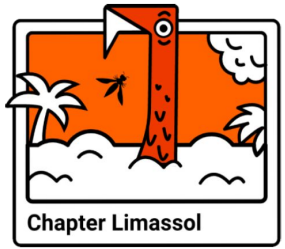
SECURITY ENGINEERING OR UNPOPULAR APPSEC

Anatoly
Makovetsky
Head of Security
Pepperstone

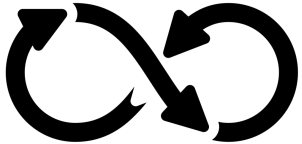


pepperstone

Table of **contents**

Why do we do AppSec?

AppSec is a part of the Software Development process focusing on **finding**, **fixing** and **preventing vulnerabilities** at the application level.



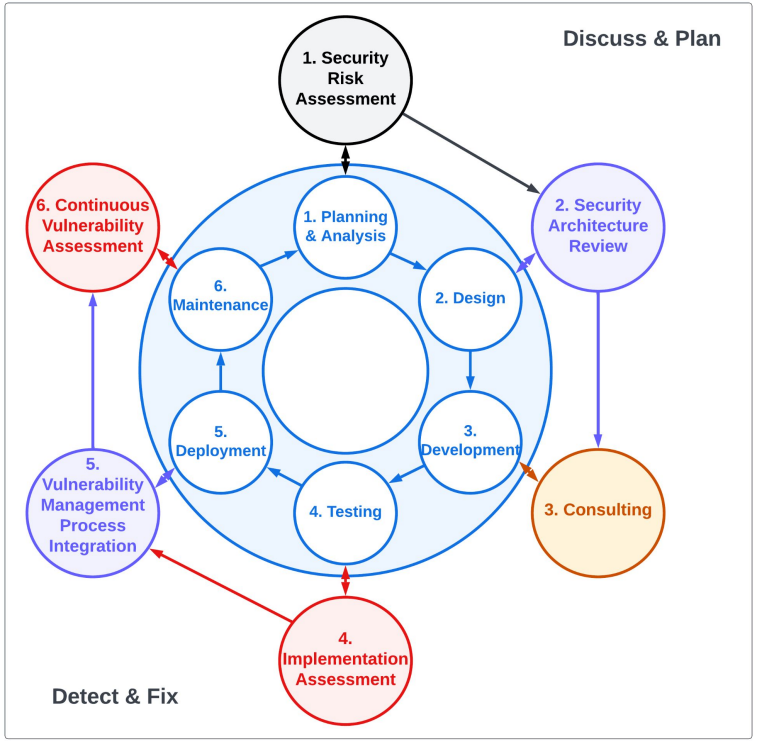
It works on overall product **quality** in the scope of **reduction** of material **risks** related to potential or real **misuse** of application functionality and processed/stored data.

“Shift left” is a popular approach of including **AppSec** to the **early stages** of the Software Development helping to act more **preventively** than detectively and correctively.

What do we do in AppSec?

Secure SDLC stages:

1. **Planning & Analysis** – Security Risk Assessment
2. **Design** – Security Architecture Review
3. **Development** – Consulting
4. **Testing** – Implementation Assessment
5. **Deployment** – Vulnerability Management Process Integration
6. **Maintenance** – Continuous Vulnerability Assessment

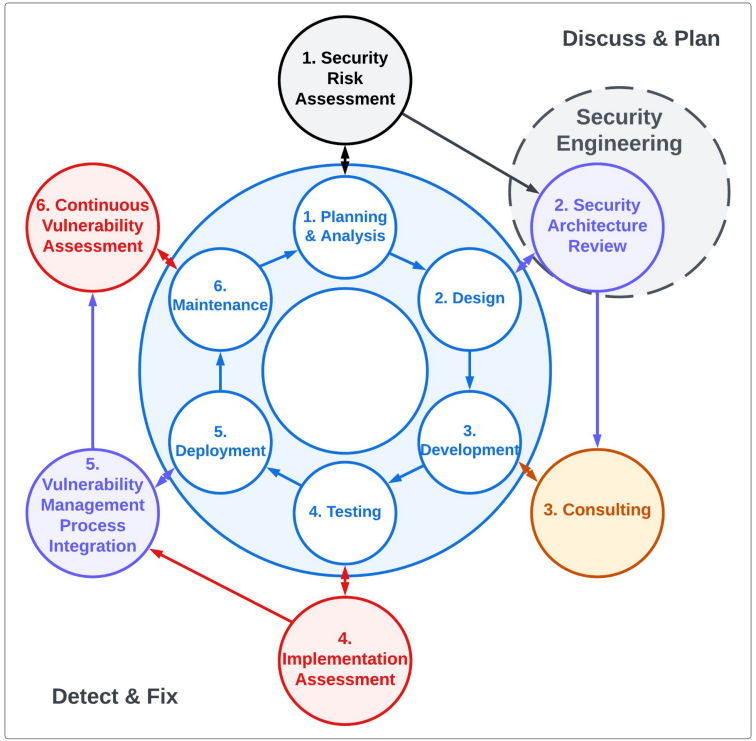


What do we do in AppSec?

"Security engineering is about building systems to remain dependable in the face of malice, error, or mischance.

As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves."

R. Anderson, "Security Engineering. Second Edition"



Where does Security Engineering stay?

Roles must be:

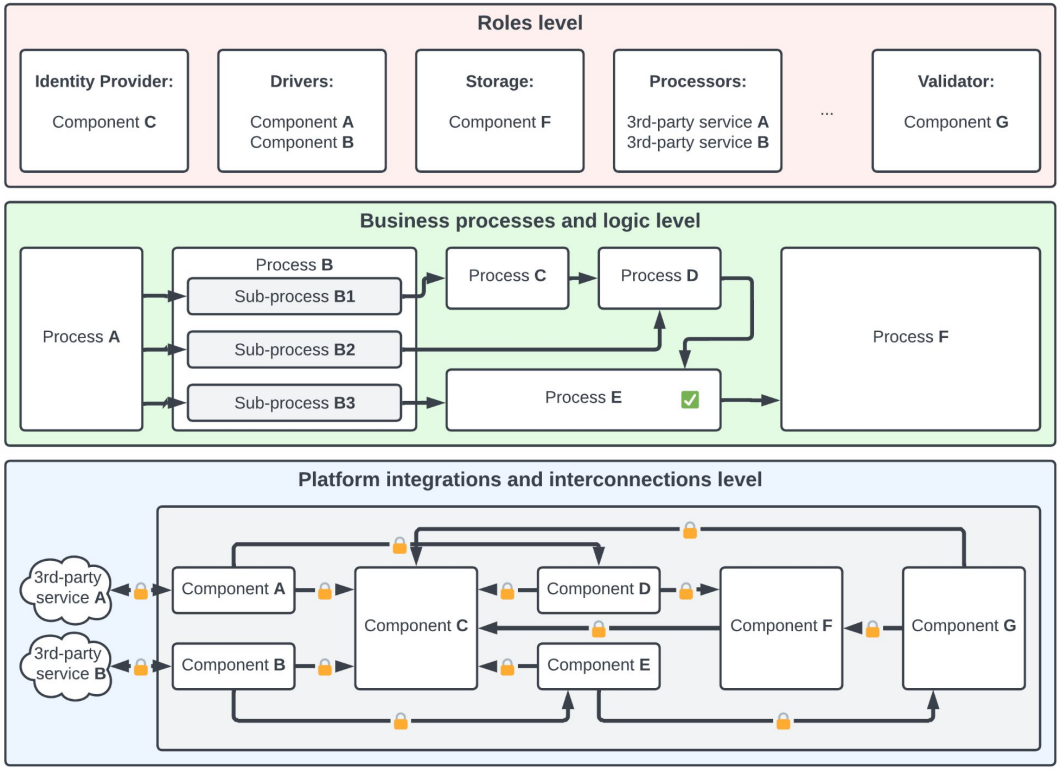
- defined
- independent
- clear

Processes must be:

- idempotent
- integral
- clean
- traceable
- verifiable
- authentic

Integrations must be:

- authorized
- trustless
- self-protected
- well-known
- controlled
- restricted
- minimum required



Where does Security Engineering stay?

Roles must be:

- defined
- independent
- clear

Clean design

Processes must be:

- idempotent
- integral
- clean
- traceable
- verifiable
- authentic

Logic and processes hardening

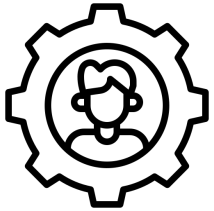
Integrations must be:

- authorized
- trustless
- self-protected
- well-known
- controlled
- restricted
- minimum required

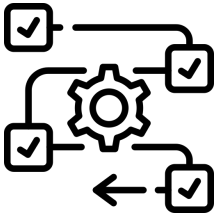
App platform and integrations hardening

What **exactly** to do? (*simplified approach*)

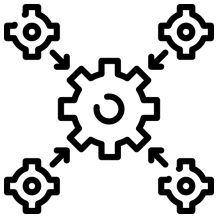
Roles:
Identify points of trust and decision making.
Unify and simplify roles.



Processes (logic):
Identify negative scenarios for decision making logic.
Put controls and verification.



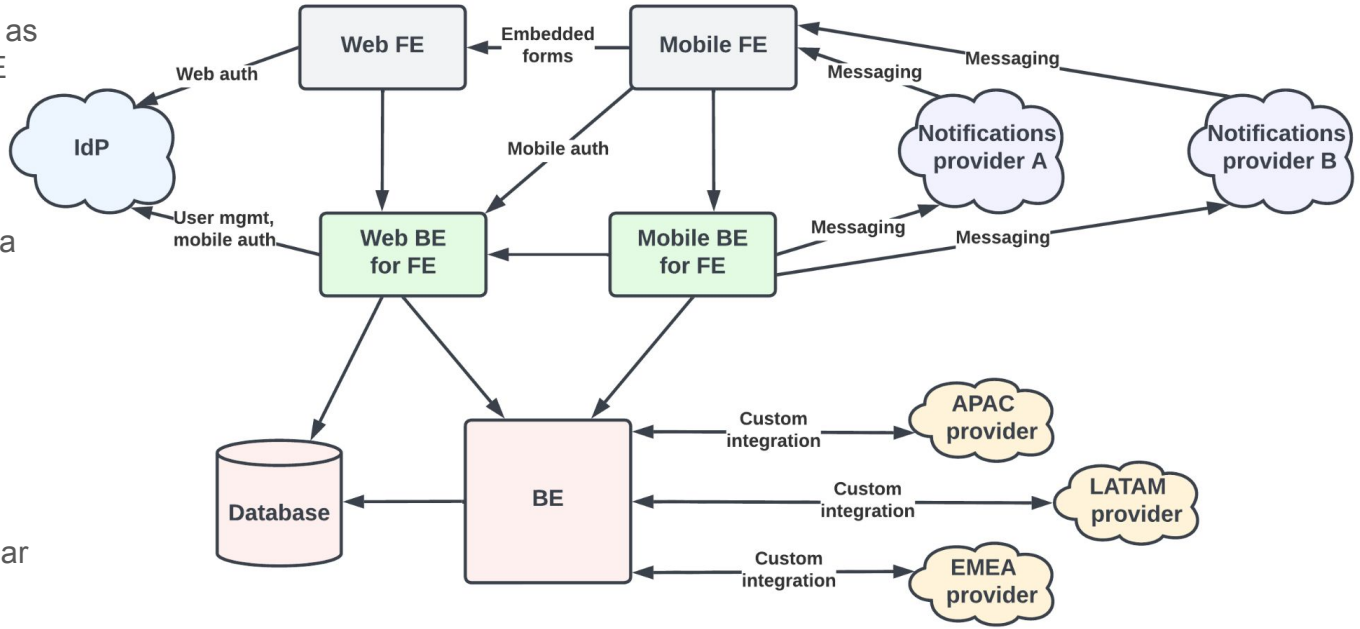
Integrations (platform):
Reduce trust points.



Example 1: Clean roles and design

dirty

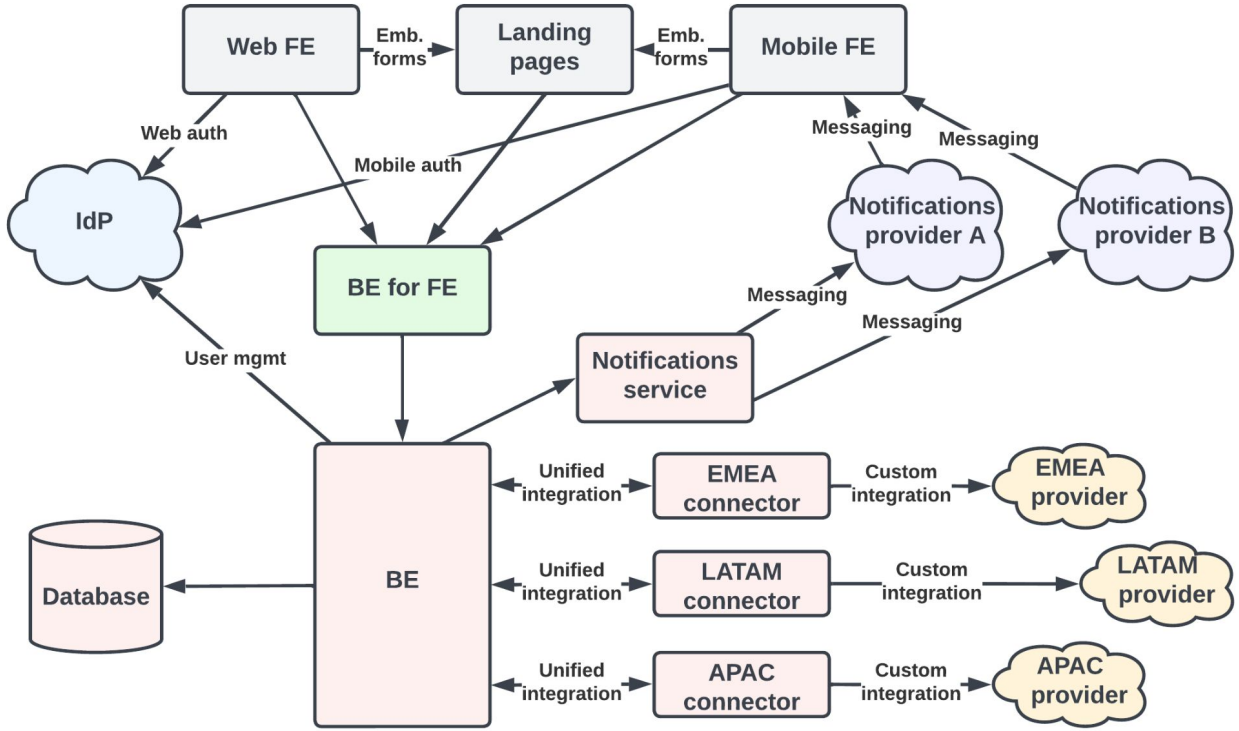
- Web BE for FE acts as an IdP for Mobile FE and an IdM for the whole app
- Mobile BE for FE middleware acts as a notifications service
- Web BE for FE middleware directly writes to the BE database
- BE handles custom integrations for similar services
- Mobile app logically depends on the Web app



Example 1: Clean roles and design

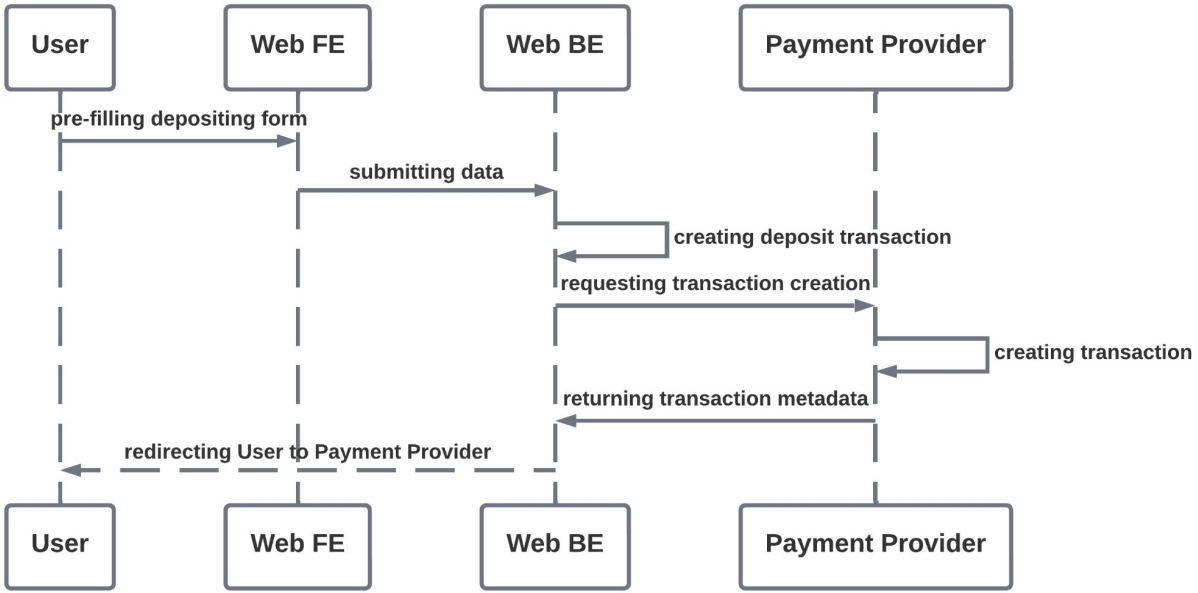
clean

- ~~Web BE for FE acts as an IdP for Mobile FE and an IdM for the whole app~~
- ~~Mobile BE for FE middleware acts as a notifications service~~
- ~~Web BE for FE middleware directly writes to the BE database~~
- ~~BE handles custom integrations for similar services~~
- ~~Mobile app logically depends on the Web app~~



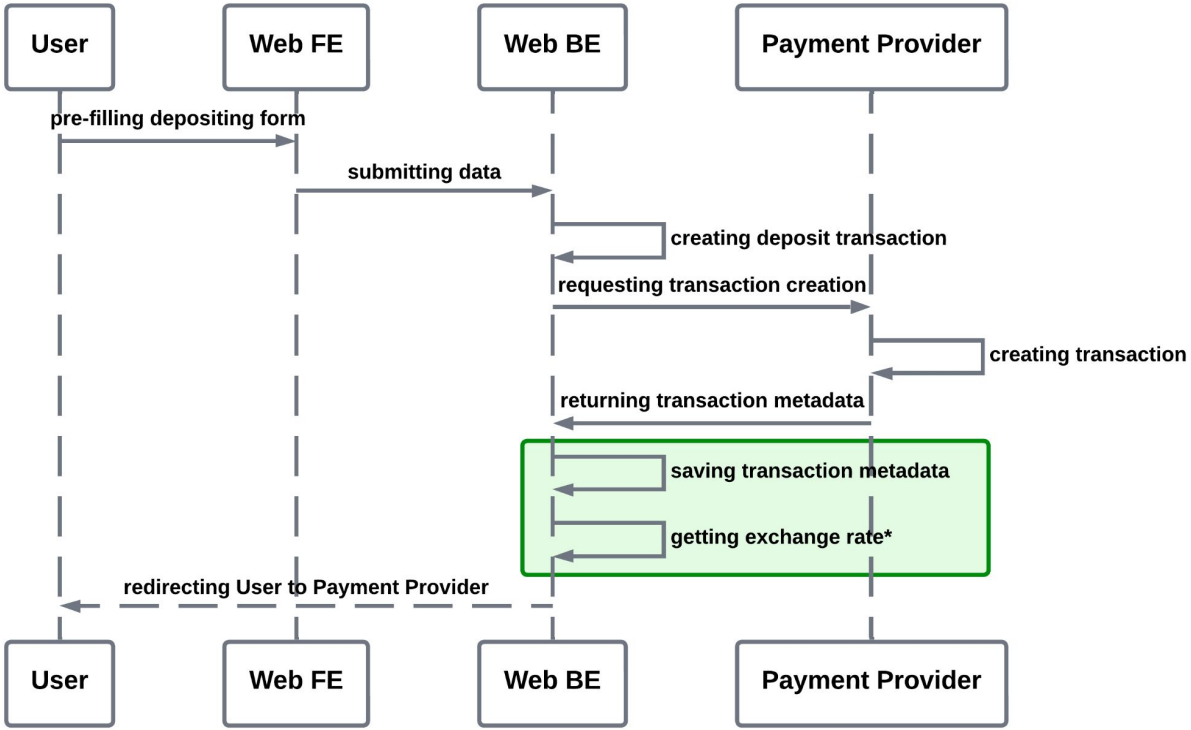
Example 2: Payment flow hardening, part 1

unprotected



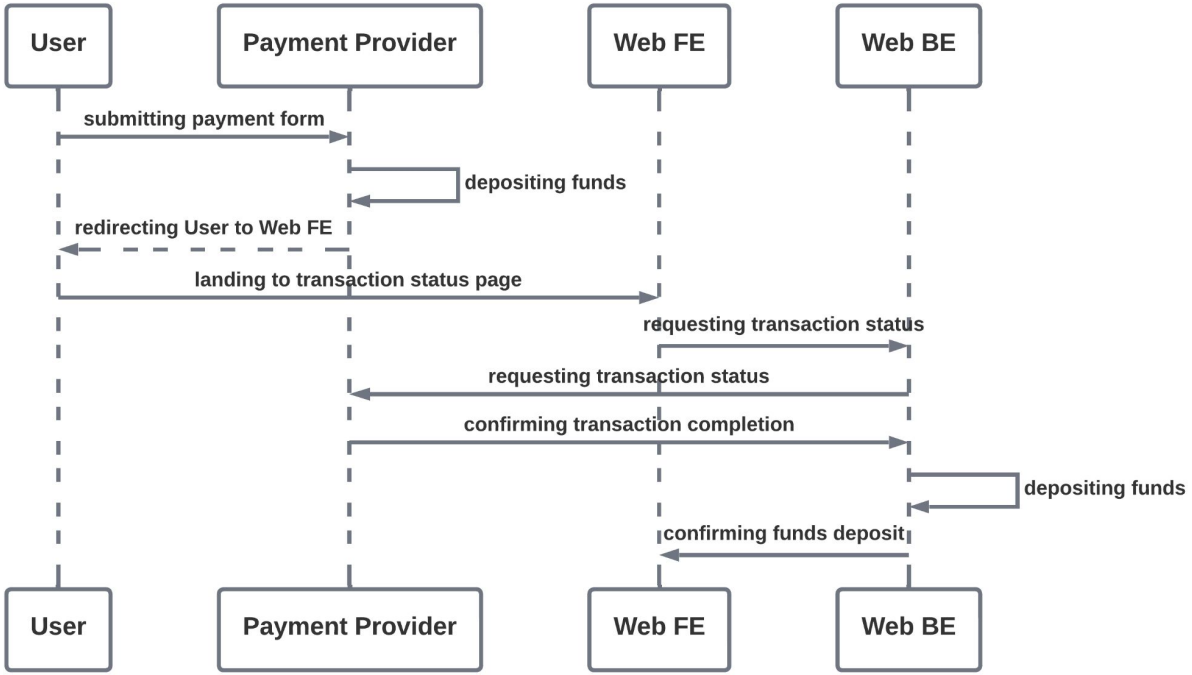
Example 2: Payment flow hardening, part 1

`protected`



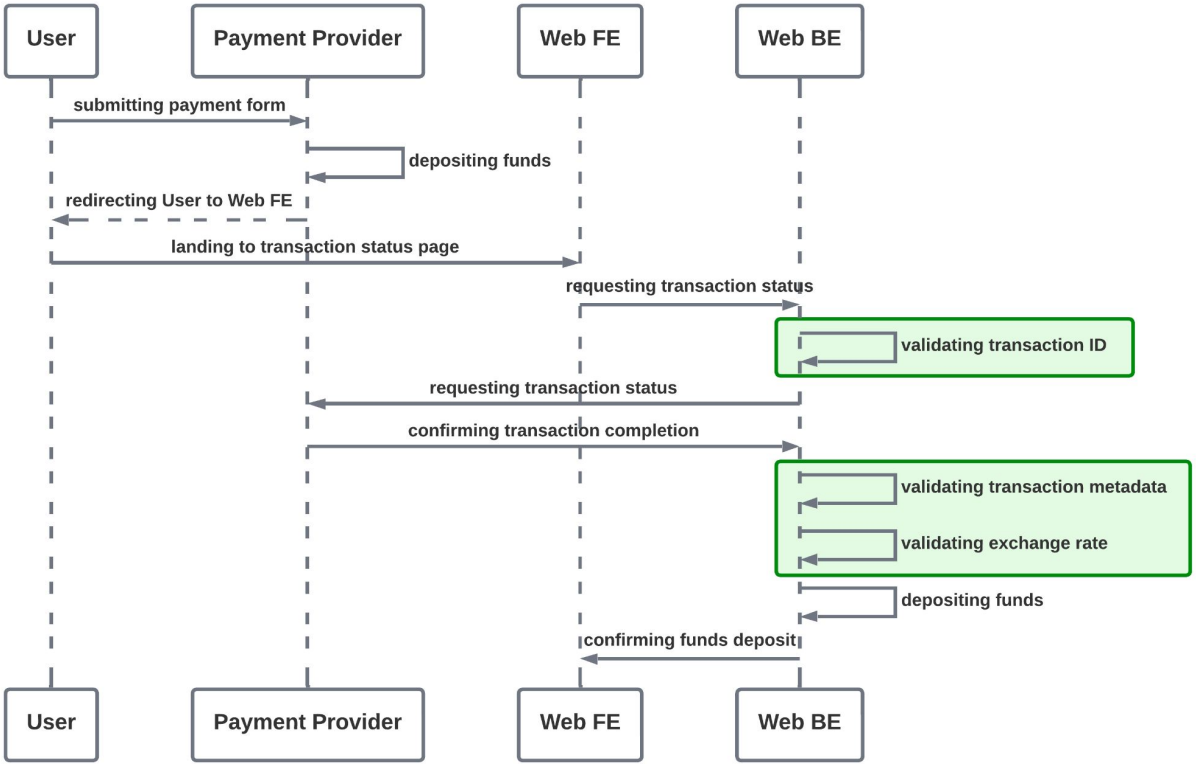
Example 2: Payment flow hardening, part 2

unprotected



Example 2: Payment flow hardening, part 2

`protected`



Example 3: App-platform hardening

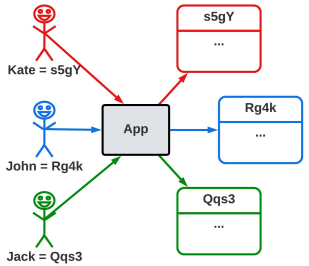
Security controls must be integral part of apps, e.g.:

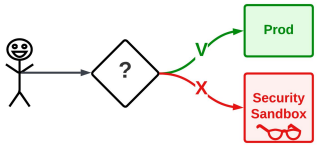
- Separating data:
  - customer per table with unique token, and/or
  - using per record encryption with unique keys
- Monitoring and sandboxing abnormal activity
- Active client app instance state control

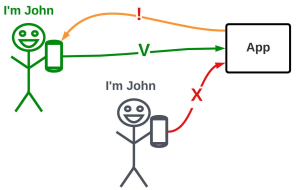
and the most obvious:

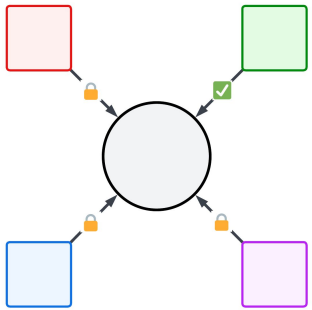
- Advanced authN and authZ

etc.



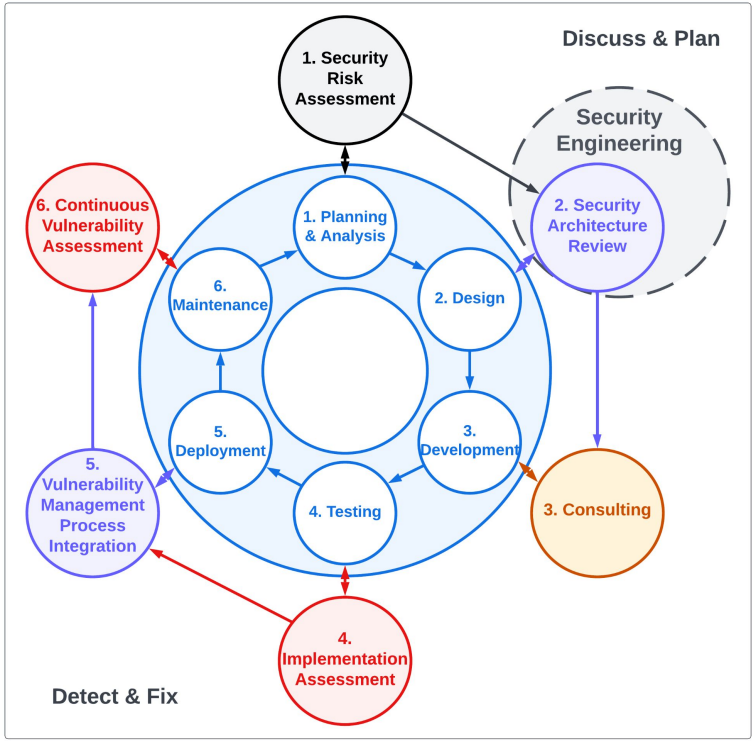






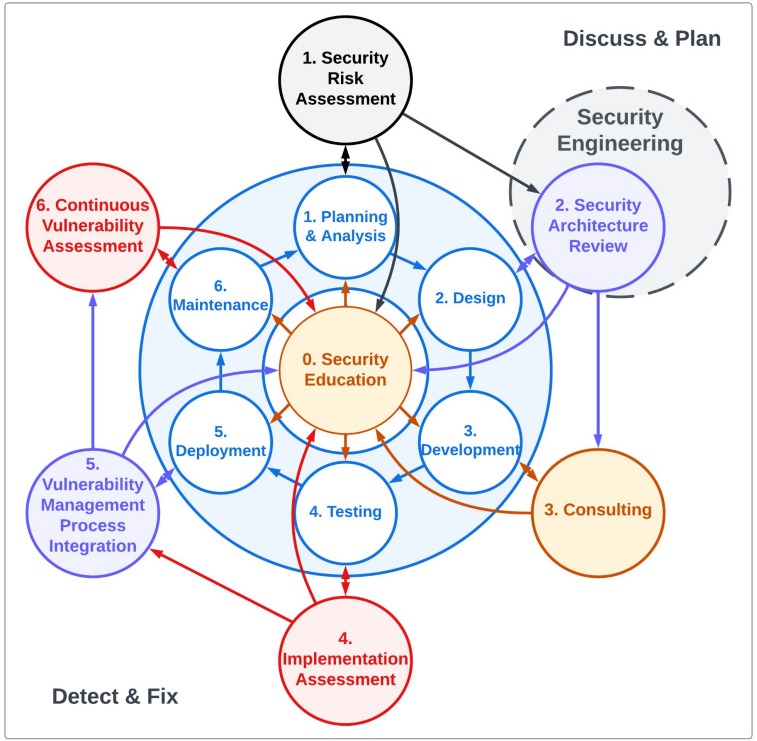
How to achieve it?

- AppSec team integration to engineering processes at early stages
- Continuous bi-directional education
- Delegation of security engineering to Software Engineering teams
- Building security controls as integral parts of products instead of post-application



How to achieve it?

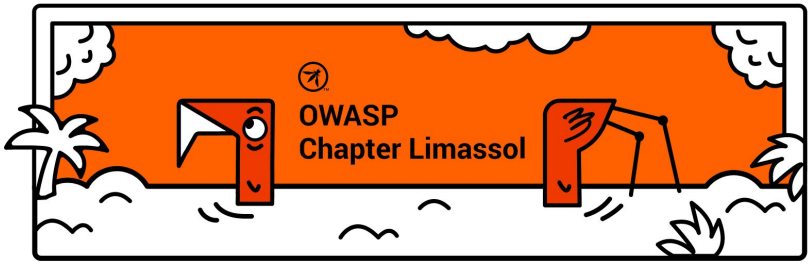
- AppSec team integration to engineering processes at early stages
- Continuous bi-directional education
- Delegation of security engineering to Software Engineering teams
- Building security controls as integral parts of products instead of post-application



Conclusions

- Keep **initial** Application Security **purpose** in mind
- You need to **work** with systems on **different layers**:
  - **application level**: app roles, platform, integrations, logic etc.
  - **system level**: infrastructure, access, business processes etc.
- **Embed security** instead of applying it, where possible:
  - **built-in** controls are usually **cheaper** and much more **effective**
  - **applied** solutions are **limited** to treating application as a whole
- **Spread** security engineering **culture** among engineers
- **Build-in** security engineering **to development** process
- **Security** practices help **healing** engineering in general

Contacts



TG: `@awetsky`
E-mail: `me@vciso.digital`
LinkedIn: `https://linkedin.com/in/anatoli-m`