PREMIER MINISTRE

Liberté
Égalité
Fraternité

Secrétariat général de la défense et de la sécurité nationale

Agence nationale de la sécurité des systèmes d'information

Paris, le 23 juillet 2021

N° **1968/ANSSI/SDE/PSS/CCN**

Référence **: ANSSI-CSPN-NOTE-08_v1.0**

NOTE D'APPLICATION

METHODOLOGIE POUR L'EVALUATION D'APPLICATIONS MOBILES

Application **:** Dès son approbation.

Diffusion **:** Publique.

Le sous-directeur adjoint « Expertise » de l'Agence nationale de la sécurité des systèmes d'information

Sébastien KUNZ-JACQUES
[ORIGINAL SIGNE]

SÉCURITÉ CERTIFICATION

SUIVI DES MODIFICATIONS

Version	Date	Modifications
1.0	23/07/2021	Création

En application du décret n°2002-535 du 18 avril 2002 modifié, la présente note a été soumise, lors de sa création, au comité directeur de la certification, qui a donné un avis favorable.

Cette note est également soumise pour avis lors de chaque modification majeure conformément au manuel qualité du centre de certification. Les évaluations mineures ne sont pas soumises au comité directeur de la certification.

La présente note est disponible en ligne sur le site institutionnel de l'ANSSI (www.ssi.gouv.fr).

TABLE DES MATIERES

1 Objet de la note et produits visés 4
2 Terminologie 5
3 Travaux d'évaluation spécifiques 6
3.1 Exigences spécifiques pour l'évaluateur 6
3.1.1 Charge d'évaluation 6
3.1.2 Travaux spécifiques 6
3.2 Exigences spécifiques pour le commanditaire/développeur 7
3.2.1 Environnement de test 8
ANNEXE A. Références 9
ANNEXE B. Évaluation du protocole de communication 10
ANNEXE C. Conformité et résistance (menaces « typiques ») 11
ANNEXE D. Conformité à l'état de l'art 14

1 Objet de la note et produits visés

Le présent document décrit la méthode d'évaluation de la sécurité des applications mobiles pour les environnements *Android* et *iOS* dans le contexte d'une Certification de sécurité de premier niveau (CSPN). La sécurité des téléphones mobiles eux-mêmes n'est pas incluse dans cette méthodologie.

La méthodologie détaille ce qui est attendu des Centres d'évaluation de la sécurité des technologies de l'information (CESTI) (section 3.1 Exigences spécifiques pour l'évaluateur), mais également des développeurs/commanditaires (section 3.2 Exigences spécifiques pour le commanditaire/développeur).

En outre, la méthodologie impose aux CESTI de prendre en compte les points d'attention suivants :

- l'identification et unicité de l'application ;
- la protection du code ;
- la détection des anomalies ;
- la GUI sécurisée ;
- la gestion des sessions utilisateurs ;
- la gestion des données volatiles ;
- la gestion des données persistantes ;
- les communications sécurisées ;
- l'authentification utilisateur.

2 Terminologie

ADB	*Android debug bridge*, outil permettant entre autre, d'avoir un accès à la console du téléphone/émulateur *Android*.
API	*Application programming interface*, Interface de programmation.
ARM	Type d'architecture processeur, souvent présents dans les systèmes embarqués (téléphone mobile, tablettes, routeurs, etc.).
ASLR	*Address Space Layout Randomisation*, distribution aléatoire de l'espace d'adressage.
CESTI	Centre d'évaluation de la sécurité des technologies de l'information.
GUI	*Graphical user interface*, interface homme-machine.
HTTP	Protocole de communication client-serveur.
IPC	*Inter-process communication*, communication interprocessus *Android*.
Manifest	Le fichier AndroidManifest.xml permet de décrire les applications *Android*. Il permet entre autre de lister : - le nom du package de l'application ; - tous les composants de l'application ; - toutes les permissions de l'application ; - les versions de l'API Android ; - les bibliothèques utilisées par l'application.
Offuscation	Technique de protection de code source rendant la compréhension plus difficile par un humain.
PIE	*Position-independent executable*.
RAM	*Random access memory*, mémoire à accès non séquentiel.
SSL	*Secure socket Layer*, protocole de sécurisation des échanges sur le réseau.
TLS	*Transport Security layer*, sécurité de la couche de transport.

3 Travaux d'évaluation spécifiques

L'évaluation d'une application doit permettre de vérifier que les fonctions de sécurité définies dans la cible de sécurité sont bien implémentées et que leur implémentation est suffisamment robuste pour résister à un attaquant dans les conditions fixées par la CSPN.

Une analyse sécuritaire d'une application mobile au titre de la CSPN comprendra obligatoirement :

- une phase d'analyse statique de l'application afin d'optimiser le temps d'évaluation. Cette analyse se fera avec la mise à disposition du code source de l'application à l'évaluateur ;
- une phase d'analyse dynamique, correspondant aux tests d'intrusion de l'application, afin notamment de confirmer les hypothèses de l'analyse statique.

3.1 Exigences spécifiques pour l'évaluateur

Certaines parties de la méthodologie CSPN [CSPN-CER-P-01] ont été reprises et modifiées afin d'adapter le contenu par rapport au contexte d'évaluation de sécurité des applications mobiles.

3.1.1 *Charge d'évaluation*

La charge de travail prévue pour la réalisation de l'évaluation sécuritaire d'une application est de :

- **au minimum 15 hommes*jours** pour l'évaluation de l'application (revue de code source et analyse dynamique) ;
- **10 hommes*jours** pour l'éventuelle analyse cryptographique.

Certaines applications implémentent leurs propres protocoles de communication. La validation sécuritaire de ces protocoles ne fait pas partie du périmètre de la charge estimée ci-dessus.

La charge supplémentaire prévue pour l'évaluation des protocoles de communication propriétaires est de :

- **10 hommes*jours** pour chaque protocole spécifique (voir ANNEXE B Évaluation du protocole de communication).

Le serveur dont l'application mobile peut dépendre n'est pas pris en compte dans cette méthodologie, mais l'évaluation de ce serveur est à prendre en considération.

3.1.2 *Travaux spécifiques*

Eu égard à l'état de l'art, des travaux spécifiques aux évaluations d'application mobile ont été définies, afin de cadrer les activités de l'évaluateur. Ils correspondent à des compléments des phases 4 à 6 définies par la procédure [CSPN-CER-P-02]. Les autres phases d'évaluation ne comportent pas de spécificités propres aux applications mobiles.

Phase	Travaux spécifiques
Phase 4 – Analyse de la conformité – revue du code source	En plus de l'analyse de la conformité et la revue du code source, il est important que d'analyser le ou les protocoles de communication entre l'application et le serveur, décrit dans l'ANNEXE B Évaluation du protocole de communication.
Phase 5 – Analyse de la conformité – tests du produit	Cette analyse doit se conformer aux exigences de l'ANNEXE C Conformité et résistance (menaces « typiques »).
Phase 6 - Résistance des mécanismes/fonctions	Cette analyse doit se conformer aux exigences de l'ANNEXE C Conformité et résistance (menaces « typiques »).
Phase 7 – Analyse de vulnérabilité (intrinsèque, de construction, d'exploitation, etc.)	Cette analyse doit se conformer aux exigences de l'ANNEXE D Conformité à l'état de l'art.

L'évaluateur est libre de mélanger les approches présentées en ANNEXE C et en ANNEXE D (menée par les risques vs menée par les bugs) de la façon qui lui paraîtra optimale.

Dans le rapport d'évaluation, l'évaluateur décrira les hypothèses techniques et l'environnement de tests qui ont été utilisés pour réaliser l'analyse dynamique. Par exemple, il sera précisé si des versions dégradées de l'application ont été utilisées ou si des outils spécifiques ont été développés.

3.2 Exigences spécifiques pour le commanditaire/développeur

Les éléments suivants sont obligatoires pour l'évaluation, à savoir :

Éléments requis	Description
Fourniture du code source de l'application	Le commanditaire/développeur s'engage à fournir l'application ainsi que son code source à l'évaluateur.
Configuration de l'environnement	Dans le cas où l'application communique avec un ou plusieurs serveurs, le commanditaire/développeur s'engage à ne pas effectuer de modification de la configuration de ces derniers durant toute la phase d'évaluation, et à fournir un identifiant unique à cet environnement.
Environnement fonctionnel	Le commanditaire/développeur s'engage à fournir tous les comptes nécessaires à la bonne utilisation de l'application (administration, utilisateurs, etc.) avant la phase de démarrage de l'évaluation.
Environnement matériel	Si l'application évaluée nécessite un matériel spécifique pour son fonctionnement alors il doit être fourni par le commanditaire/développeur dès la date de début effective de l'évaluation (cette liste sera établie avant le début d'évaluation). Ci-dessous une liste d'exemples de ce type de matériel : - un téléphone dans la configuration spécifique prévue par le développeur ; - un terminal de paiement pour effectuer des transactions ; - une carte de crédit.

Éléments requis	Description
Environnement logiciel	Selon la nature des protections implémentées dans l'application et afin d'optimiser le temps total d'évaluation, le commanditaire/développeur devra, si l'évaluateur en fait la demande, fournir[1] : - le fichier contenant la chaîne de compilation utilisée pour la génération de l'application évaluée afin de permettre à l'évaluateur de vérifier la façon dont les bibliothèques externes sont intégrées ; - dans le cas d'une offuscation du code de l'application, le fichier généré après la compilation permettant de tracer le lien entre les classes, méthodes et variables originales avec le résultat de l'offuscation. Ce fichier permettra à l'évaluateur d'une part de préciser le niveau de protection appliqué, et d'autre part de pouvoir reconstituer la pile d'exécution lors de l'identification d'un bogue ; - une version non offusquée de l'application ; - une version de l'application avec les mécanismes de sécurité désactivés. Ci-dessous une liste d'exemples de ce type de mécanismes : o la détection des accès avec des droits privilégiés, o la détection de l'activation du canal de débogage, o la détection de l'émulateur *Android* ; - une version de débogage de l'application.

3.2.1 *Environnement de test*

Afin d'effectuer des tests de validation sur une plateforme mobile, différentes solutions permettent de créer un environnement de test : l'émulateur de *Google*, mais aussi le simulateur d'*Apple* ainsi que des équipements physiques tels que téléphones, tablettes ou un quelconque équipement développé spécifiquement sur processeur *ARM*.

Néanmoins, la cible de sécurité devant préciser une plateforme de référence, il est obligatoire que le l'évaluateur teste l'application sur cette plateforme.

Les CESTI au titre de la CSPN sont libres d'utiliser leurs propres outils et environnements. Toutefois, une couverture des catégories suivantes doit être assurée par l'environnement durant l'évaluation. Il s'agit de :

- l'utilisation d'ADB ou l'équivalent pour la validation de certaines fonctions de sécurité ;
- l'analyse du binaire de l'application ;
- l'analyse statique du code source et du contenu de l'application ;
- l'analyse des logs ;
- les captures de trafic des communications ;
- l'analyse des fichiers locaux et des bases de données manipulés par l'application ;
- l'analyse de la mémoire du téléphone (un émulateur peut également être utilisé pour réaliser cette analyse) ;
- l'analyse des bibliothèques externes utilisées par l'application.

[1] Cette liste sera établie avant le début d'évaluation.

ANNEXE A. Références

Référence	Document
[CSPN-CER-P-01]	Certification de sécurité de premier niveau des produits des technologies de l'information, référence ANSSI-CSPN-CER-P-01, version en vigueur.
[CSPN-CER-P-02]	Critères pour l'évaluation en vue d'une certification de sécurité de premier niveau, référence ANSSI-CSPN-CER-P-02, version en vigueur.
[ANSSI_TLS]	Recommandations de sécurité relatives à TLS, référence SDE-NT-35/ANSSI/SDE/NP.
[RGS]	Référentiel général de sécurité : - Annexe B1 Mécanismes cryptographiques ; - Annexe B2 Gestion des clés cryptographiques ; - Annexe B3 Authentification.
[GUIDES]	Règles de sécurité : - https://developer.android.com/training/articles/security-tips.html - https://developer.apple.com/documentation/security Règles de sécurité et conception : - https://developer.android.com/google/play/billing/billing_best_practices.html OWASP Mobile Security Project : - https://www.owasp.org/index.php/OWASP_Mobile_Project. Guide API : - https://developer.android.com/guide/components/index.html - https://developer.apple.com/documentation/ - [ANSSI_DAT] : Recommandations de sécurité relatives aux ordiphones, référence DAT-NT-010/ANSSI/SDE/NP.

La plupart de ces documents peuvent être consultés et téléchargés depuis le site de l'ANSSI (www.ssi.gouv.fr).

ANNEXE B. Évaluation du protocole de communication

L'objectif est de disposer d'un avis d'expert sur le(s) protocole(s) de communication entre l'application et les serveurs.

Cette partie couvre non seulement l'établissement et l'implémentation des communications sécurisées, mais aussi tout niveau de protection additionnel (c'est-à-dire, ajouté au chiffrement classique SSL/TLS). Le document [ANSSI_TLS] peut ainsi être utilisé comme référence concernant les suites cryptographiques (*ciphersuites*) recommandées.

ANNEXE C. Conformité et résistance (menaces « typiques »)

Le but de l'analyse de la conformité et de la résistance de l'application est de vérifier :

- la conformité de l'implémentation de l'application par rapport à l'état de l'art et par rapport aux documents du développeur (si l'évaluateur dispose de cette documentation). Il est convenu que cette analyse puisse s'appuyer sur des outils automatiques ou semi-automatiques d'analyse de code ;
- la résistance des fonctions de sécurité. Il est convenu que la partie « dynamique » de l'analyse s'effectue en utilisant une plateforme de tests d'intrusion instrumentée dans cette optique.

Le Tableau 1 précise les travaux attendus pour les menaces « typiques » et les fonctions de sécurité associées, que l'on s'attendrait à voir pour une application mobile.

Tableau 1- Menaces « typiques » à couvrir lors de l'analyse

Si la cible envisage une menace de type…	… il est recommandé de chercher dans le code…	… et de tester…
Usurpation identité de l'application	L'évaluateur doit s'assurer de l'efficacité des fonctions permettant l'identification de l'application et l'unicité de son identifiant. L'identification couvre l'association entre un utilisateur, un téléphone et l'accès à un service. L'unicité concerne les protections contre le clonage de l'application par ces mécanismes. L'évaluateur est amené à étudier, par exemple, comment cette identification est faite, sur quels éléments de l'équipement mobile elle s'appuie, quelle transformation (empreinte) est appliquée, etc. Le but final est d'estimer si l'identification et l'unicité peuvent être remises en cause, par exemple par un clonage.	L'évaluateur doit vérifier la facilité, ou la difficulté de compromission de l'identification et l'unicité de l'application, en fonction des éléments sur lesquels l'application s'appuie.
Altération du contexte d'exécution de l'application	L'évaluateur doit vérifier : - où la protection anti *jailbreak* éventuelle est faite dans le code ; - l'utilisation des bonnes pratiques pour l'implémentation de la fonction ; - les fichiers ou binaires permettant de détecter un possible *jailbreak/rooting* ; - quand les appels à cette fonction sont réalisés ; - etc.	L'évaluateur doit s'assurer de : - l'efficacité de la fonction ; - l'appel effectif à la fonction ; - etc.

Si la cible envisage une menace de type…	… il est recommandé de chercher dans le code…	… et de tester…
Récupération de biens sensibles dans le code source	Si protection de code il y a, l'évaluateur doit s'assurer que les mesures mises en place protègent efficacement les biens contre un attaquant. Pour cela, il doit s'assurer que des mesures contre la rétro-ingénierie sont en place. Il s'agit également d'étudier que des biens ne sont pas facilement accessibles dans le code (par exemple, des biens confidentiels codés en dur dans le code). L'évaluateur vérifie que l'application se protège en s'appuyant sur des mécanismes de la plateforme pour sécuriser le stockage de l'information ou des échanges. L'évaluateur fera une analyse de la complexité d'offuscation du code. La revue des fichiers de configuration de l'offuscateur (ex : *ProGuard*) fait partie du travail de l'évaluateur.	L'évaluateur doit vérifier que la protection de code implémentée est efficace.
Compromission des entrées utilisateur	L'objectif de l'évaluateur est ici de s'assurer que les biens qui sont saisis par l'utilisateur, et qui ont des besoins en sécurité, sont entrés au moyen de composants graphiques répondant aux critères de sécurité du bien. Les composants graphiques concernent autant la partie affichée par l'application que la partie saisie par l'utilisateur.	L'évaluateur doit vérifier que les composants graphiques utilisés par l'application permettent de protéger efficacement les biens sensibles.
Mauvaise gestion des sessions utilisateurs	L'évaluateur doit s'assurer que la session utilisateur est correctement gérée par l'application. Pour cela, l'étude du code source doit permettre de faire ressortir toute mauvaise gestion telle que, l'accès à une interface nécessitant une authentification sans qu'une authentification utilisateur n'ait eu lieu.	L'évaluateur doit tenter de compromettre les sessions utilisateurs pour, soit réaliser les actions dans le cas où une authentification utilisateur serait nécessaire, ou pour accéder à des informations confidentielles nécessitant une authentification.
Récupération de biens sensibles dans la mémoire volatile	L'évaluateur doit s'assurer que les biens sensibles manipulés par l'application ne sont pas compromis ou modifiés par une mauvaise gestion des données volatiles. Pour cela, l'évaluateur sera amené à étudier l'utilisation adéquate d'objet ou de méthode permettant la gestion du besoin de sécurité.	L'évaluateur doit tenter de récupérer des biens sensibles en mémoire volatile.
Récupération de biens dans la	L'évaluateur doit s'assurer que les sensibles biens manipulés par	L'évaluateur doit s'assurer qu'une fois l'application installée, qu'elle

Si la cible envisage une menace de type…	… il est recommandé de chercher dans le code…	… et de tester…
mémoire persistante	l'application, ne sont pas compromis ou modifiés par une mauvaise gestion des données persistantes. Pour cela, l'évaluateur sera amené à étudier l'utilisation de méthodes offertes par la plateforme pour sécuriser le stockage ou l'utilisation de cryptographie pour rendre la donnée inintelligible.	soit exécutée ou non, les sensibles données persistantes sont effectivement sécurisées. Pour cela, il devra tenter d'accéder aux données persistantes de l'application.
Compromission ou altération des communications	L'évaluateur doit s'assurer que les biens sensibles transmis à l'extérieur du produit évalué sont effectivement protégés durant cette communication. Pour cela l'évaluateur sera amené à étudier : - l'utilisation correcte de la cryptographie ; - l'utilisation correcte de clés cryptographiques (taille de clé) ; - le choix algorithmique ; - les librairies utilisées ; - etc.	L'évaluateur doit s'assurer que les communications ne sont pas faites sans protections. Il doit également s'assurer de la robustesse de ces protections.
Usurpation de l'utilisateur	L'évaluateur doit s'assurer qu'une authentification est effectuée avant d'ouvrir une session utilisateur, ou un lien de communication sécurisée.	L'évaluateur doit s'assurer de la robustesse du mécanisme d'authentification. Il doit également s'assurer qu'il n'y a un moyen pour contourner ce mécanisme.

ANNEXE D. Conformité à l'état de l'art

Cette section décrit des points de vérification à tester, afin de s'assurer d'une adhérence minimale à l'état de l'art. Ces points pourront, dans une version ultérieure, être présentés sous forme de checklist qui devra être suivie par l'évaluateur. Les points suivants s'appliquent pour certains à *Android* et d'autres à *iOS*. Enfin, ces points peuvent parfois être indifféremment testés par revue de code source ou par analyse dynamique.

Tableau 2 - Points techniques à couvrir lors de l'analyse de la résistance des mécanismes

Thème	Elément à vérifier	Action
ASLR	Activation	Vérification avec *otool* de l'activation de PIE.
BDD	Analyse	Analyser les bases de données de l'application.
Code	*Best Practice*	Respect des bonnes pratiques pour la plateforme concernée.
Code	Offuscation	Analyse de la complexité d'offuscation, revue de la configuration de l'offuscateur.
Code	Chargement dynamique de code	Vérifier que l'application ne charge pas du code extérieur (*DexClassLoader* par exemple) à l'apk (carte SD, autre application, filesystem).
Code	*Best Practice*	Identifier : - les biens et la façon dont ils sont stockés (système de fichiers, chiffrement avant stockage, utilisation du trousseau de l'OS, etc.) ; - les bibliothèques utilisées et vérifier si elles sont à jour ; - tous les points d'entrée, y compris les possibles IPC ; - les erreurs de configuration ; - si l'application contient du code mort ; - si les exceptions sont correctement gérées ; - les possibles injections de commandes ; - les biens codés en dur dans le code.
Code	*Best Practice*	- Rechercher les fichiers manipulés par l'application tout au long de son cycle de vie ; - Surveiller les activités de journalisation ; - Analyser la mémoire (RAM, flash) ; - Déboguer l'application ; - Tirer profit des erreurs de configuration précédemment identifiées ; - Accéder aux caches de la plateforme avec pour but de retrouver les biens ; - Tester les interfaces de l'application avec l'intention de contourner des fonctions de sécurité de l'application (par exemple l'authentification utilisateur).

Thème	Elément à vérifier	Action
Communication	Protocole sécurisé	Utilisation de protocole sécurisé (HTTPS à la place de HTTP par exemple).
Communication	Certificats	Validation des certificats.
Communication	*Sniff*	Récupération des trames réseaux.
Communication	Proxy	Utilisation d'un proxy pour pouvoir modifier à la volée les informations échangées.
Composant *Android*	Composants	Lister les composants *Android* utilisés par l'application (*Activity*, *Service*, *Content Provider*, *Broadcast receiver*).
Cryptographie	Génération de clé	Utilisation conforme au [RGS].
Cryptographie	Gestion de clé	Utilisation conforme au [RGS].
Débogage	Analyse	Déboguer l'application pour inspecter les objets en mémoire, les fonctions et les méthodes, remplacer des variables et des méthodes pendant l'exécution de l'application.
Exploitation	Analyse	Reprendre les vulnérabilités ou les erreurs de codage permettant une exploitation de l'application.
Filesystem	Cache	Récupération du cache (clavier, UI).
Filesystem	Fichiers	Etudier le contenu du dossier de l'application (*Data at Rest*).
IPC	Exported Attribute	L'attribut exported doit être mis à false.
IPC	Assainissement	L'application assainit-elle les IPC ?
IPC	*Intent*	Lister les intents filtrés par l'application.
IPC	*Intent*	Contourner les protections en appelant directement les composants.
Log	Non utilisation	L'application ne produit pas de fichier de journalisation.
Permission	Moindre privilèges	L'application ne fait pas appel à des permissions non nécessaires.
Permission	Protection composants	Les composants *Android* sont protégés par des permissions.
Permission	Niveau de permission	Les permissions utilisées pour protéger les composants sont de type Signature.
WebView	JavaScript	Activation de *JavaScript* dans *WebView*.