Chapitre 5
Tests non fonctionnels

1. Introduction

Sur une grosse majorité des projets rencontrés, il s'avère que les tests non fonctionnels sont éludés. À l'évocation de cette catégorie de test, un soupir sonore « *Aaaaah* » un peu gêné est en général l'unique Exigence Non Fonctionnelle (ENF ou NFR en anglais - *Non Functional Requirements*) connue de tous. Dans certains cas, on aura des tentatives telles qu'un essai de l'application sur X navigateurs différents ou une équipe spécialisée qui fait des tests de charge ou de sécurité.

Les anomalies ne proviennent des fonctionnalités qu'à hauteur de 16 % [Beizer 1994], mais **si certaines contraintes rencontrées en exploitation ne sont pas respectées, le produit ne sera qu'un prototype ou une expérience de laboratoire**. C'est pourquoi une stratégie de test ne doit pas faire l'économie de tests non fonctionnels avec une **prise en compte dès la conception**.

Exigences non fonctionnelles dans la mobilité

Dans le domaine de la mobilité, il existe une ENF connue de tous : l'énorme choix de combinaisons OS/Version/Matériel et si l'utilisation du produit sur différentes plateformes mobiles n'est pas prise en compte dès le début, on risque de développer pour iOS et de devoir reprendre tous les développements pour Android ou WinPhone !

Remarque

On pourrait multiplier dans ce livre les informations sur l'importance des ENF, mais ce ne sera que par le vécu et un partage au sein de l'équipe que la maturité émergera pour impliquer réellement ces contraintes. La ***culture est un facteur majeur*** *dans la prise en compte des ENF. Lorsqu'une ENF est bien ancrée dans celle-ci, les solutions techniques et les moyens de tests sont impliqués et budgétés sans avoir ni à réfléchir, ni à convaincre :*

- une technologie multi-plateforme est impliquée : Ionic (https://fr.wikipedia.org/wiki/Ionic_(framework)), Cordova (https://fr.wikipedia.org/wiki/Apache_Cordova), Xamarin (https://fr.wikipedia.org/wiki/Xamarin), NativeScript (https://en.wikipedia.org/wiki/NativeScript), ReactNative (http://www.reactnative.com/), Titanium (https://fr.wikipedia.org/wiki/Appcelerator_Titanium) ou Flutter (https://flutter.dev/) ;

- une ferme de serveur est montée avec OpenStf (https://openstf.io/) ou louée à un prestataire dans le Cloud comme BrowserStack (https://www.browserstack.com/) ou SauceLab (https://saucelabs.com/) pour avoir autant de combinaisons que son marché le nécessite.

On l'aura compris, **cette culture provient de contraintes opérationnelles** (voir chapitre Versant technique du test) qui imposent une **vision proche du terrain**.

2. Tests de sécurité

Depuis le 26 mai 2018, l'Europe a promulgué une loi (RGPD - voir plus bas dans ce chapitre) pour lutter contre le faible niveau de sécurité lié au traitement et à la protection des données personnelles. Cela a fait beaucoup de bruit au vu des sanctions annoncées (https://fr.wikipedia.org/wiki/R%C3%A8glement_g%C3%A9n%C3%A9ral_sur_la_protection_des_donn%C3%A9es) et n'est qu'un prétexte de plus pour mettre au centre des préoccupations les tests de sécurité déjà abordés par l'industrie des cartes bancaires [PCI-DSS], l'Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) qui a produit le « Référentiel général de sécurité » [RGS] des standards internationaux de sécurité [ISO27002] ou encore ceux liés aux sociétés cotées en bourse [SAS70].

Cette section se penche avec ambition sur un domaine souvent obscur.

2.1 Généralités sur les tests de sécurité

La sécurité est un domaine des plus vastes, car il **touche toutes les parties du produit et les personnes qui interagissent avec**. Cette configuration fait du test de sécurité :

– une activité **pluridisciplinaire** avec des Experts qui comprennent les règles, les lois, l'organisation sous-jacente, les opérations, les processus et les technologies utilisées ;

– une exploration des dangers **dans l'obscurité** avec pour seul éclairage l'expertise afin de trouver les failles avant qu'elles ne soient exploitées ;

– une activité de contrôle **perpétuel** des failles connues pour éviter qu'elles n'impactent (à nouveau) le système ;

– une tâche qui ne saurait être simple…

Astuce

Le MVP pour les tests de sécurité

*Ce sujet est très difficile à aborder, car **techniquement complexe, culturellement ambitieux et politiquement délicat**. Aussi, comme à chaque fois avec l'agilité, on tentera d'identifier le MVP de la sécurité ; ce MVP peut être complètement empirique (on fait avec les moyens et la connaissance du bord), sous-traité à un tiers compétent dans le domaine, ou pris en main dès qu'on a un peu de temps devant soi, par exemple en lisant ce chapitre pour, au moins, aborder le « SHU » de la sécurité.*

Les problèmes de sécurité ne sont pas seulement réservés au domaine de l'infrastructure. La communauté « *Common Weakness Enumeration* » (énumération des failles récurrentes) recense une liste de failles directement liées au développement avec près de 700 vulnérabilités identifiées (voir https://cwe.mitre.org/). Selon Jeffery Payne, **les failles liées au code représentent 50 % des menaces** [Payne 2016].

Astuce

Challenge « 30 jours sur les tests de sécurité »

Pour commencer l'acculturation d'une équipe aux tests de sécurité, Melissa Eaden propose un challenge de 30 jours sur ce thème avec un défi par jour pour avancer sur différents axes [Eaden 2016] :

1. Lire un blog sur la sécurité.

2. Sélectionner et lire un livre sur le test de sécurité.

3. Utiliser un outil de test de sécurité tel que ZAP (https://www.zaptest.com/) ou BurpSuite (https://fr.wikipedia.org/wiki/Burp_suite).

4. Apprendre quoique ce soit sur le scan de vulnérabilité.

5. Connaître le modèle de menace (voir par exemple le modèle « STRIDE »).

6. Explorer ces sites : Google gruyere; HackYourself First; Ticket Magpie; The Bodgelt store.

7. Apprendre une chose ou plus sur les tests de pénétration.

8. Utiliser un outil de type proxy pour observer le trafic d'une application web ou mobile.

9. Découvrir le processus et les procédures liées à l'audit de sécurité.

10. Apprendre ce qu'est le Hacking Ethique (White Hat).

11. Tenter de se représenter une posture de cybersécurité pour une application.

12. Se documenter sur le test de sécurité et établir une discussion sur le meilleur endroit du cycle de développement logiciel où on pourrait en introduire.

13. Réaliser une analyse de sécurité dans une US.

14. Faire l'ingénierie d'un plan de test incluant des tests de sécurité.

15. Écrire et partager des idées sur les tests de sécurité sur Twitter ou un blog.

16. Faire des recherches sur comment construire une « Tiger Box ».

17. Faire des recherches sur une faille de sécurité ou un hack récent.

18. Se documenter sur la sécurisation des Headers.

19. Se documenter sur les Script Kiddies et/ou les Packet monkeys.

20. Se documenter sur les attaques DoS/DDoS. Partager quelques exemples sur un réseau social.

21. Se documenter sur les vulnérabilités d'un réseau et les rechercher dans son organisation.

22. Se documenter sur la sécurité des systèmes et l'appliquer sur son organisation.

23. Répondre à la question : quel est le top 10 des menaces de sécurité cette année ?

24. Utiliser une suggestion de la check-list OWASP pour les applications web.

25. Identifier et utiliser un outil de test de sécurité pour mobile.

26. Faire un comparatif sur les réseaux sociaux des tests web et mobile.

27. Comment le BYOA (Bring Your Own Application - Amenez votre propre application) peut jouer un rôle dans la sécurité ?

28. Partager des idées de tests de sécurité pour un domaine particulier.

29. Faire des recherches sur les contraintes réglementaires de sécurité pour un domaine particulier.

30. Décrouvrir la différence entre le Hacking White Hat, Grey Hat et Black Hat.

31. BONUS : participer à un Bug Bounty.

2.2 Orientations en termes de sécurité

Pour tenter de lutter contre la menace liée aux problèmes de sécurité, plusieurs standards existent et Aleatha Shanley propose une comparaison de différents modèles [Shanley 2015] :

- BSIMM - *Building Security in Maturity Model* : propose différentes pratiques liées à la sécurité (dont les tests de pénétration), mais aucun outil ;
- ISSAF - *Information Systems Security Assessment Framework* : un framework de test de pénétration avec des processus et des outils ;
- MSF - *Metasploit Framework* : palette d'outils de tests de pénétration et de détection d'intrusion ;
- OSSTMM - *Open Source Security Testing Methodology Manual* : méthode d'audit de sécurité ;
- OWASP - *Open Web Application Security Project* : outils, guides et méthodes de travail pour les applications web qui couvrent tout le cycle de développement ;
- PTES - *Penetration Testing Execution Standard* : standard de tests de sécurité qui regroupe différentes sources dont le OWASP [Hayes 2012].

Aleatha Shanley utilise d'autres critères de comparaison :

- l'ISO25010 comme base de son *benchmark* ;
- la longévité, le nombre de révisions et la date de dernière mise à jour ;
- le facteur de lisibilité tel que le *Gunning Fog Index* - GFI (voir chapitre Facteurs de succès) pour guider le choix d'orientation pour une organisation.

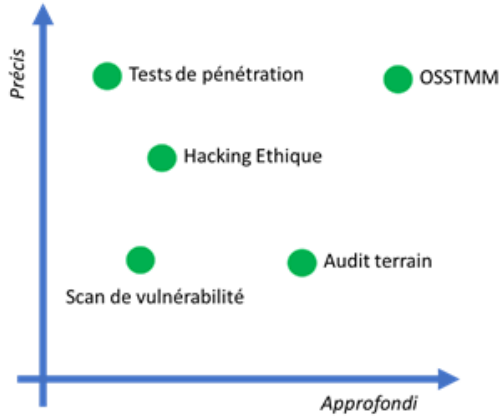

Figure 1 : Positionnement de l'OSSTMM par rapport aux pratiques usuelles de tests de sécurité [Leita 2011]

Le positionnement de l'OSSTMM semble être le meilleur, étant donné la représentation choisie ; en revanche, sur la perspective de la mise en pratique, ce standard se situe loin derrière des approches basées sur les scans de vulnérabilité comme l'OWASP ou MSF qui fournissent des outils.

Définir sa stratégie de test de sécurité

Même si Aleatha Shanley ne va pas jusqu'au bout de la démarche en montrant l'étude comparative des six standards, cela peut donner des pistes pour vous aider à choisir le vôtre ; c'est ce que fait Corrado Leita en comparant à sa façon les frameworks OSSTMM, ISSAF et NIST.

La norme ISO 27001 (voir https://fr.wikipedia.org/wiki/ISO/CEI_27001) aborde la complexité de la sécurité suivant l'approche du progrès continu, le PDCA - Prévoir/Donner corps/Contrôler/Améliorer (en anglais : « Plan/Do/Check/Act » - voir https://fr.wikipedia.org/wiki/Roue_de_Deming) avec les activités suivantes :

- *Prévoir :*
  - *on délimite le périmètre du Système de Management de la Sécurité de l'Information (SMSI) ;*
  - *on s'engage sur une politique de SMSI en fonction des enjeux métier ;*
  - *on s'engage sur une approche du risque ;*
  - *on réalise une analyse de risques ;*
- *Donner corps :*
  - *sensibilisation et communication ;*
  - *gestion de l'exploitation du SMSI ;*
  - *gestion des ressources du SMSI ;*
  - *définition des procédures de détection des incidents de sécurité ;*
  - *définition de procédures de gestion de la documentation ;*
- *Contrôler :*
  - *mesure de l'efficacité du SMSI ;*
  - *révision de l'analyse des risques ;*
  - *conduite d'audits internes ;*
  - *archivage des incidents ;*
  - *collecte et analyse des enregistrements ;*
- *Améliorer :*
  - *mise en place des améliorations identifiées ;*
  - *communication des améliorations ;*
  - *vérification de l'efficacité des améliorations.*