gematik

CI-Pipelines mit Sicherheit:

Erfahrungen aus der E-Rezept-App

Martin Fiebig
gematik | iOS Development E-Rezept
martin.fiebig@gematik.de
mrtnfbg
mfiebig

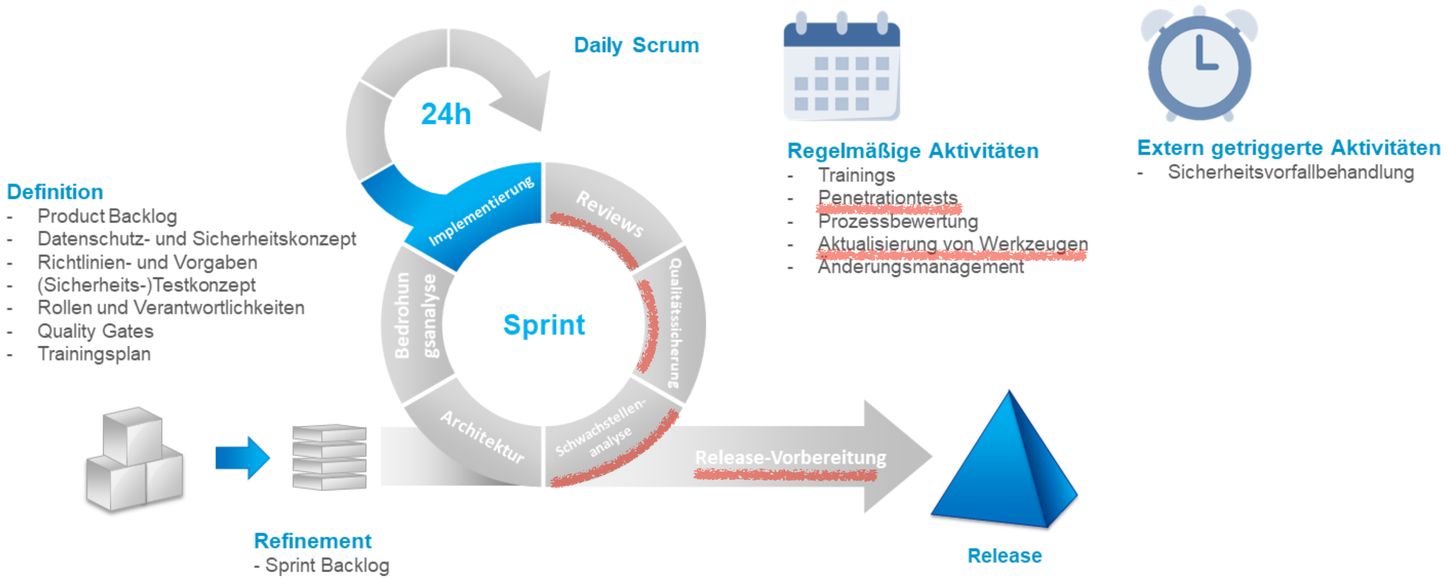
Joachim Gärtner
gematik | Android Development E-Rezept
joachim.gaertner@gematik.de
fnordlicht
fnordlicht

Agenda


Start
Motivation
CI Pipeline
Release Pipeline
Publish Pipeline
End

Motivation

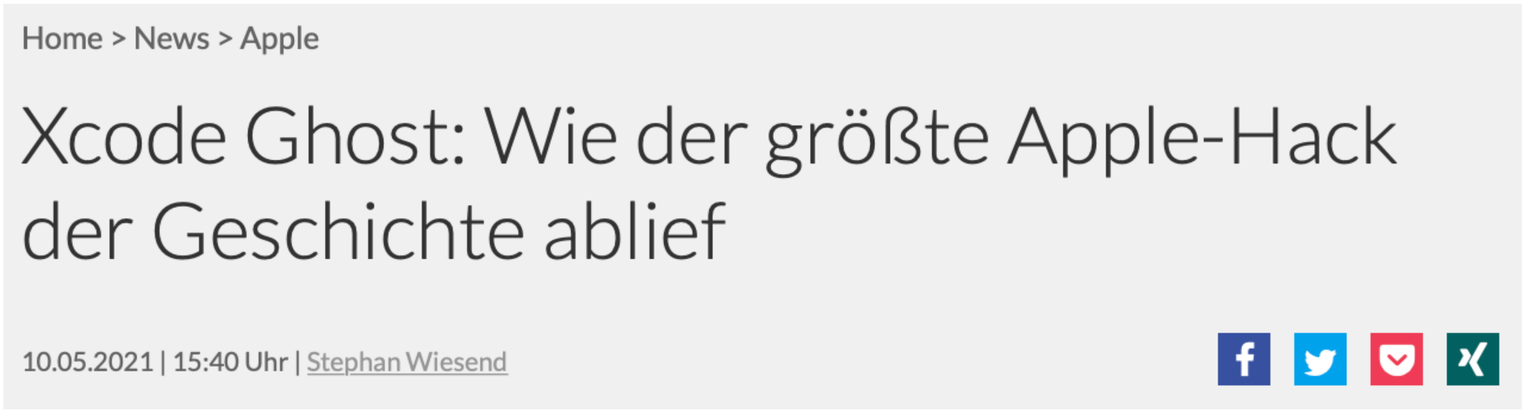
- Teil des **S**ecure **S**oftware **D**evelopment **L**ife **C**ycle



Source: gematik

Motivation

- Teil des **S**ecure **S**oftware **D**evelopment **L**ife **C**ycle
  - Application Security Risks
  - Supply Chain Attack
    - Xcode Ghost: 128 Millionen Opfer?



https://www.macwelt.de/news/Xcode-Ghost-Wie-der-groesste-Apple-Hack-der-Geschichte-ablief-11026422.html

Pipelines

fastlane

ANSIBLE

macOS

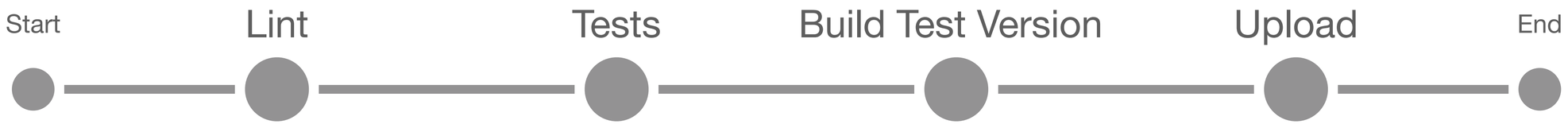
docker

CI	Every Commit
Release	Release Branches
SAST (48h SLA)	Release Branches /on demand
Source Code Publish	Manually
(System-)Integration Tests	Nightly/On Demand

CI	Every Commit
Release	Tag based
SAST (48h SLA)	Release Branches /on demand
Source Code Publish	Manually
Dev/Test Builds	Nightly/On Demand

Source Git Icons: https://dribbble.com/afnizarnur

CI Pipeline



Lint/Static Analysis

- Same style makes reading easier
  - Spot todos/issues
  - Avoid "goto fail"
- Avoid problematic patterns
  - Keep cyclic complexity low

```
...
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto ↓fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto ↓fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto ↓fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto ↓fail;
    goto ↓fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto ↓fail;

err = sslRawVerify(ctx,
                   ctx->peerPubKey,
```

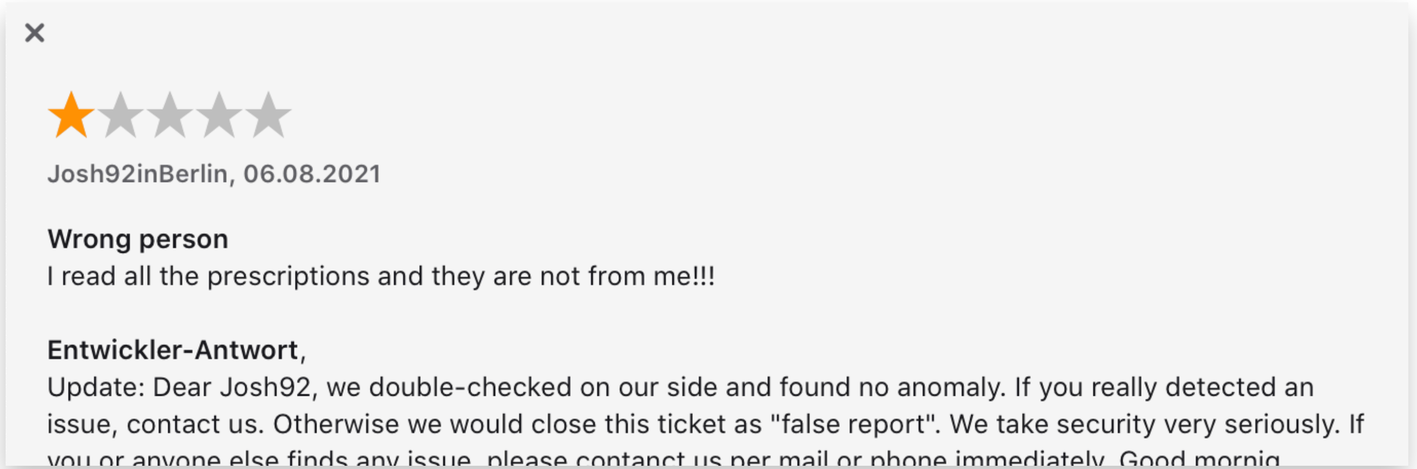
Source: https://www.synopsys.com/blogs/software-security/understanding-apple-goto-fail-vulnerability-2/

- swiftformat
- swiftlint
- swiftgen
- sourcery

- ktLint
- Detekt (geplant)
- (Sonartype)

Build Test Version

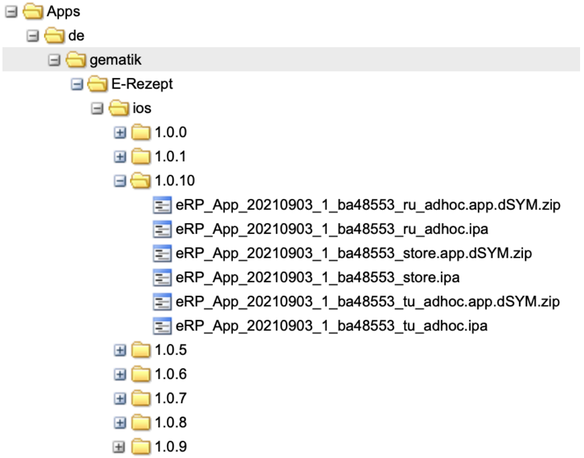
- Automated tests with simulators
- Automated testing with device cloud (attached eGK)
- Manual testing
  - Explorative
  - Pentesting
  - Alpha/Beta Tester



Release Pipeline

Archive Artifacts/Buildlog

- Reproducible builds
- Which (build)tools (+version) are used?
- Which libraries (+version) are used?



Screenshot: Artifacts within Nexus

- Gemfile.lock, Podfile.lock, Cartfile.lock
- xcversion

- build.gradle
- Dockerfile

SAST

Static Application Security Testing

- Fortify
- Sonarcube
- Gitlab CI

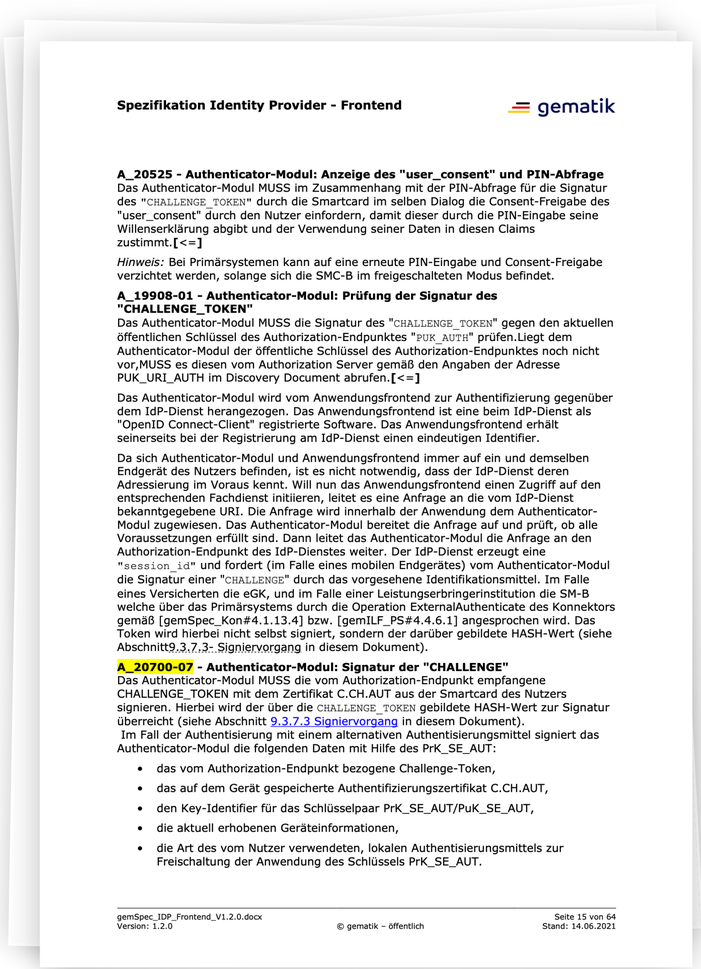
Try before you buy!

- Fortify on Demand

Source Code Publish

Build Documentation

- Implementation based on Requirements



Spezifikation Identity Provider - Frontend gematik

A_20525 - Authenticator-Modul: Anzeige des "user_consent" und PIN-Abfrage
Das Authenticator-Modul MUSS im Zusammenhang mit der PIN-Abfrage für die Signatur des "CHALLENGE_TOKEN" durch die Smartcard im selben Dialog die Consent-Freigabe des "user_consent" durch den Nutzer einfordern, damit dieser durch die PIN-Eingabe seine Willenserklärung abgibt und der Verwendung seiner Daten in diesen Claims zustimmt.**[<=]**

Hinweis: Bei Primärsystemen kann auf eine erneute PIN-Eingabe und Consent-Freigabe verzichtet werden, solange sich die SMC-B im freigeschalteten Modus befindet.

A_19908-01 - Authenticator-Modul: Prüfung der Signatur des "CHALLENGE_TOKEN"
Das Authenticator-Modul MUSS die Signatur des "CHALLENGE_TOKEN" gegen den aktuellen öffentlichen Schlüssel des Authorization-Endpunktes "PUK_AUTH" prüfen.Liegt dem Authenticator-Modul der öffentliche Schlüssel des Authorization-Endpunktes noch nicht vor,MUSS es diesen vom Authorization Server gemäß den Angaben der Adresse PUK_URI_AUTH im Discovery Document abrufen.**[<=]**

Das Authenticator-Modul wird vom Anwendungsfrontend zur Authentifizierung gegenüber dem IdP-Dienst herangezogen. Das Anwendungsfrontend ist eine beim IdP-Dienst als "OpenID Connect-Client" registrierte Software. Das Anwendungsfrontend erhält seinerseits bei der Registrierung am IdP-Dienst einen eindeutigen Identifier.

Da sich Authenticator-Modul und Anwendungsfrontend immer auf ein und demselben Endgerät des Nutzers befinden, ist es nicht notwendig, dass der IdP-Dienst deren Adressierung im Voraus kennt. Will nun das Anwendungsfrontend einen Zugriff auf den entsprechenden Fachdienst initiieren, leitet es eine Anfrage an die vom IdP-Dienst bekanntgegebene URI. Die Anfrage wird innerhalb der Anwendung dem Authenticator-Modul zugewiesen. Das Authenticator-Modul bereitet die Anfrage auf und prüft, ob alle Voraussetzungen erfüllt sind. Dann leitet das Authenticator-Modul die Anfrage an den Authorization-Endpunkt des IdP-Dienstes weiter. Der IdP-Dienst erzeugt eine "session_id" und fordert (im Falle eines mobilen Endgerätes) vom Authenticator-Modul die Signatur einer "CHALLENGE" durch das vorgesehene Identifikationsmittel. Im Falle eines Versicherten die eGK, und im Falle einer Leistungserbringerinstitution die SM-B welche über das Primärsystems durch die Operation ExternalAuthenticate des Konnektors gemäß [gemSpec_Kon#4.1.13.4] bzw. [gemILF_PS#4.4.6.1] angesprochen wird. Das Token wird hierbei nicht selbst signiert, sondern der darüber gebildete HASH-Wert (siehe Abschnitt9.3.7.3- Signiervorgang in diesem Dokument).

A_20700-07 - Authenticator-Modul: Signatur der "CHALLENGE"
Das Authenticator-Modul MUSS die vom Authorization-Endpunkt empfangene CHALLENGE_TOKEN mit dem Zertifikat C.CH.AUT aus der Smartcard des Nutzers signieren. Hierbei wird der über die CHALLENGE_TOKEN gebildete HASH-Wert zur Signatur überreicht (siehe Abschnitt 9.3.7.3 Signiervorgang in diesem Dokument).
Im Fall der Authentisierung mit einem alternativen Authentisierungsmittel signiert das Authenticator-Modul die folgenden Daten mit Hilfe des PrK_SE_AUT:

- das vom Authorization-Endpunkt bezogene Challenge-Token,
- das auf dem Gerät gespeicherte Authentifizierungszertifikat C.CH.AUT,
- den Key-Identifier für das Schlüsselpaar PrK_SE_AUT/PuK_SE_AUT,
- die aktuell erhobenen Geräteinformationen,
- die Art des vom Nutzer verwendeten, lokalen Authentisierungsmittels zur Freischaltung der Anwendung des Schlüssels PrK_SE_AUT.

gemSpec_IDP_Frontend_V1.2.0.docx Seite 15 von 64
Version: 1.2.0 © gematik – öffentlich Stand: 14.06.2021

🔗 https://fachportal.gematik.de

Build Documentation

- Implementation based on Requirements
- Referenced in Code with special "Tag"

```
321     func sign(challenge session: IDPChallengeSession) -> AnyPublisher<SignedChallenge, Failure> {
322         flatMap { secureCard in
323             secureCard
324                 // [REQ:gemSpec_IDP_Frontend:A_20700-07] C.CH.AUT
325                 // [REQ:gemF_Tokenverschlüsselung:A_20526-01] Smartcard signature
326                 // [REQ:gemF_Tokenverschlüsselung:A_20700-06] sign
327                 .readAutCertificate()
328                 .flatMap { certificate -> AnyPublisher<SignedChallenge, Swift.Error> in
329                     // [REQ:gemSpec_Krypt:A_17207] Assure only brainpoolP256r1 is used
330                     guard let alg = certificate.info.algorithm.alg else {
331                         return Fail(error: NFCSignatureProviderError.signingFailure(nil)).eraseToAnyPublis
332                     }
333                     // [REQ:gemSpec_IDP_Frontend:A_20700-07] sign with C.CH.AUT
334                     return session.sign(
335                         with: EGKSigner(card: secureCard),
336                         using: [certificate.certificate],
337                         alg: alg
338                     )
339                     .eraseToAnyPublisher()
340                 }
341                 .mapError { $0.asNFCSignatureError() }
342                 .eraseToAnyPublisher()
343         }.eraseToAnyPublisher()
344     }
345 }
```

Sources/eRpApp/Screens/CardWall/ReadCard/NFCSignatureProvider.swift#L324

Build Documentation

- Implementation based on Requirements
- Referenced in Code with special "Tag"
- Aggregate all References ...

```
610       block.each do |line|
611         # Format: [REQ:<SPEC>:<AFOS>:<SUBAFOS?>] <DESC>
612         if matches = line.match(/\[REQ:(?<SPEC>[^:]*):(?<AFOS>[^:]*)(?:(?::)(?<SUBAFOS>[^:]*))?\](?<DESC>.*)/)
613
614           register(matches, file, number, trimmedCode)
615         end

                ...

658
659     erb2_path = File.join(File.dirname(__FILE__), '..', 'Templates', 'ERB', 'requirements.md.erb')
660     erb2_out_path = erb2_path.sub('.erb','').sub('/Templates/ERB','/build/docs/generated')
661     erb2 = ERB.new(File.read(erb2_path))
```

fastlane/Fastfile#L570

Build Documentation

- Implementation based on Requirements
- Referenced in Code with special "Tag"
- Aggregate all References and create a markdown list

A_20700-07

`../Sources/IDP/DefaultSecureEnclaveSignatureProvider.swift:127`: Biometrics only, other modes currently not supported

```
// [REQ:gemSpec_IDP_Frontend:A_20700-07] Biometrics only, other modes currently not supported
amr: [
```

`../Sources/eRpApp/Screens/CardWall/ReadCard/CardWallReadCardDomain.Environment+Biometrics.swift:21`: sign with C.CH.AUT

```
// [REQ:gemSpec_IDP_Frontend:A_20700-07] sign with C.CH.AUT
return challengeSession.sign(
```

`../Sources/eRpApp/Screens/CardWall/ReadCard/CardWallReadCardDomain.Environment+Biometrics.swift:109`: C.CH.AUT

generated/requirements.md#a_20700-07

Joachim Gärtner

gematik | Android Development E-Rezept

joachim.gaertner@gematik.de

fnordlicht

fnordlicht

Martin Fiebig

gematik | iOS Development E-Rezept

martin.fiebig@gematik.de

mrtnfbg

mfiebig

Thank you!

E-Rezept App

Download on the App Store

GET IT ON Google Play

EXPLORE IT ON AppGallery

Source

gematik/E-Rezept-App-iOS

gematik/E-Rezept-App-Android