Cahier de recettes

Version	1.2
Date	12.12.13
Rédigé par	Boisseleau W. et Latimier M.
Relu par	-
Approuvé par	X

MISES À JOUR

Version	Date	Modifications réalisées
1.1	10.12.13	Création et complétion du cahier de recette
1.2	10.12.13	Complétion du cahier de recette

Table des matières

1 Objet

L'objet de ce cahier de recettes est de lister les tests praticables et pratiqués sur les différentes fonctionnalités liées au projet Audit des implantations SSL/TLS avant sa livraison au client. Il décrit notamment une série de scénarios précisant les démarches à suivre afin de prouver l'efficacité des différents produits. Il permet en effet de vérifier que la série de logiciels/services est conforme aux spécifications définies avec le client et qu'elles répondent efficacement au cahier des charges.

2 Terminologie et sigles utilisés

Les acronymes suivants peuvent être utilisés dans le rapport ci-présent :
- DAL : Documents d'Architecture Logicielle
- ZMAP : réfère au logiciel ZMAP.
- RC : réfère au logiciel de récupération de certificat
- SD : réfère au logiciel de suppression de doublons
- F : réfère au logiciel de calcul de GCD d'une liste d'entiers deux à deux

Des sigles sont également définis et utilisés localement pour faciliter la définition des procédures de tests.

3 Documents applicables et de référence

4 Environnement de test

4.1 Lieu

L'ensemble des procédures de tests s'effectueront en salle machine U2.2.34 de l'Université de Rouen, en compagnie d'un ou deux testeurs minimum. Quelques tests peu gourmands en temps et en mémoire, mais nécessaires à la validation des fonctionnalités du produit final, pourront être effectués sur des machines personnelles. Cependant, la majorité d'entre eux nécessiteront la monopolisation d'une machine pour plusieurs heures, voire plusieurs jours.

4.2 Machines utilisées

Nous serons amenés à utiliser des machines virtuelles lors de tests mettant en scène un réseau complexe sous différents scénarii possibles (mécanisme de pare-feu, zone (dé)militarisée, ports ouverts/fermés pour des connexions SSH, ...).

Nous avons choisi le logiciel de virtualisation de réseau Netkit pour mener à bien ces tests. Le DAL indiquera l'ensemble des machines réelles utilisables pour la plate-forme de tests et/ou de développement.

4.3 Base de données de test

Plusieurs fonctionnalités du projet seront déjà implantées et pourront servir d'appui pour nos tests (dont ZMAP ou l'algorithme de factorisation).

Nos procédures de tests se succèdent pour la plupart, les résultats des précédents tests nous serviront de faits pour les phases de tests suivantes.

4.4 Contraintes

La modification - même minime - de la partie applicative de notre projet aura pour conséquence le relancement de toutes les procédures de tests directement et indirectement concernés.

Nous aurons des contraintes de temps selon le planning prévu dans le plan de développement. Les procédures de tests doivent être fonctionnelles rapidement afin de valider l'avancement du projet.

Nos procédures de tests doivent être les plus simples possibles afin d'éviter toute perte de temps, et toutes contraintes d'espace (ce qui n'enlève en rien l'efficacité des tests).

5 Responsabilités

Latimier et Boisseleau sont les principaux testeurs sur ce projet, ils devront mener à bien la conception, l'exécution et la vérification de chacun d'entre eux. D'autres membres du projet pourront consacrer leur temps sur les tests en cas de demande explicite du chef de projet, ou durant leur temps libre (considéré comme non planifié).

Dès lors qu'un outil ou parti d'outil sujet à des procédures de test est développé, les testeurs auront la responsabilité de lancer les procédures de tests sur celui-ci afin de permettre une gestion des erreurs efficace et la plus en amont possible.

De plus, les testeurs s'appliqueront à respecter ou faire respecter les conditions de tests définies dans ce rapport.

Dans les cas ou des erreurs sont mises en évidence, l'équipe de testeurs devra d'une part analyser l'origine du problème et d'autre part communiquer aux développeurs et au chef de projet les problèmes identifiés, et, lorsque c'est possible, une proposition de solution ou de piste solutionnante.

6 Stratégie de tests

6.1 Gestion des tâches de tests

Les outils de tests devront être développés en parallèle aux développement des fonctionnalités du projet. L'écart de développement ne doit pas être trop grand, afin d'optimiser le rendement de chaque acteur.

Les procédures de tests sont les plus fines possibles, elles mettent en place individuellement un test spécifique ; ceci afin de les distribuer plus facilement lors de la répartition des tâches, et de pouvoir

gagner du temps dans nos phases de débogage.

De plus, elles devront avoir le moins d'inter-dépendance possible, pour éviter des modifications en cascade lors des modifications des procédures parentes. Lorsqu'un outil de test est fonctionnel, il est inutile d'y apporter des améliorations. Nous y reviendrons uniquement lors de modifications éventuelles des fonctionnalités du projet.

6.2 La campagne de test

Elle s'effectuera principalement sur la première partie de notre projet "audit de clefs cryptographiques", et éventuellement sur la troisième partie "analyse dynamique" selon le planning et l'attente du client (non exigé pour l'instant).

La seconde partie "analyse statique" ne nécessite pas d'exigences techniques particulières, mais plutôt une réflexion sur un problème cryptographique.

6.3 Critères d'arrêt

Une procédure de tests sera considérée comme valide lors de l'envoi des livrables fonctionnelles lui étant associée et le retour favorable du client.

7 Gestion des anomalies

Lors de nos phases de tests, l'ensemble des anomalies rencontrées devront être signalées dans un document attitré au format CSV (ou ODS). Lorsqu'une anomalie est rencontrée, nous devons calculer son taux de criticité sur le projet et le cahier des charges. Si une exigence technique du client s'en trouve affectée, elle doit être réglée au plus vite.

Une politique de prévention doit être mis en place afin d'éviter tout renouvellement, en cas de modification d'un logiciel par exemple.

8 Procédures de test

8.1 ZMAP

Terminologie.

Sont définis les acronymes suivants :
- `R` : réseau virtuel fixé représentant toutes les configurations réseau envisageables,
- `R'` : réseau virtuel altéré de `R`,
- `lA` : liste d'adresses prédéfinie de machines de `R`,
- `A1` : liste d'adresses de machines ayant des ports SSH ouverts sur `R`,
- `A2` : liste d'adresses de machines ayant des ports SSH protégés et fermés sur `R`,
- `A1'` : liste d'adresses de machines ayant des ports SSH ouverts sur `R'`,
- `A2'` : liste d'adresses de machines ayant des ports SSH protégés et fermés sur `R'`,

– `VA1` : liste d'adresses de machines attendues ayant des ports SSH ouverts sur `R`,
– `VA2` : liste d'adresses de machines attendues ayant des ports SSH protégés et fermés sur `R`,
– `VA1'` : liste d'adresses de machines attendues ayant des ports SSH ouverts sur `R'`,
– `VA2'` : liste d'adresses de machines attendues ayant des ports SSH protégés et fermés sur `R'`.

Procédures.

Objet testé : ZMAP		**Version : 1.0**		
Objectif de test : ZMAP reconnaît port O/F/N-A				
Procédure P1 : TestPortsZMAP(ZMAP,R,R',lA,VA1,VA2,VA1',VA2')				
N.	**Actions**	**Résultats attendus**	**Exig.**	**OK/NOK**
1	On lance ZMAP sur `R` avec `lA` en entrée	ZMAP retourne `A1` et `A2`. On vérifie que `A1=VA1` et `A2=VA2`.		/
2	On relance ZMAP sur `R'` avec `A2` en entrée	ZMAP retourne `A1'` et `A2'`. On vérifie que `A1'=VA1'` et `A2'=VA2'`.		/

Objet testé : ZMAP		**Version : 1.0**		
Objectif de test : Tester la portabilité du résultat de ZMAP pour l'application RC				
Procédure P2 : Portabilité(ZMAP,lA,R)				
N.	**Actions**	**Résultats attendus**	**Exig.**	**OK/NOK**
1	Sur le système Linux, on lance ZMAP sur `R` avec `lA` en entrée. On récupère A1 et A2. On relance ZMAP sur `A1` et `A2`	ZMAP s'exécute correctement sur `A1` à `A2`	P1	/
2	Même procédé sur le système Windows	ZMAP se termine correctement sur `A1` à `A2`	P1	/

8.2 RC

Terminologie.

Sont définis les acronymes suivants :
– `LAd` : liste d'adresses de machines ayant des ports ouverts générées par ZMAP,
– M_i : machine i de la `LAd`,
– C_i : certificat ou chaine de certification de la machine i selon RC,
– VC_i : certificat ou chaine de certification de la machine i selon M_i,
– C'_i : certificat ou chaine de certification de la machine i selon RC,
– VC'_i : certificat ou chaine de certification de la machine i après stockage.

Procédures.

Objet testé : RC		Version : 1.0		
Objectif de test : Établissement de connexion avec RC				
Procédure P3 : conRC(RC,lA)				
N.	**Actions**	**Résultats attendus**	**Exig.**	**OK/NOK**
1	RC lit `LAd`.	RC ne retourne pas d'erreur, il reconnaît les M_i comme valides	P2	/

Objet testé : RC		Version : 1.0		
Objectif de test : Récupération des certificats				
Procédure P4 : recCertifRC(RC,lA)				
N.	**Actions**	**Résultats attendus**	**Exig.**	**OK/NOK**
1	RC échange de certificats avec M_i.	Si M_i autorise l'échange, récupération de C_i et $C_i = VC_i$, sinon RC passe à M_{i+1}	P3	/
2	RC stocke dans une base de données le certificat s'il existe	Le certificat est stocké correctement et C_i et $C'_i = VC'_i$	P3	/

8.3 SD

Terminologie.

Sont définis les acronymes suivants :
- `B` base de données contenant des certificats uniques et des certificats identiques,
- `BA` base de données B après suppression des doublons selon SD,
- `VBA` base de données B sans doublons

Procédure.

Objet testé : SD		Version : 1.0		
Objectif de test : Tests de suppression de doublons				
Procédure P4 : supD(SD,B,VBA)				
N.	**Actions**	**Résultats attendus**	**Exig.**	**OK/NOK**
1	SD s'exécute sur B	SD s'exécute correctement et BA=VBA		/

8.4 F

Terminologie.

Sont définis les acronymes suivants :
- lA_1 : liste de taille impaire d'entiers de même taille binaire,
- lA_2 : liste de taille paire d'entiers de même taille binaire,
- $F1$: section de l'algorithme de F calculant l'arbre des produits,
- $F2$: $F - F1$ section de l'algorithme de F calculant l'arbre des restes,

- lF_1 : liste des fils de lA_1 calculée suivant $F1$ sur lA_1,
- lF_2 : liste des fils de lA_2 calculée suivant $F1$ sur lA_2,
- VlF_1 : liste des fils attendue pour lA_1,
- VlF_2 : liste des fils attendue pour lA_2,
- P : entier produit de facteurs après lancement de $F1$ sur lA_A,
- $modP1$: élément gauche après calcul du modulo de P suivant $F2$,
- $modP2$: élément droite après calcul du modulo de P suivant $F2$,
- $VmodP1$: élément gauche modulo de P suivant $F2$,
- $VmodP2$: élément droite modulo de P suivant $F2$.

Procédures.

Objet testé : F		**Version : 1.0**		
Objectif de test : Calcul des fils selon $F1$ suivant une liste				
Procédure P5 : calcF1($F1$,lA_1,lA_2,VlF_1,VlF_2)				
N.	**Actions**	**Résultats attendus**	**Exig.**	**OK/NOK**
1	On calcule $F1(lA_1) = lF_1$	$F1$ s'exécute correctement et $lF_1 = VlF_1$		/
2	On calcule $F1(lA_2) = lF_2$	$F1$ s'exécute correctement et $lF_2 = VlF_2$		/

Objet testé : F		**Version : 1.0**		
Objectif de test : Calcul des modulos fils avec $F2$ suivant P				
Procédure P6 : calcF2($F2$,P,$VmodP1$,$VmodP2$)				
N.	**Actions**	**Résultats attendus**	**Exig.**	**OK/NOK**
1	On calcule $F2(P) = (modP1, modP2)$	$F2$ s'exécute correctement et $modP1 = VmodP1$ et $modP2 = VmodP2$	P5	/