLD
Link to runbook	../User_details/Runbook

Collect information on the building blocks: step 2

1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…

2. Identify and describe infrastructure services: ID, short description, repo link,…



Identify and describe infrastructure services

- Testing environment
- System architect/Development Lead
- HLD/LLD artifacts

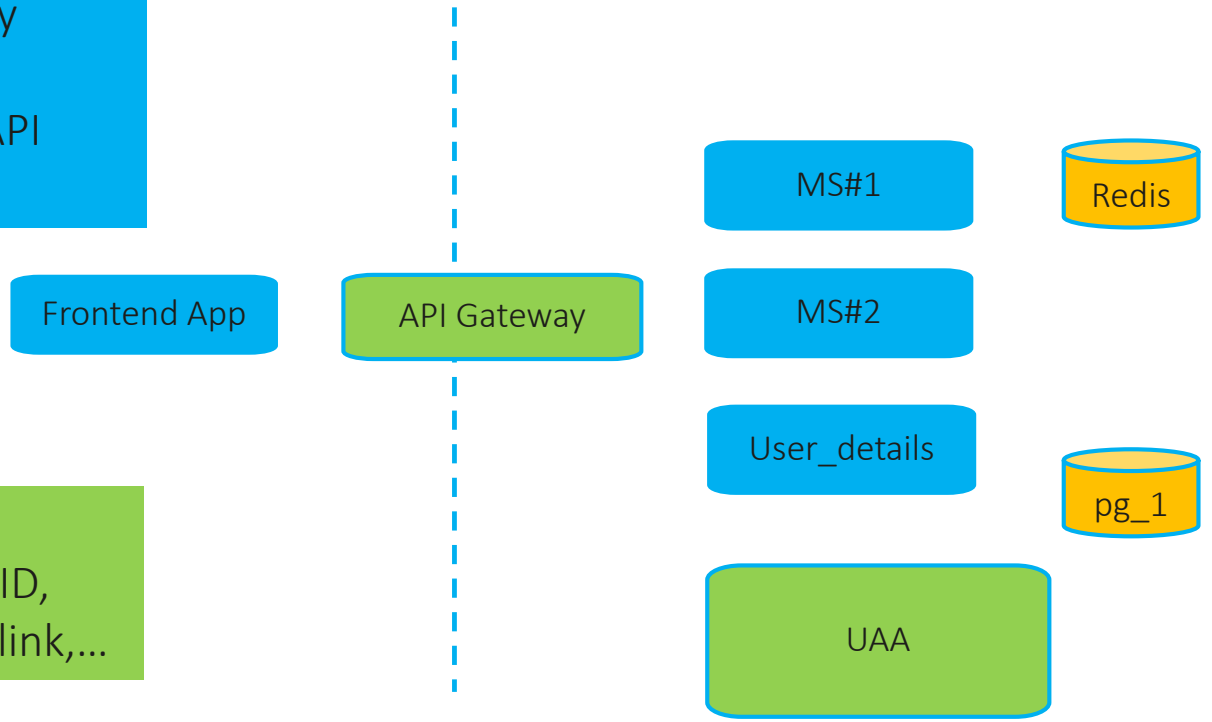
Parameter name	Description
Service name (ID)	UAA
Short description	OAuth2 Authorization service, JWT generation
Link to source code repository	https://github.com/cloudfoundry/uaa
Link to the service documentation	https://github.com/cloudfoundry/uaa/tree/develop/docs

Collect information on the building blocks: step 3

1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…

2. Identify and describe infrastructure services: ID, short description, repo link,…

3. Identify and describe data storages: ID, type



Identify and describe data storages

- Testing environment
- Parse configuration (yaml) file
- System architect/Development Lead
- HLD/LLD artifacts

Parameter name	Description
Storage name (ID)	pg_1
Software type	PostgreSQL

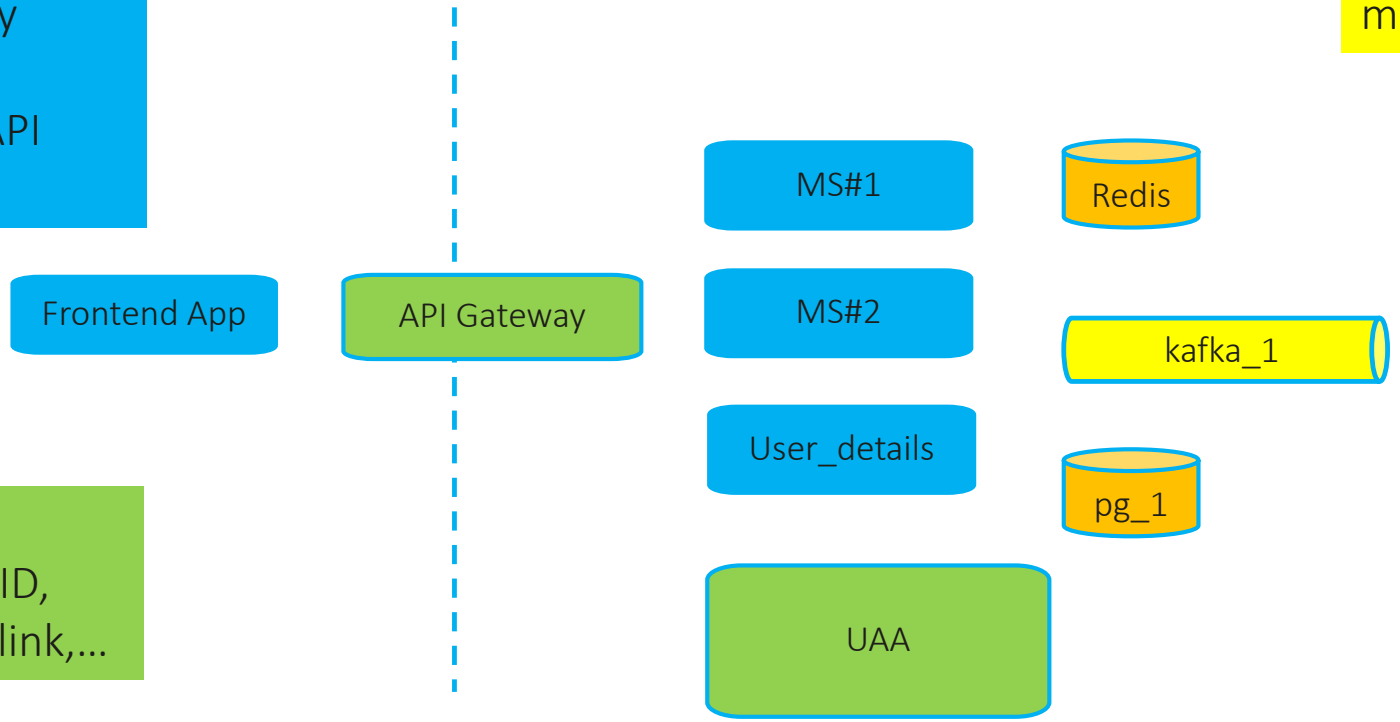
Collect information on the building blocks: step 4

1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…

2. Identify and describe infrastructure services: ID, short description, repo link,…

3. Identify and describe data storages: ID, type

4. Identify and describe messages queues: ID, type



Identify and describe messages queues

- Testing environment
- Parse configuration (yaml) file
- System architect/Development Lead
- HLD/LLD artifacts

Parameter name	Description
Queue name (ID)	kafka_1
Software type	Apache Kafka

Collect information on the building blocks: step 5

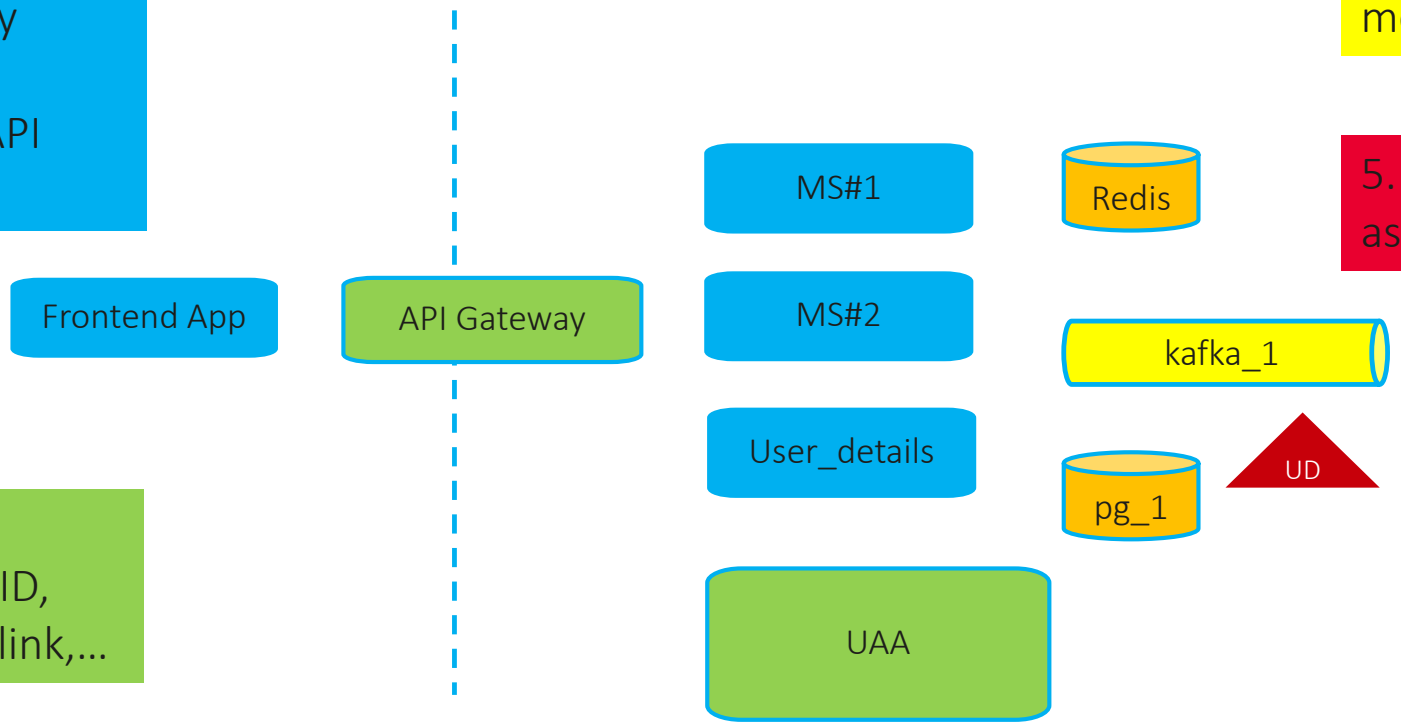
1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…

2. Identify and describe infrastructure services: ID, short description, repo link,…

3. Identify and describe data storages: ID, type

4. Identify and describe messages queues: ID, type

5. Identify and describe data assets: ID, type (e.g., PII)



Identify and describe data assets

- Product owner/System architect
- Source code repository
- Testing environment

Parameter name	Description
Asset name (ID)	User_data/UD
Protection level	PII
Additional info	User name, user roles, user attributes, social security number

Collect information on relations between building blocks: step 1

1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…

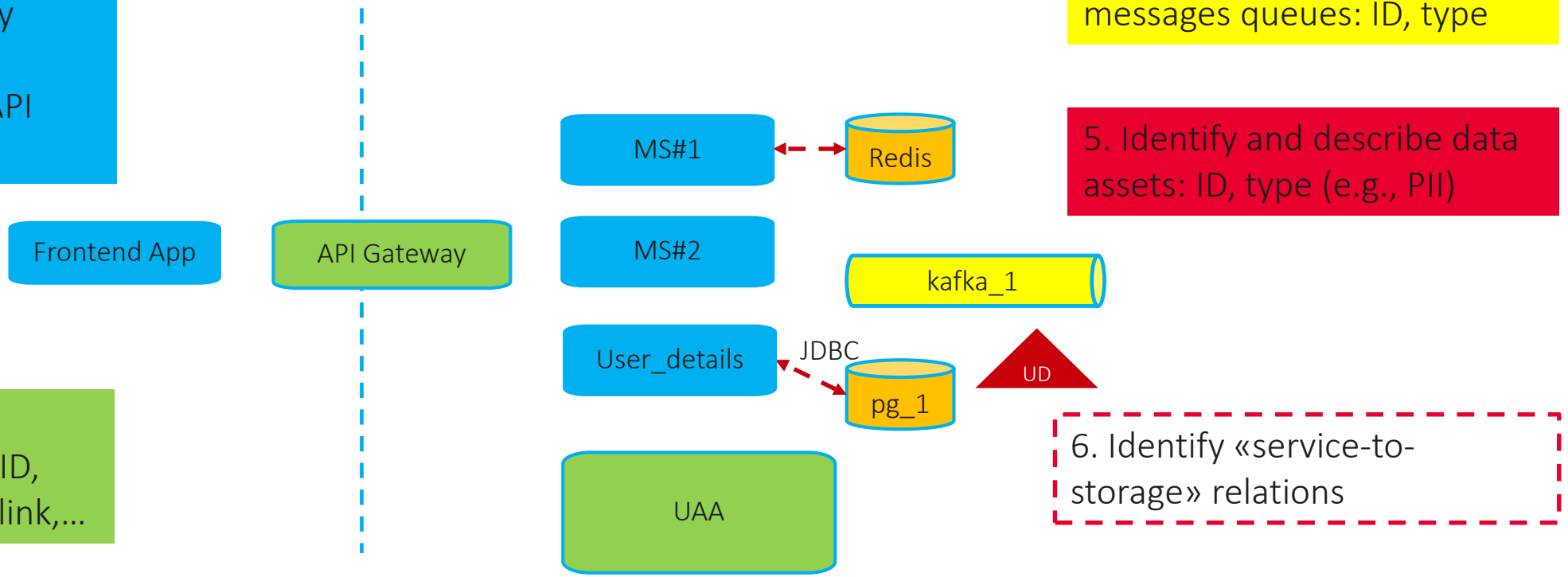
2. Identify and describe infrastructure services: ID, short description, repo link,…

3. Identify and describe data storages: ID, type

4. Identify and describe messages queues: ID, type

5. Identify and describe data assets: ID, type (e.g., PII)

6. Identify «service-to-storage» relations



Service-to-storage relations

- Testing environment
- Parse configuration (yaml) file
- System architect/Development Lead
- HLD/LLD artifacts

Parameter name	Description
Service name (ID)	User_details
Storage name (ID)	pg_1
Access type	Read/Write

Collect information on relations between building blocks: step 2

1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…

2. Identify and describe infrastructure services: ID, short description, repo link,…

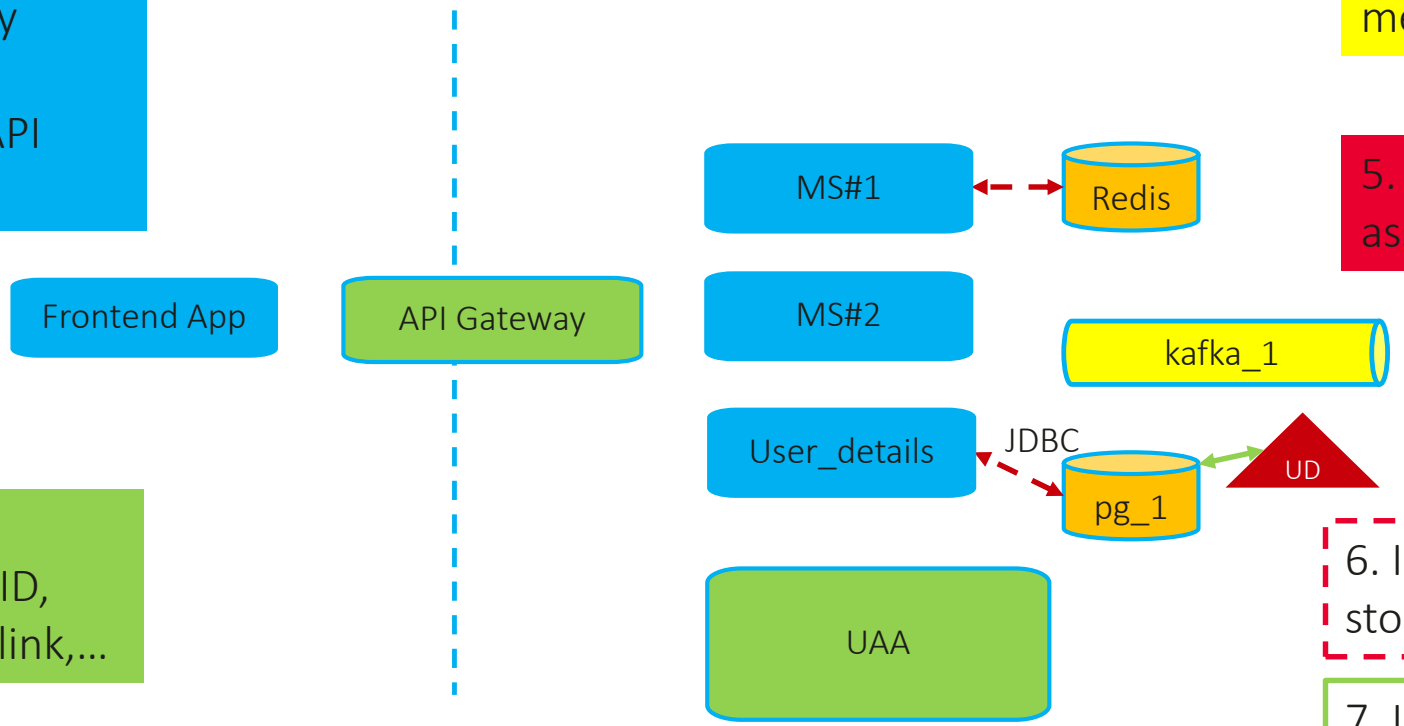
3. Identify and describe data storages: ID, type

4. Identify and describe messages queues: ID, type

5. Identify and describe data assets: ID, type (e.g., PII)

6. Identify «service-to-storage» relations

7. Identify «asset-to-storage» relations



Asset-to-storage relations

- Testing environment
- System architect/Development Lead
- HLD/LLD artifacts

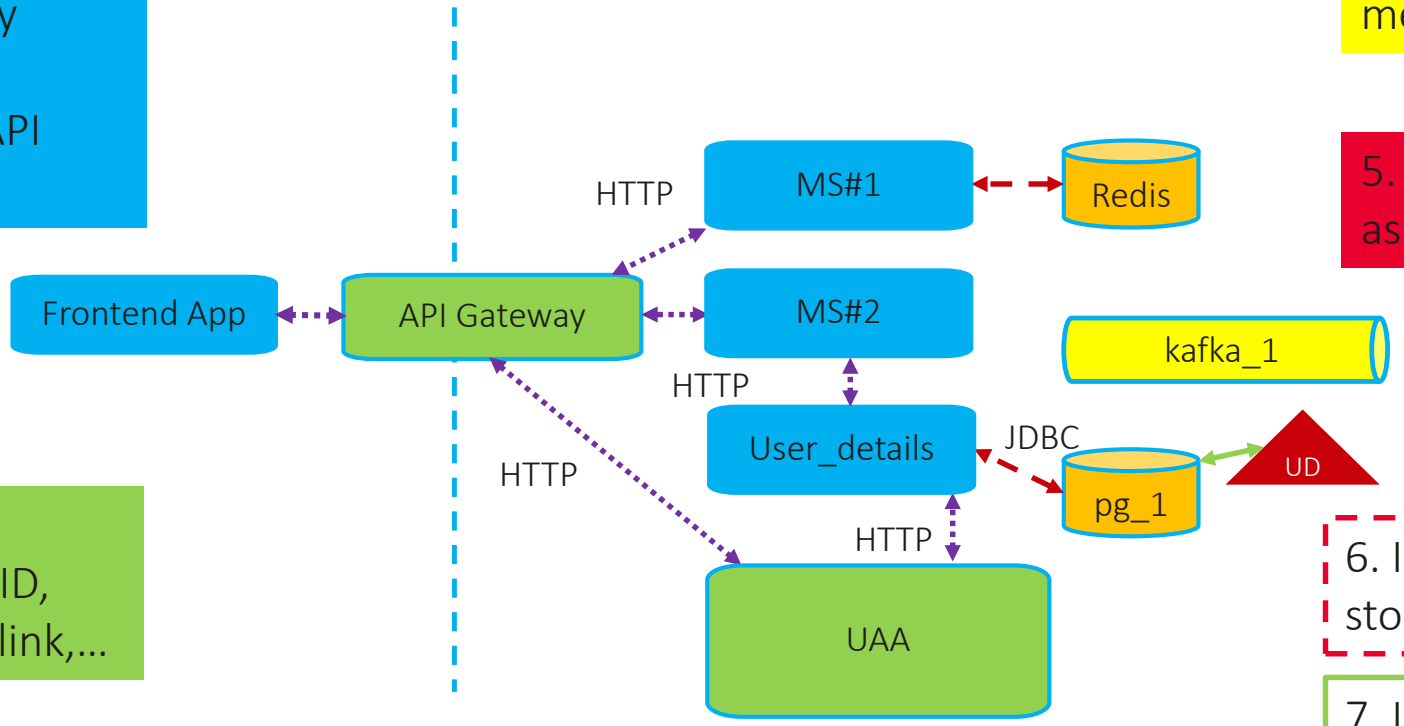
Parameter name	Description
Asset name (ID)	User_data/UD
Storage name (ID)	pg_1
Storage type	golden source

Collect information on relations between building blocks: step 3

1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…

2. Identify and describe infrastructure services: ID, short description, repo link,…

3. Identify and describe data storages: ID, type



4. Identify and describe messages queues: ID, type

5. Identify and describe data assets: ID, type (e.g., PII)

6. Identify «service-to-storage» relations

7. Identify «asset-to-storage» relations

8. Identify «service-to-service» sync relations

Service-to-service synchronous communications

- Testing environment
- Parse configuration (yaml) file
- System architect/Development Lead
- HLD/LLD artifacts

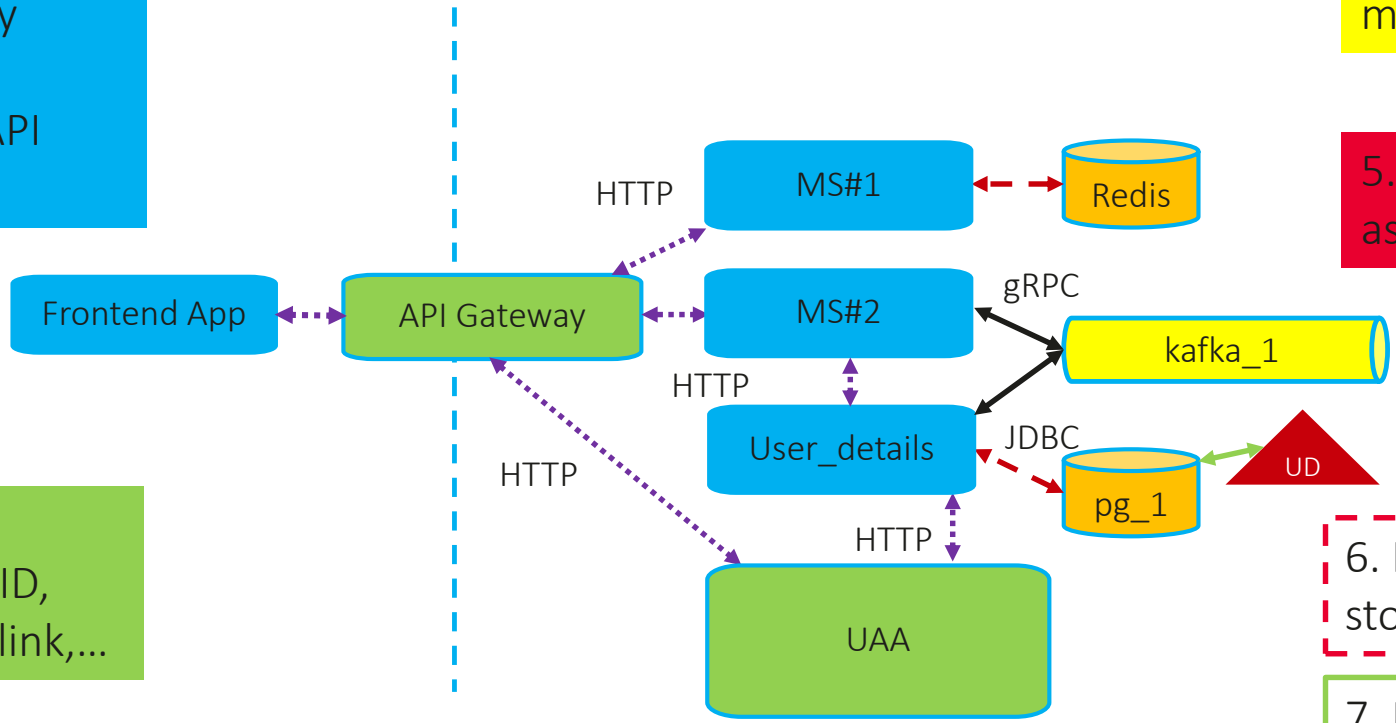
Parameter name	Description
Caller service name (ID)	User_details
Called service name (ID)	UAA
Protocol/framework used	HTTP REST
Short description	Get user security attributes

Collect information on relations between building blocks: step 4

1. Identify and describe application-functionality services: ID, short description, repo link, API spec,…

2. Identify and describe infrastructure services: ID, short description, repo link,…

3. Identify and describe data storages: ID, type



4. Identify and describe messages queues: ID, type

5. Identify and describe data assets: ID, type (e.g., PII)

6. Identify «service-to-storage» relations

7. Identify «asset-to-storage» relations

8. Identify «service-to-service» sync relations

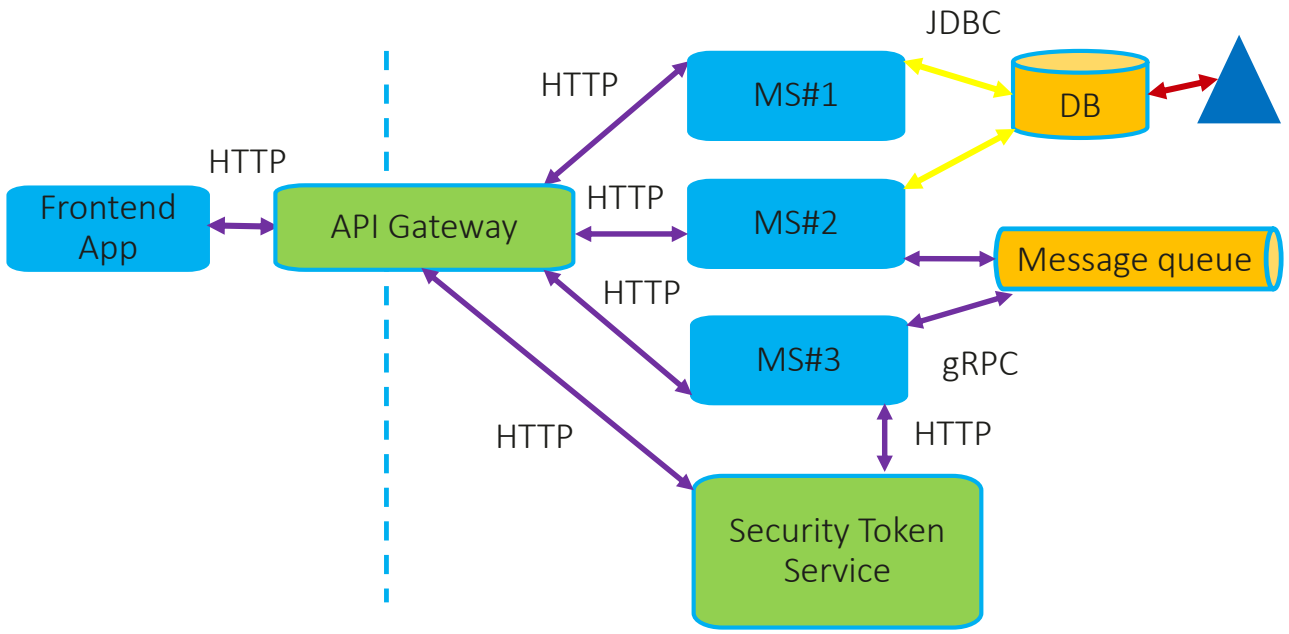
9. Identify «service-to-service» async relations

Service-to-service asynchronous communications

- Testing environment
- Parse configuration (yaml) file
- System architect/Development Lead
- HLD/LLD artifacts

Parameter name	Description
Publisher service name (ID)	User_details
Subscriber service name (ID)	MS#2
Message queue (ID)	kafka_1
Short description	User home address update

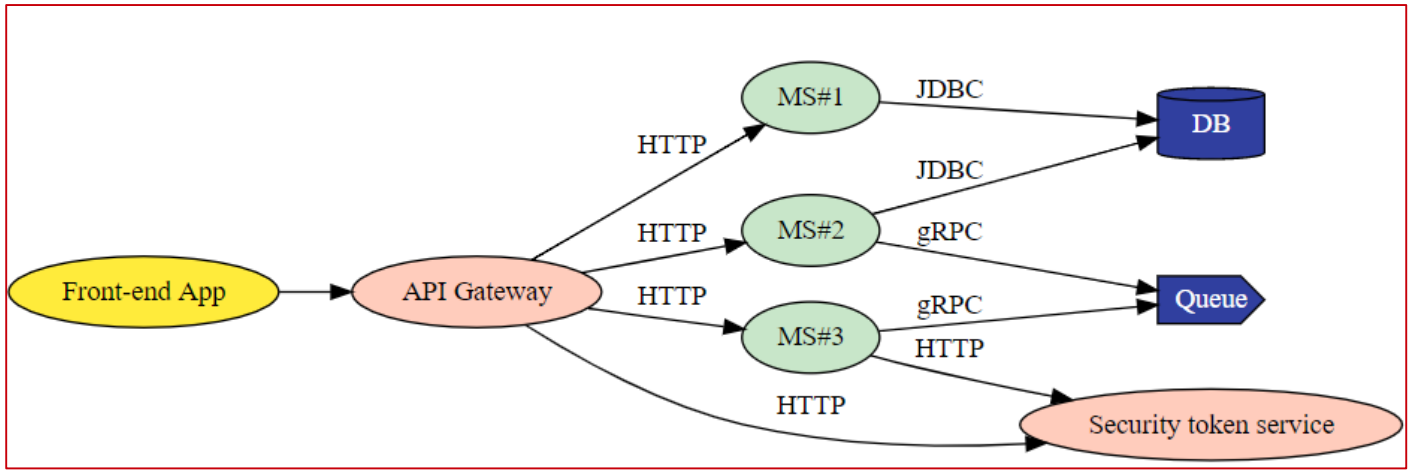
Create a graphical presentation of application architecture




```
subgraph client_side_apps {
    front_end -> {API_GW};
}

subgraph api_gateways {
    API_GW -> {STS, ms_1, ms_2, ms_3} [label="HTTP"];
}

subgraph microservices {
    ms_1 -> {db}[label="JDBC"];
    ms_2 -> {db}[label="JDBC"];
    ms_2 -> {queue}[label="gRPC"];
    ms_3 -> {queue}[label="gRPC"];
    ms_3 -> {STS}[label="HTTP"];
}
```



https://en.wikipedia.org/wiki/DOT_(graph_description_language)

OWASP ASVS: recap

The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development

OWASP ASVS requirements were developed with the following objectives in mind:

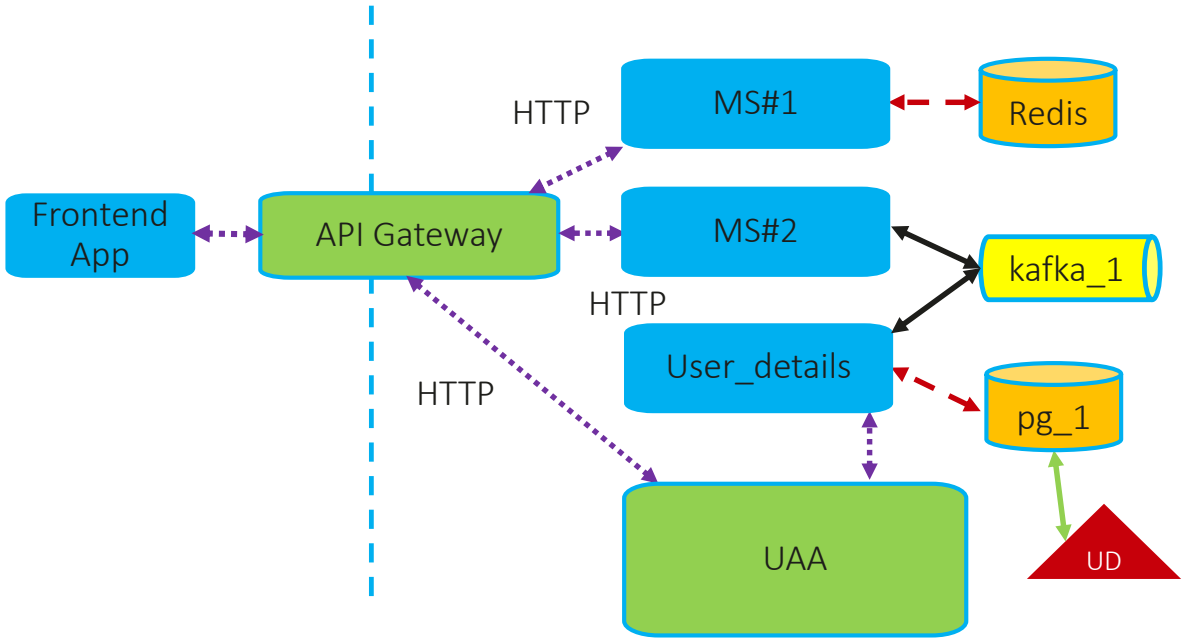
- Use as a metric - Provide application developers and application owners with a yardstick with which to assess the degree of trust that can be placed in their Web applications,
- Use as guidance - Provide guidance to security control developers as to what to build into security controls in order to satisfy application security requirements, and
- Use during procurement - Provide a basis for specifying application security verification requirements in contracts

V1.4 Access Control Architectural Requirements

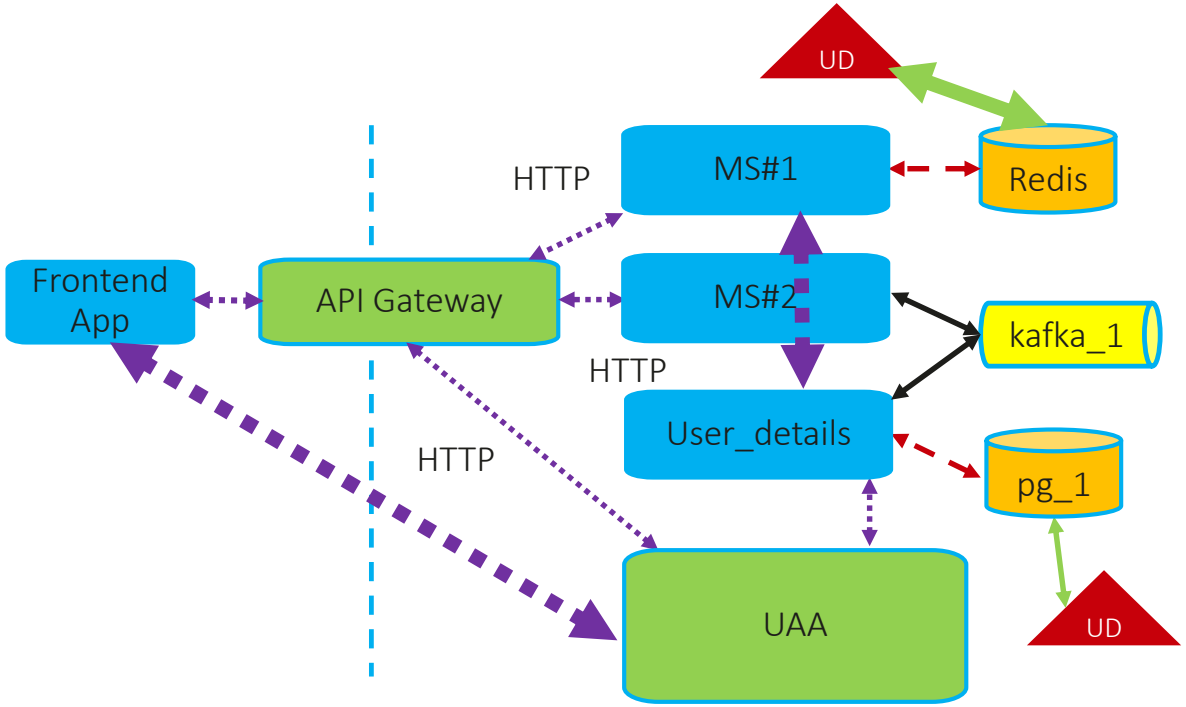
#	Description	L1	L2	L3	CWE
1.4.1	Verify that trusted enforcement points such as at access control gateways, servers, and serverless functions enforce access controls. Never enforce access controls on the client.		✓	✓	602
1.4.2	Verify that the chosen access control solution is flexible enough to meet the application's needs.		✓	✓	284
1.4.3	Verify enforcement of the principle of least privilege in functions, data files, URLs, controllers, services, and other resources. This implies protection against spoofing and elevation of privilege.		✓	✓	272

https://owasp.org/www-project-application-security-verification-standard/

Use collected information: use of threat modeling for every design change or sprint planning



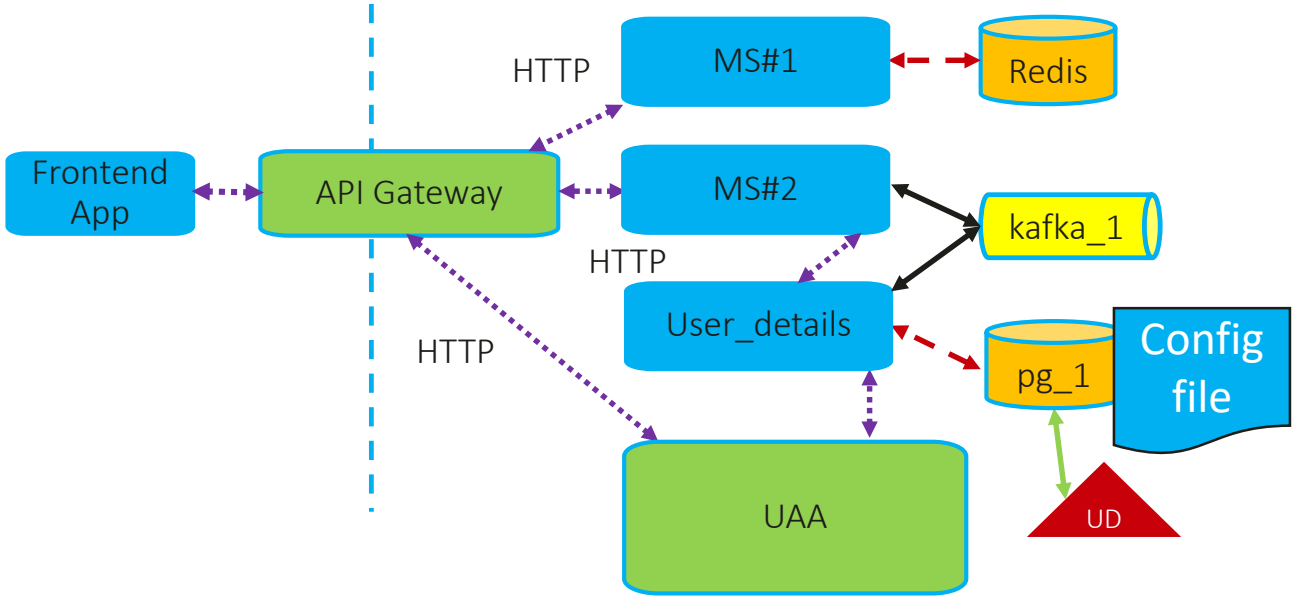
Sprint N



Sprint N+1

- Do we really need connection between frontend and UAA?
- Do MS#1 really need an access to user_details microservice and an access to user data asset?
- Where does sensitive information stored? Are there any information leakage via MS#1 and Redis?

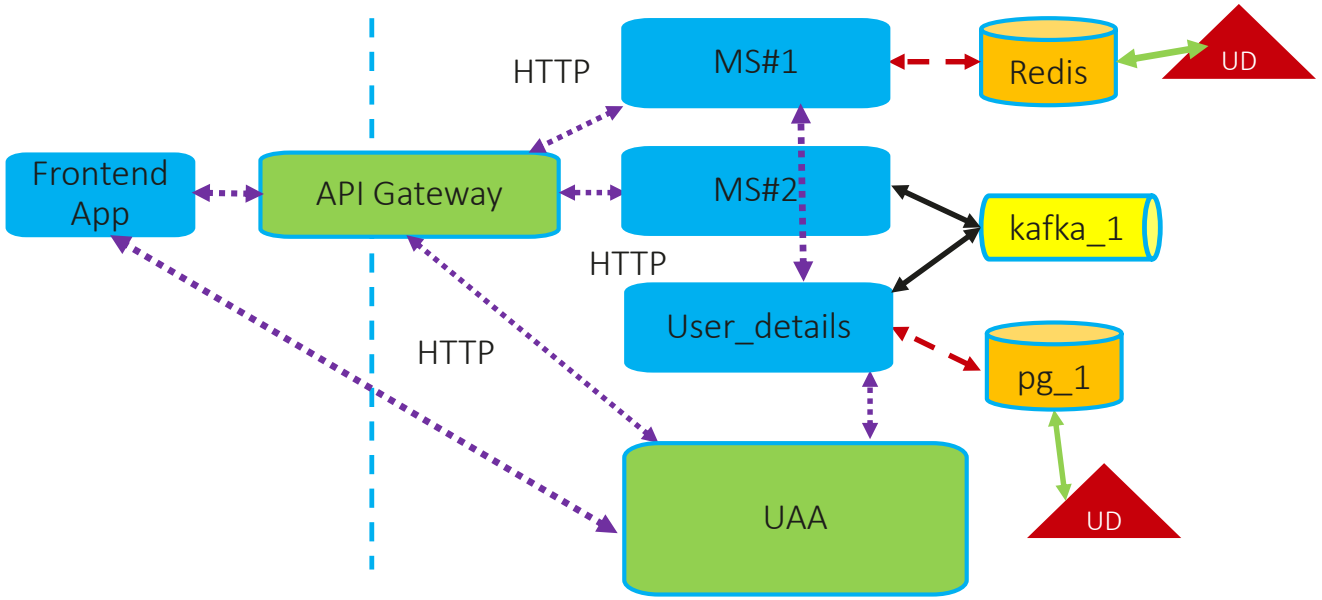
Use collected information: principle of least privilege



Parameter name	Description
Service name (ID)	User_details
Storage name (ID)	pg_1
Access type	Read/Write

- Does user_details really need Read/Write to pg_1?
- Does MS#2 really need an access to user_details?
- Does actual configuration file is equal to required permissions in the table?

Use collected information: sensitive data identification



- Where does sensitive information located?
- Does microservice read/write data asset from/to specific database or message queue?
- What microservices are invoked by dedicated microservice? Is sensitive data passed between microservices?

Takeaways

1. Document you security architecture – it will help you to find more architectural vulnerabilities (flaws)
2. Document only necessary types of information that may help you in your application security activities.
3. Actively use collected information at a daily basis (during spring planning) and keep it up to date
4. Use “software architecture as code” approach
5. Do not bury your head in the sand – collaborate with developers/sys architects/QA
6. References
   - Microservice security architecture documentation OWASP Cheat Sheet
   - Microservies Security OWASP Cheat Sheet:
     - Edge-level authorization
     - Service-level authorization
     - External entity identity propagation
     - Service-to-service authentication

Thanks for your attention!