Application Security Verification Standard 4.0.1 (fr)

Final

Mars 2019

Table of Contents

Frontispice

À propos de la norme

Le référentiel de vérification de la sécurité des applications est une liste d'exigences ou de tests de sécurité des applications qui peut être utilisée par les architectes, les développeurs, les testeurs, les professionnels de la sécurité, les fournisseurs d'outils et les utilisateurs pour définir, construire, tester et vérifier des applications sécurisées.

Copyright et licence

Version 4.0.1, mars 2019

Chefs de projet

- Andrew van der Stock
- Daniel Cuthbert
- Jim Manico
- Josh C Grossman
- Mark Burnett

Collaborateurs et relecteurs

- Osama Elnaggar
- Erlend Oftedal
- Serg Belkommen
- David Johansson
- Tonimir Kisasondi
- Ron Perris
- Jason Axley
- Abhay Bhargav
- Benedikt Bauer
- Elar Lang
- ScriptingXSS
- Philippe De Ryck
- Grog's Axle
- Marco Schnüriger
- Jacob Salassi
- Glenn ten Cate
- Anthony Weems
- bschach
- javixeneize
- Dan Cornell

- hello7s
- Lewis Ardern
- Jim Newman
- Stuart Gunter
- Geoff Baskwill
- Talargoni
- Ståle Pettersen
- Kelby Ludwig
- Jason Morrow
- Rogan Dawes
- Daniël Geerts

Le référentiel de vérification de la sécurité des applications repose sur les épaules des personnes concernées, de ASVS 1.0 en 2008 à 3.0 en 2016. Une grande partie de la structure et des éléments de vérification qui sont encore dans l'ASVS aujourd'hui ont été écrits à l'origine par Mike Boberski, Jeff Williams et Dave Wichers, mais il y a beaucoup plus de contributeurs. Merci à tous ceux qui y ont participé précédemment. Pour une liste complète de tous ceux qui ont contribué aux versions précédentes, veuillez consulter chaque version antérieure.

S'il manque un crédit dans la liste des crédits 4.0 ci-dessus, veuillez contacter vanderaj@owasp.org ou enregistrer un ticket sur GitHub pour être reconnu dans les futures mises à jour 4.x.

Préface

Bienvenue dans la version 4.0 du référentiel de vérification de la sécurité des applications (ASVS). L'ASVS est un effort communautaire visant à établir un cadre d'exigences et de contrôles de sécurité qui se concentre sur la définition des contrôles de sécurité fonctionnels et non fonctionnels requis lors de la conception, du développement et du test d'applications et de services web modernes.

La version 4.0.2 est la deuxième mise à jour mineure de la v4.0 destinée à corriger les erreurs d'orthographe et à rendre les exigences plus claires sans modifier les exigences ou ajouter/supprimer des exigences.

L'ASVS v4.0 est l'aboutissement des efforts de la communauté et des réactions de l'industrie au cours de la dernière décennie. Nous avons tenté de faciliter l'adoption de l'ASVS pour une variété de cas d'utilisation différents tout au long du cycle de vie du développement de logiciels sécurisés.

Nous pensons qu'il n'y aura très probablement jamais un accord à 100% sur le contenu de n'importe quel standard d'application web, y compris l'ASVS. L'analyse des risques est toujours subjective dans une certaine mesure, ce qui constitue un défi lorsque l'on tente de généraliser dans une norme unique. Cependant, nous espérons que les dernières mises à jour effectuées dans cette version sont un pas dans la bonne direction, et qu'elles améliorent les concepts introduits dans cette norme industrielle essentielle.

Quoi de neuf dans la version 4.0

Le changement le plus important dans cette version est l'adoption des directives NIST 800-63-3 sur l'identité numérique, introduisant des contrôles d'authentification modernes, basés sur des preuves et avancés. Bien que nous nous attendions à un certain recul sur l'alignement avec une norme d'authentification avancée, nous estimons qu'il est essentiel que les normes soient alignées, principalement lorsqu'une autre norme de sécurité des applications bien considérée est fondée sur des preuves.

Les normes de sécurité de l'information devraient essayer de minimiser le nombre d'exigences uniques, afin que les organisations qui s'y conforment n'aient pas à décider de contrôles concurrents ou incompatibles. Le Top 10 2017 de l'OWASP et maintenant la norme de vérification de la sécurité des applications de l'OWASP sont désormais alignés sur la norme NIST 800-63 pour l'authentification et la gestion des sessions. Nous encourageons les autres organismes de normalisation à travailler avec nous, le NIST et d'autres organismes pour parvenir à un ensemble généralement accepté de contrôles de sécurité des applications afin de maximiser la sécurité et de minimiser les coûts de mise en conformité.

ASVS 4.0 a été entièrement renuméroté du début à la fin. Le nouveau schéma de numérotation nous a permis de combler les lacunes de chapitres disparus depuis longtemps, et de segmenter les chapitres plus longs afin de minimiser le nombre de contrôles auxquels un développeur ou une équipe doit se conformer. Par exemple, si une application n'utilise pas JWT, toute la section sur JWT dans la gestion des sessions n'est pas applicable.

La nouveauté de la version 4.0 est une mise en correspondance complète avec le Common Weakness Enumeration (CWE), l'une des demandes de fonctionnalités les plus souvent souhaitées que nous avons eues au cours de la dernière décennie. Le mappage CWE permet aux fabricants d'outils et à ceux qui utilisent un logiciel de gestion des vulnérabilités de faire correspondre les résultats d'autres outils et des versions ASVS précédentes à la version 4.0 et aux versions ultérieures. Pour faire place à l'entrée CWE, nous avons dû retirer la colonne "Depuis", qui, comme nous l'avons complètement renumérotée, est moins logique que dans les versions précédentes de l'ASVS. Tous les éléments de l'ASVS ne sont pas associés à une CWE, et comme la CWE comporte de nombreux doublons, nous avons essayé d'utiliser la correspondance la plus courante plutôt que la plus proche. Les contrôles de vérification ne peuvent pas toujours être mis en correspondance avec des faiblesses équivalentes. Nous nous félicitons des discussions en cours avec la communauté du CWE et, plus généralement, avec le secteur de la sécurité de l'information, en vue de combler cette lacune.

Nous nous sommes efforcés de satisfaire, voire de dépasser, les exigences relatives au Top 10 de l'OWASP 2017 et aux contrôles proactifs de l'OWASP 2018. Comme le Top 10 2017 de l'OWASP est le strict minimum pour éviter toute négligence, nous avons délibérément fait tous les contrôles de niveau 1, sauf les contrôles spécifiques de journalisation des exigences du Top 10, ce qui permet aux adoptants du Top 10 de l'OWASP de passer plus facilement à une norme de sécurité réelle.

Nous avons voulu faire en sorte que l'ASVS 4.0 Niveau 1 soit un sur-ensemble complet des sections 6.5 de la norme PCI DSS 3.2.1, pour la conception d'applications, le codage, les tests, les examens de code sécurisé et les tests de pénétration. Cela a nécessité de couvrir les débordements de mémoire tampon et les opérations de mémoire non sécurisées dans la version 5, et les drapeaux de compilation liés à la mémoire non sécurisée dans la version 14, en plus des exigences de vérification des applications et des services web déjà en vigueur dans le secteur.

Nous avons achevé le passage de l'ASVS des contrôles monolithiques côté serveur uniquement, à la fourniture de contrôles de sécurité pour toutes les applications et API modernes. À l'époque de la programmation fonctionnelle, des API sans serveur, des mobiles, du cloud, des conteneurs, des CI/CD et des DevSecOps, de la fédération et bien d'autres choses encore, nous ne pouvons pas continuer à ignorer l'architecture moderne des applications. Les applications modernes sont conçues de manière très différente de celles construites lors de la sortie du premier ASVS en 2009. L'ASVS doit toujours être tournée vers l'avenir afin de fournir des conseils judicieux à notre public principal, les développeurs. Nous avons clarifié ou supprimé toute exigence qui suppose que les applications sont exécutées sur des systèmes appartenant à une seule organisation.

En raison de la taille de l'ASVS 4.0, ainsi que de notre désir de devenir l'ASVS de référence pour tous les autres efforts de l'ASVS, nous avons retiré la section mobile, en faveur de la norme MASVS (Mobile Application Security Verification Standard). L'appendice "Internet of Things" apparaîtra dans un futur IoT ASVS du projet "Internet of Things" de l'OWASP. Nous avons inclus un premier aperçu de l'IoT ASVS dans l'annexe C. Nous remercions l'équipe mobile de l'OWASP et l'équipe du projet IoT de l'OWASP pour leur soutien à l'ASVS, et nous nous réjouissons de travailler avec elles à l'avenir pour fournir des normes complémentaires.

Enfin, nous avons supprimé et retiré les contrôles ayant moins d'impact. Avec le temps, l'ASVS a commencé à être un ensemble complet de contrôles, mais tous les contrôles ne sont pas égaux pour produire des logiciels sécurisés. Cet effort visant à éliminer les éléments à faible impact pourrait aller plus loin. Dans une prochaine édition de l'ASVS, le système de notation des faiblesses communes (CWSS) permettra de hiérarchiser davantage les contrôles qui sont vraiment importants et ceux qui devraient être retirés.

À partir de la version 4.0, l'ASVS se concentrera uniquement sur le fait d'être le principal standard de services et d'applications web, couvrant l'architecture traditionnelle et moderne des applications, ainsi que les pratiques de sécurité agiles et la culture DevSecOps.

Utiliser l'ASVS

L'ASVS a deux objectifs principaux :

- aider les organisations à développer et à maintenir des applications sécurisées.
- permettre aux fournisseurs de services de sécurité, aux vendeurs d'outils de sécurité et aux consommateurs d'aligner leurs exigences et leurs offres.

Niveaux de vérification de la sécurité des applications

La norme de vérification de la sécurité des applications définit trois niveaux de vérification de la sécurité, chacun d'eux étant plus approfondi.

- Le niveau 1 de l'ASVS est destiné aux niveaux d'assurance faibles, et peut être testé via des tests d'intrusions classiques.
- Le niveau 2 de l'ASVS est destiné aux applications qui contiennent des données sensibles, ce qui nécessite une protection et constitue le niveau recommandé pour la plupart des applications
- Le niveau 3 de l'ASVS est destiné aux applications les plus critiques - les applications qui effectuent des transactions de grande valeur, qui contiennent des données médicales sensibles, ou toute application qui requiert le plus haut niveau de confiance.

Chaque niveau ASVS contient une liste d'exigences de sécurité. Chacune de ces exigences peut également être mise en correspondance avec des caractéristiques et des capacités de sécurité spécifiques qui doivent être intégrées dans les logiciels par les développeurs.

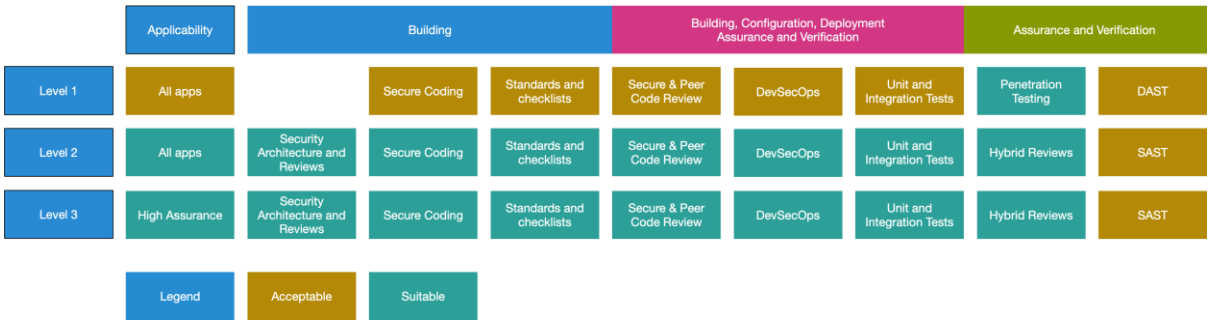

Figure 1 - Niveaux de vérification de la sécurité des applications OWASP 4.0

Le niveau 1 est le seul qui soit entièrement adéquat pour des tests d'intrusions fait par des humains. Tous les autres niveaux nécessitent l'accès à la documentation, au code source, à la configuration et aux personnes impliquées dans le processus de développement. Cependant, même si le niveau 1 permet de réaliser des tests de "boîte noire" (pas de documentation et pas de source), ce n'est pas une activité d'assurance efficace et doit être activement découragée. Les attaquants malveillants ont beaucoup de temps, la plupart des tests d'intrusions sont terminés en quelques semaines. Les défenseurs doivent mettre en place des contrôles de sécurité, protéger, trouver et résoudre toutes les faiblesses, et détecter et répondre aux acteurs malveillants dans un délai raisonnable. Les acteurs malveillants disposent essentiellement d'un temps infini et n'ont besoin que d'une seule défense poreuse, d'une seule faiblesse ou d'une détection manquante pour réussir. Les tests de la boîte noire, souvent effectués en fin de développement, rapidement ou pas du tout, sont totalement incapables de faire face à cette asymétrie.

Au cours des 30 dernières années, les tests en boîte noire ont prouvé à maintes reprises qu'ils passaient à côté de problèmes de sécurité critiques qui ont directement conduit à des violations de plus en plus massives. Nous encourageons vivement l'utilisation d'un large éventail de mesures d'assurance et de vérification de la sécurité, notamment le remplacement des tests d'intrusions par des tests d'intrusions (hybrides) de niveau 1 basés sur le code source, avec un accès complet aux développeurs et à la documentation tout au long du processus de développement. Les régulateurs financiers ne tolèrent pas les audits financiers externes sans accès aux livres, aux échantillons de transactions ou aux personnes effectuant les contrôles. L'industrie et les gouvernements doivent exiger le même niveau de transparence dans le domaine du génie logiciel.

Nous encourageons fortement l'utilisation d'outils de sécurité, mais dans le cadre du processus de développement lui-même, tels que les outils DAST et SAST utilisés en permanence par le pipeline de construction pour trouver facilement des problèmes de sécurité qui ne devraient jamais être présents.

Les outils automatisés et les scanners en ligne ne permettent pas de réaliser plus de la moitié des contrôles ASVS sans assistance humaine. Si une automatisation complète des tests pour chaque construction est nécessaire, on utilise alors une combinaison de tests unitaires et d'intégration personnalisés, ainsi que des scans en ligne initiés par la construction. Les failles de la logique métier et les tests de contrôle d'accès ne sont possibles qu'avec l'aide d'une personne. Ces tests doivent être transformés en tests unitaires et d'intégration.

Comment utiliser ce document

L'une des meilleures façons d'utiliser le référentiel de vérification de la sécurité des applications est de l'utiliser comme plan directeur pour créer une liste de contrôle de codage sécurisé spécifique à votre application, plate-forme ou organisation. En adaptant l'ASVS à vos cas d'utilisation, vous pourrez mieux vous concentrer sur les exigences de sécurité les plus importantes pour vos projets et vos environnements.

Niveau 1 - Premières étapes, automatisé, ou vue de l'ensemble du portefeuille

Une application atteint le niveau 1 de l'ASVS si elle se défend de manière adéquate contre les vulnérabilités de sécurité des applications qui sont faciles à découvrir, et qui figurent dans le Top 10 de l'OWASP et d'autres listes de contrôle similaires.

Le niveau 1 est le strict minimum que toutes les applications devraient s'efforcer d'atteindre. Il est également utile comme première étape dans un effort en plusieurs phases ou lorsque les applications ne stockent pas ou ne traitent pas de données sensibles et n'ont donc pas besoin des contrôles plus rigoureux du niveau 2 ou 3. Les contrôles de niveau 1 peuvent être vérifiés soit automatiquement par des outils, soit simplement manuellement sans accès au code source. Nous considérons que le niveau 1 est le minimum requis pour toutes les applications.

Les menaces pour l'application proviendront très probablement d'attaquants qui utilisent des techniques simples et peu exigeantes pour identifier des vulnérabilités faciles à trouver et à exploiter. En revanche, un attaquant déterminé dépensera une énergie considérable pour cibler spécifiquement l'application. Si les données traitées par votre application ont une valeur élevée, vous voudrez rarement vous arrêter à un examen de niveau 1.

Niveau 2 - La plupart des demandes

Une application atteint le niveau 2 (ou norme) de l'ASVS si elle se défend adéquatement contre la plupart des risques associés aux logiciels d'aujourd'hui.

Le niveau 2 garantit que les contrôles de sécurité sont en place, efficaces et utilisés au sein de l'application. Le niveau 2 est généralement approprié pour les applications qui traitent des transactions interentreprises importantes, y compris celles qui traitent des informations sur les soins de santé, mettent en œuvre des fonctions critiques ou sensibles, traitent d'autres actifs sensibles, ou les secteurs où l'intégrité est une facette essentielle pour protéger leur activité, comme le secteur des jeux pour contrecarrer les tricheurs et les pirates.

Les menaces pesant sur les applications de niveau 2 seront généralement des attaquants compétents et motivés qui se concentrent sur des cibles spécifiques en utilisant des outils et des techniques très pratiques et efficaces pour découvrir et exploiter les faiblesses des applications.

Niveau 3 - Haute valeur, haute assurance ou haute sécurité

Le niveau 3 de l'ASVS est le plus haut niveau de vérification au sein de l'ASVS. Ce niveau est généralement réservé aux applications qui nécessitent des niveaux de vérification de sécurité importants, comme celles qui peuvent se trouver dans les domaines de l'armée, de la santé et de la sécurité, des infrastructures critiques, etc.

Les organisations peuvent exiger le niveau 3 de l'ASVS pour les applications qui remplissent des fonctions critiques, où une défaillance pourrait avoir un impact significatif sur les opérations de l'organisation, et même sur sa capacité de survie. Des exemples de conseils sur l'application de l'ASVS niveau 3 sont fournis ci-dessous. Une application atteint le niveau 3 ASVS (ou avancé) si elle se défend de manière adéquate contre les

vulnérabilités de sécurité des applications avancées et si elle démontre également les principes d'une bonne conception de la sécurité.

Une application au niveau 3 ASVS nécessite une analyse plus approfondie de l'architecture, du code et des tests que tous les autres niveaux. Une application sécurisée est modulaire de manière significative (pour faciliter la résilience, l'évolutivité et, surtout, les couches de sécurité), et chaque module (séparé par une connexion réseau et/ou une instance physique) prend en charge ses propres responsabilités en matière de sécurité (défense en profondeur), qui doivent être correctement documentés. Les responsabilités comprennent des contrôles pour garantir la confidentialité (par exemple, le cryptage), l'intégrité (par exemple, les transactions, la validation des entrées), la disponibilité (par exemple, gérer la charge avec élégance), l'authentification (y compris entre les systèmes), la non-répudiation, l'autorisation et l'audit (journalisation).

Application de l'ASVS en pratique

Les différentes menaces ont des motivations différentes. Certains secteurs d'activité disposent d'atouts uniques en matière d'information et de technologie et ont des exigences de conformité réglementaire spécifiques à leur domaine.

Les organisations sont fortement encouragées à examiner en profondeur leurs caractéristiques de risque uniques en fonction de la nature de leur activité, et à déterminer le niveau ASVS approprié en fonction de ce risque et des exigences commerciales.

Évaluation et certification

Position de l'OWASP sur les certifications et marques de confiance ASVS

L'OWASP, en tant qu'organisation à but non lucratif neutre, ne certifie actuellement aucun vendeur, vérificateur ou logiciel.

Toutes ces assertions d'assurance, marques de confiance ou certifications ne sont pas officiellement vérifiées, enregistrées ou certifiées par l'OWASP, de sorte qu'une organisation qui s'appuie sur ce point de vue doit être prudente quant à la confiance placée dans une tierce partie ou une marque de confiance revendiquant la certification ASVS.

Cela ne devrait pas empêcher les organisations d'offrir de tels services d'assurance, tant qu'elles ne revendiquent pas la certification officielle de l'OWASP.

Directives pour les organismes de certification

La norme de vérification de la sécurité des applications peut être utilisée comme une vérification à livre ouvert de l'application, comprenant un accès ouvert et sans entrave aux ressources clés telles que les architectes et les développeurs, la documentation du projet, le code source, l'accès authentifié aux systèmes de test (y compris l'accès à un ou plusieurs comptes dans chaque rôle), en particulier pour les vérifications de L2 et L3.

Historiquement, les tests de pénétration et les examens de code sécurisé ont inclus des questions "par exception" - c'est-à-dire que seuls les tests échoués apparaissent dans le rapport final. Un organisme de certification doit inclure dans tout rapport la portée de la vérification (en particulier si un élément clé est hors de portée, comme l'authentification du SSO), un résumé des résultats de la vérification, y compris les tests réussis et les tests échoués, avec des indications claires sur la manière de résoudre les tests échoués.

Certaines exigences de vérification peuvent ne pas être applicables à l'application testée. Par exemple, si vous fournissez à vos clients une API de couche de service sans état sans implémentation client, de nombreuses exigences de la gestion de session V3 ne sont pas directement applicables. Dans de tels cas, un organisme de certification peut toujours prétendre à une conformité totale aux ASVS, mais doit clairement indiquer dans tout rapport une raison de non-applicabilité de ces exigences de vérification exclues.

La conservation de documents de travail détaillés, de captures d'écran ou de films, de scripts permettant d'exploiter un problème de manière fiable et répétée, ainsi que d'enregistrements électroniques des tests, tels que l'interception de journaux de proxy et de notes associées comme une liste de nettoyage, est considérée comme une pratique standard de l'industrie et peut être vraiment utile comme preuve des résultats pour les développeurs les plus douteux. Il ne suffit pas de faire fonctionner un outil et de signaler les échecs ; cela ne prouve pas (du tout) que tous les problèmes à un niveau de certification ont été testés et éprouvés de manière approfondie. En cas de litige, il doit y avoir suffisamment de preuves d'assurance pour démontrer que chaque exigence vérifiée a bien été testée.

Méthode de test

Les organismes certificateurs sont libres de choisir la ou les méthodes d'essai appropriées, mais doivent les indiquer dans un rapport.

Selon l'application testée et l'exigence de vérification, différentes méthodes de test peuvent être utilisées pour obtenir une confiance similaire dans les résultats. Par exemple, la validation de l'efficacité des mécanismes de vérification des entrées d'une application peut être analysée soit par un test d'intrusion manuel, soit par des analyses du code source.

Le rôle des outils de test de sécurité automatisés

L'utilisation d'outils automatisés de test de pénétration est encouragée afin d'assurer une couverture aussi large que possible.

Il n'est pas possible d'effectuer une vérification complète de l'ASVS en utilisant uniquement des outils de test de pénétration automatisés. Alors qu'une grande majorité des exigences de la L1 peuvent être réalisées à l'aide de tests automatisés, la majorité globale des exigences ne se prête pas à l'utilisation de tests d'intrusion automatisés.

Veuillez noter que les limites entre les tests automatisés et manuels se sont estompées au fur et à mesure que le secteur de la sécurité des applications a évolué. Les outils automatisés sont souvent réglés manuellement par des experts et les testeurs manuels utilisent souvent une grande variété d'outils automatisés.

Le rôle des tests de pénétration

Dans la version 4.0, nous avons décidé de rendre la L1 entièrement testable par pénétration sans accès au code source, à la documentation ou aux développeurs. Deux éléments d'enregistrement, qui doivent être conformes au Top 10 2017 A10 de l'OWASP, nécessiteront des entretiens, des captures d'écran ou d'autres collectes de preuves, tout comme dans le Top 10 2017 de l'OWASP. Cependant, les tests sans accès aux informations nécessaires ne constituent pas une méthode idéale de vérification de la sécurité, car ils laissent de côté la possibilité d'examiner la source, d'identifier les menaces et les contrôles manquants, et de réaliser un test beaucoup plus approfondi dans un délai plus court.

Dans la mesure du possible, l'accès aux développeurs, à la documentation, au code et l'accès à une application de test avec des données de non-production, est nécessaire lors de l'exécution d'une évaluation de niveau 2 ou 3. Les tests de pénétration effectués à ces niveaux nécessitent ce niveau d'accès, que nous appelons "examens hybrides" ou "tests de pénétration hybrides".

Autres utilisations de l'ASVS

En plus de servir à l'évaluation de la sécurité d'une application, nous avons identifié plusieurs autres potentielles utilisations de l'ASVS.

Guide pour les tests unitaires et d'intégration automatisés

L'ASVS est conçu pour être hautement testable, à la seule exception des exigences en matière d'architecture et de code malveillant. Grâce à des tests unitaires et d'intégration qui permettent de détecter des variantes et des cas d'abus spécifiques et pertinents, l'application se vérifie pratiquement d'elle-même à chaque nouvelle itération. Par exemple, des tests supplémentaires peuvent être élaborés pour la suite de tests d'un contrôleur de connexion, en testant le paramètre de nom d'utilisateur pour les noms d'utilisateur communs par défaut, l'énumération des comptes, l'énumération exhaustive (brute-force), l'injection LDAP et SQL, et XSS. De plus, un test sur le paramètre du mot de passe devrait inclure les mots de passe communs, la longueur du mot de passe, l'injection d'octets nuls, la suppression du paramètre, XSS, et plus encore.

Pour la formation au développement sécurisé

L'ASVS peut également être utilisé pour définir les caractéristiques des logiciels sécurisés. De nombreux cours de "codage sécurisé" sont simplement des cours de piratage éthique avec une légère trace de conseils de codage. Cela n'aide pas nécessairement les développeurs à écrire un code plus sûr. Les cours de développement sécurisé peuvent plutôt utiliser l'ASVS en mettant l'accent sur les contrôles proactifs qu'il contient, plutôt que sur les dix principales erreurs à ne pas faire.

En tant que pilote pour la sécurité des applications agiles

L'ASVS peut être utilisé dans un processus de développement agile comme cadre pour définir les tâches spécifiques qui doivent être mises en œuvre par l'équipe pour avoir un produit sécurisé. Une approche pourrait être : En commençant par le niveau 1, vérifier l'application ou le système spécifique selon les exigences de l'ASVS pour le niveau spécifié, trouver les contrôles manquants et soulever des tickets/tâches spécifiques dans le backlog. Cela permet de hiérarchiser les tâches spécifiques (ou grooming) et de rendre la sécurité visible dans le processus agile. Cela peut également être utilisé pour hiérarchiser les tâches d'audit et de révision dans l'organisation, où une exigence ASVS spécifique peut être un moteur de révision, de remaniement ou d'audit pour un membre spécifique de l'équipe et être visible en tant que "dette" dans le backlog qui doit finalement être fait.

Comme cadre pour guider l'acquisition de logiciels sécurisés

L'ASVS est un cadre idéal pour aider à sécuriser l'acquisition de logiciels ou de services de développement personnalisés. L'acheteur peut simplement exiger que le logiciel qu'il souhaite acquérir soit développé au niveau X de l'ASVS, et demander au vendeur de prouver que le logiciel satisfait au niveau X de l'ASVS. Cette procédure fonctionne bien lorsqu'elle est combinée avec l'annexe du contrat de logiciel sécurisé de l'OWASP.

V1 : Architecture, conception et exigences en matière de modélisation des menaces

Objectif de contrôle

L'architecture de sécurité est presque devenue un art perdu dans de nombreuses organisations. Les jours de l'architecte d'entreprise sont passés à l'ère des DevSecOps. Le domaine de la sécurité des applications doit rattraper son retard et adopter des principes de sécurité agiles tout en réintroduisant les principes de l'architecture de sécurité de pointe auprès des praticiens du logiciel. L'architecture n'est pas une implémentation, mais une façon de penser à un problème qui peut avoir plusieurs réponses différentes, et pas une seule réponse "correcte". Trop souvent, la sécurité est considérée comme rigide et exige que les développeurs corrigent le code d'une manière particulière, alors qu'ils connaissent peut-être un bien meilleur moyen de résoudre le problème. Il n'existe pas de solution unique et simple pour l'architecture, et prétendre le contraire est un mauvais service rendu au domaine du génie logiciel.

Une implémentation spécifique d'une application web est susceptible d'être révisée continuellement tout au long de sa vie, mais l'architecture globale ne changera probablement que rarement, mais évoluera lentement. L'architecture de sécurité est identique - nous avons besoin d'une authentification aujourd'hui, nous en aurons besoin demain, et nous en aurons besoin dans cinq ans. Si nous prenons des décisions judicieuses aujourd'hui, nous pouvons économiser beaucoup d'efforts, de temps et d'argent si nous sélectionnons et réutilisons des solutions conformes à l'architecture. Par exemple, il y a dix ans, l'authentification multifactorielle était rarement mise en œuvre.

Si les développeurs avaient investi dans un modèle de fournisseur d'identité unique et sécurisé, tel que l'identité fédérée SAML, le fournisseur d'identité pourrait être mis à jour pour intégrer de nouvelles exigences telles que la conformité NIST 800-63, sans pour autant modifier les interfaces de l'application d'origine. Si de nombreuses applications partageaient la même architecture de sécurité et donc le même composant, elles bénéficient toutes de cette mise à jour en même temps. Toutefois, SAML ne restera pas toujours la meilleure ou la plus appropriée des solutions d'authentification - il pourrait être nécessaire de le remplacer par d'autres solutions au fur et à mesure que les exigences changent. De telles modifications sont soit compliquées, si coûteuses qu'elles nécessitent une réécriture complète, soit carrément impossibles sans architecture de sécurité.

Dans ce chapitre, l'ASVS couvre les principaux aspects de toute architecture de sécurité solide : disponibilité, confidentialité, intégrité du traitement, non-répudiation et respect de la vie privée. Chacun de ces principes de sécurité doit être intégré et inné à toutes les applications. Il est essentiel de "tourner à gauche", en commençant par l'habilitation des développeurs avec des listes de contrôle de codage sécurisé, le mentorat et la formation, le codage et les tests, la construction, le déploiement, la configuration et les opérations, et en terminant par des tests indépendants de suivi pour s'assurer que tous les contrôles de sécurité sont présents et fonctionnels. La dernière étape était autrefois tout ce que nous faisions en tant qu'industrie, mais cela ne suffit plus lorsque les développeurs mettent le code en production des dizaines ou des centaines de fois par jour. Les professionnels de la sécurité des applications doivent se tenir au courant des techniques agiles, ce qui signifie adopter les outils des développeurs, apprendre à coder et travailler avec eux plutôt que de critiquer le projet des mois après que tout le monde est passé à autre chose.

V1.1 Exigences relatives au cycle de vie du développement de logiciels sécurisés

#	Description	L1	L2	L3	CWE
1.1.1	Vérifier l'utilisation d'un cycle de développement de logiciel sécurisé qui prend en compte la sécurité à tous les stades du développement. (C1)		✓	✓	
1.1.2	Vérifier l'utilisation de la modélisation des menaces pour chaque modification de conception ou planification de sprint afin d'identifier les menaces, de planifier les contre-mesures, de faciliter les réponses appropriées aux risques et d'orienter les tests de sécurité.		✓	✓	1053

#	Description	L1	L2	L3	CWE
1.1.3	Vérifiez que toutes les récits utilisateurs et les fonctionnalités contiennent des contraintes de sécurité fonctionnelles, telles que "En tant qu'utilisateur, je devrais pouvoir consulter et modifier mon profil. Je ne devrais pas pouvoir voir ou modifier le profil de quelqu'un d'autre"		✓	✓	1110
1.1.4	Vérifier la documentation et la justification de toutes les frontières de confiance de la demande, de ses composantes et des flux de données importants.		✓	✓	1059
1.1.5	Vérifier la définition et l'analyse de sécurité de l'architecture de haut niveau de l'application et de tous les services à distance connectés. (C1)		✓	✓	1059
1.1.6	Vérifier la mise en œuvre de contrôles de sécurité centralisés, simples (économie de conception), vérifiés, sécurisés et réutilisables pour éviter les contrôles en double, manquants, inefficaces ou peu sûrs. (C10)		✓	✓	637
1.1.7	Vérifier que tous les développeurs et testeurs disposent d'une liste de contrôle de codage sécurisé, d'exigences de sécurité, de lignes directrices ou de politiques.		✓	✓	637
1.1.8	Vérifier la disponibilité d'un fichier security.txt accessible au public à la racine ou dans le répertoire .known de l'application, qui définit clairement un lien ou une adresse électronique permettant aux personnes de contacter les propriétaires pour des questions de sécurité.		✓	✓	1059

V1.2 Exigences architecturales d'authentification

Lors de la conception de l'authentification, il importe peu que vous disposiez d'un matériel solide permettant une authentification multifactorielle si un attaquant peut réinitialiser un compte en appelant un centre d'appel et en répondant à des questions courantes. Lors de la vérification de l'identité, toutes les méthodes d'authentification doivent avoir la même force.

#	Description	L1	L2	L3	CWE
1.2.1	Vérifier que les communications entre les composants de l'application, y compris les API, les intergiciels et les couches de données, sont authentifiées et utilisent des comptes utilisateurs individuels. (C3)		✓	✓	306
1.2.2	Vérifiez que l'application utilise un mécanisme d'authentification unique et contrôlé qui est connu pour être sûr, qui peut être étendu pour inclure une authentification forte et qui dispose d'une journalisation et d'une surveillance suffisantes pour détecter les abus ou les violations de compte.		✓	✓	306
1.2.3	Vérifier que toutes les méthodes d'authentification et les API de gestion de l'identité mettent en œuvre un contrôle de sécurité de l'authentification cohérent, de sorte qu'il n'y ait pas d'alternatives plus faibles par rapport au risque de l'application.		✓	✓	306

V1.3 Exigences architecturales pour la gestion des sessions

Il s'agit d'un point de repère pour les futures exigences architecturales.

V1.4 Exigences architecturales en matière de contrôle d'accès

#	Description	L1	L2	L3	CWE
1.4.1	Vérifiez que des points d'application de confiance tels que les passerelles de contrôle d'accès, les serveurs et les fonctions sans serveur font respecter les contrôles d'accès. N'imposez jamais de contrôles d'accès au client.		✓	✓	602
1.4.2	Vérifiez que la solution de contrôle d'accès choisie est suffisamment souple pour répondre aux besoins de l'application.		✓	✓	284
1.4.3	Vérifier l'application du principe du moindre privilège dans les fonctions, fichiers de données, URL, contrôleurs, services et autres ressources. Cela implique une protection contre l'usurpation et l'élévation des privilèges.		✓	✓	272
1.4.4	Vérifier que les communications entre les composants de l'application, y compris les API, les intergiciels et les couches de données, sont effectuées avec le moins de privilèges possibles. (C3)		✓	✓	272
1.4.5	Vérifier que l'application utilise un mécanisme de contrôle d'accès unique et bien étudié pour accéder aux données et ressources protégées. Toutes les demandes doivent passer par ce mécanisme unique pour éviter le copier-coller ou les chemins alternatifs non sécurisés. (C7)		✓	✓	284
1.4.6	Vérifiez que le contrôle d'accès basé sur les attributs ou les caractéristiques est utilisé, c'est-à-dire que le code vérifie l'autorisation de l'utilisateur pour une caractéristique ou une donnée plutôt que son seul rôle. Les autorisations doivent tout de même être attribuées à l'aide de rôles. (C7)		✓	✓	275

V1.5 Exigences architecturales d'entrée et de sortie

Dans la version 4.0, nous avons abandonné le terme "côté serveur" ayant la connotation de limite de confiance. La limite de confiance est toujours un enjeu - il est possible de contourner les décisions prises sur des navigateurs ou des appareils clients non fiables. Cependant, dans les déploiements architecturaux courants d'aujourd'hui, le point d'application de la confiance a considérablement changé. Par conséquent, lorsque le terme "couche de service de confiance" est utilisé dans l'ASVS, nous entendons par là tout point d'application de confiance, quel que soit son emplacement, tel qu'un microservice, un API sans serveur, côté serveur, un API de confiance sur un périphérique client qui a un démarrage sécurisé, des API partenaires ou externes, etc.

#	Description	L1	L2	L3	CWE
1.5.1	Vérifier que les exigences en matière d'entrée et de sortie définissent clairement la manière de traiter et d'exploiter les données en fonction du type, du contenu et de la conformité aux lois, règlements et autres politiques applicables.		✓	✓	1029
1.5.2	Vérifiez que la sérialisation n'est pas utilisée lorsque vous communiquez avec des clients non fiables. Si cela n'est pas possible, assurez-vous que des contrôles d'intégrité adéquats (et éventuellement un cryptage si des données sensibles sont envoyées) sont appliqués pour empêcher les attaques de désérialisation, y compris l'injection d'objets.		✓	✓	502
1.5.3	Vérifiez que la validation des entrées est appliquée sur une couche de service de confiance. (C5)		✓	✓	602
1.5.4	Vérifiez que l'encodage de sortie se fait à proximité ou par l'interprète auquel il est destiné. (C4)		✓	✓	116

V1.6 Exigences en matière d'architecture cryptographique

Les applications doivent être conçues avec une architecture cryptographique solide pour protéger les données selon leur classification. Tout chiffrer est un gaspillage, ne rien chiffrer est une négligence légale. Un équilibre doit être trouvé, généralement lors de la conception architecturale ou de haut niveau, des sprints de conception ou des pics architecturaux. Concevoir la cryptographie au fur et à mesure ou l'adapter coûtera inévitablement beaucoup plus cher à mettre en œuvre de manière sécurisée que de l'intégrer dès le départ.

Les exigences architecturales sont intrinsèques au code, et donc difficiles à unifier ou à intégrer dans les tests. Les exigences architecturales doivent être prises en compte dans les normes de codage, tout au long de la phase de codage, et doivent être examinées au cours de l'architecture de sécurité, des revues du code par les pairs, ou des rétrospectives.

#	Description	L1	L2	L3	CWE
1.6.1	Vérifier qu'il existe une politique explicite de gestion des clés cryptographiques et que le cycle de vie d'une clé cryptographique suit une norme de gestion des clés telle que NIST SP 800-57.		✓	✓	320
1.6.2	Vérifier que les consommateurs de services cryptographiques protègent les clés et autres secrets en utilisant des coffres-forts de clés ou des alternatives basées sur l'API.		✓	✓	320
1.6.3	Vérifiez que toutes les clés et tous les mots de passe sont remplaçables et font partie d'un processus bien défini de recryptage des données sensibles.		✓	✓	320
1.6.4	Vérifier que les clés symétriques, les mots de passe ou les secrets d'API générés par les clients ou partagés avec eux ne sont utilisés que pour protéger des secrets à faible risque, comme le chiffrement du stockage local, ou des utilisations éphémères temporaires comme l'obscurcissement des paramètres. Le partage de secrets avec les clients est équivalent à du texte en clair et, d'un point de vue architectural, il doit être traité comme tel.		✓	✓	320

V1.7 Erreurs, enregistrement et vérification des exigences architecturales

#	Description	L1	L2	L3	CWE
1.7.1	Vérifier qu'un format de journalisation communs soit utilisés dans le système. (C9)		✓	✓	1009
1.7.2	Vérifiez que les journaux sont transmis de manière sécurisée à un système de préférence distant pour analyse, détection, alerte et escalade. (C9)		✓	✓	

V1.8 Exigences architecturales en matière de protection des données et de la vie privée

#	Description	L1	L2	L3	CWE
1.8.1	Vérifier que toutes les données sensibles sont identifiées et classées en niveaux de protection.		✓	✓	
1.8.2	Vérifier que tous les niveaux de protection sont associés à un ensemble d'exigences de protection, telles que des exigences de cryptage, d'intégrité, de conservation, de respect de la vie privée et d'autres exigences de confidentialité, et que celles-ci sont appliquées dans l'architecture.		✓	✓	

V1.9 Exigences en matière d'architecture des communications

#	Description	L1	L2	L3	CWE
1.9.1	Vérifier que l'application chiffre les communications entre les composants, en particulier lorsque ces composants se trouvent dans des conteneurs, systèmes, sites ou fournisseurs de cloud différents. (C3)		✓	✓	319
1.9.2	Vérifier que les composants de l'application vérifient l'authenticité de chaque partie d'un lien de communication afin de prévenir les attaques de type "man-in-the-middle". Par exemple, les composants d'application doivent valider les certificats et les chaînes TLS.		✓	✓	295

V1.10 Exigences en matière d'architecture des logiciels malveillants

#	Description	L1	L2	L3	CWE
1.10.1	Vérifier qu'un système de contrôle du code source est utilisé, avec des procédures pour s'assurer que les enregistrements sont accompagnés de tickets d'émission ou de modification. Le système de contrôle du code source doit disposer d'un contrôle d'accès et d'utilisateurs identifiables pour permettre la traçabilité de toute modification.		✓	✓	284

V1.11 Exigences en matière d'architecture de la logique d'entreprise

#	Description	L1	L2	L3	CWE
1.11.1	Vérifier la définition et la documentation de tous les composants de l'application en ce qui concerne la logique métier ou de sécurité qu'ils fournissent.		✓	✓	1059
1.11.2	Vérifiez que tous les flux de logique métier de grande valeur, y compris l'authentification, la gestion de session et le contrôle d'accès, ne partagent pas un état non synchronisé.		✓	✓	362
1.11.3	Vérifier que tous les flux de logique métier de grande valeur, y compris l'authentification, la gestion de session et le contrôle d'accès, sont sécurisés et résistants aux conditions de concurrence ("race condition") au temps de contrôle et au temps d'utilisation.			✓	367

V1.12 Téléchargement de fichiers sécurisés Exigences architecturales

#	Description	L1	L2	L3	CWE
1.12.1	Vérifiez que les fichiers envoyés par l'utilisateur sont stockés en dehors de la racine du site web.		✓	✓	552
1.12.2	Vérifiez que les fichiers envoyés par l'utilisateur - s'ils doivent être affichés ou téléchargés à partir de l'application - sont servis par des téléchargements en flux d'octets, ou à partir d'un domaine sans rapport, comme un compartiment de stockage de fichiers en nuage. Mettre en œuvre une politique de sécurité du contenu (CSP) appropriée pour réduire le risque de vecteurs XSS ou d'autres attaques provenant du fichier téléchargé.		✓	✓	646

V1.13 Exigences architecturales de l'API

Il s'agit d'un point de repère pour les futures exigences architecturales.

V1.14 Configuration des exigences architecturales

#	Description	L1	L2	L3	CWE
1.14.1	Vérifier l'utilisation d'un compte à faible privilège pour tous les composants d'application, services et serveurs. (C3)		✓	✓	250
1.14.2	Vérifiez la séparation des composants de différents niveaux de confiance via des contrôles de sécurité bien définis, des règles de pare-feu, des passerelles API, des proxys inverses, des groupes de sécurité basés sur le cloud ou des mécanismes similaires.		✓	✓	923
1.14.3	Vérifier que les signatures binaires, les connexions de confiance et les nœuds vérifiés sont utilisés pour déployer des binaires sur des dispositifs distants.		✓	✓	494
1.14.4	Vérifier que le pipeline de construction signale les composants obsolètes ou peu sûrs et prend les mesures appropriées.		✓	✓	1104
1.14.5	Vérifier que le pipeline de construction contient une étape de compilation pour créer automatiquement et vérifier le déploiement sécurisé de l'application, en particulier si l'infrastructure de l'application est définie par un logiciel, comme les scripts de construction d'un environnement en nuage.		✓	✓	
1.14.6	Vérifier que les déploiements d'applications sont correctement sandboxés, conteneurisés et/ou isolés au niveau du réseau pour retarder et dissuader les attaquants d'attaquer d'autres applications, en particulier lorsqu'ils effectuent des actions sensibles ou dangereuses telles que la désérialisation. (C5)		✓	✓	265
1.14.7	Vérifiez que l'application n'utilise pas de technologies côté client non prises en charge, peu sûres ou dépréciées telles que les plugins NSAPI, Flash, Shockwave, ActiveX, Silverlight, NACL ou les applets Java côté client.		✓	✓	477

Références

Pour plus d'informations, voir aussi :

- OWASP Threat Modeling Cheat Sheet
- OWASP Attack Surface Analysis Cheat Sheet
- OWASP Threat modeling
- OWASP Software Assurance Maturity Model Project
- Microsoft SDL
- NIST SP 800-57

V2 : Exigences de vérification de l'authentification

Objectif de contrôle

L'authentification est l'acte qui consiste à établir ou à confirmer qu'une personne (ou quelque chose) est authentique et que les affirmations faites par une personne ou au sujet d'un dispositif sont correctes, résistantes à l'usurpation d'identité et empêchent la récupération ou l'interception des mots de passe.

Lorsque l'ASVS a été publiée pour la première fois, le nom d'utilisateur + mot de passe était la forme d'authentification la plus courante en dehors des systèmes de haute sécurité. L'authentification à facteurs multiples (MFA) était communément acceptée dans les cercles de sécurité mais rarement requise ailleurs. Avec l'augmentation du nombre de brèches de mots de passe, l'idée que les noms d'utilisateur sont en quelque sorte confidentiels et les mots de passe inconnus, a rendu de nombreux contrôles de sécurité inefficaces. Par exemple, le NIST 800-63 considère les noms d'utilisateur et l'authentification basée sur la connaissance (KBA) comme des informations publiques, les notifications par SMS et par courrier électronique comme des types d'authentificateurs "restreints", et les mots de passe comme des préviolations. Cette réalité rend inutiles les authentificateurs basés sur la connaissance, la récupération de SMS et de courriels, l'historique des mots de passe, la complexité et les contrôles de rotation. Ces contrôles ont toujours été peu utiles, obligeant souvent les utilisateurs à trouver des mots de passe faibles tous les quelques mois, mais avec la publication de plus de 5 milliards de violations de noms d'utilisateur et de mots de passe, il est temps de passer à autre chose.

De toutes les sections de l'ASVS, ce sont les chapitres sur l'authentification et la gestion des sessions qui ont le plus changé. L'adoption de pratiques de pointe efficaces et fondées sur des preuves sera un défi pour beaucoup, et c'est tout à fait normal. Nous devons commencer dès maintenant la transition vers un avenir post-mot de passe.

NIST 800-63 - Norme d'authentification moderne et fondée sur des preuves

NIST 800-63b est une norme moderne, basée sur des preuves, et représente le meilleur conseil disponible, indépendamment de son applicabilité. La norme est utile à toutes les organisations du monde entier, mais elle est particulièrement pertinente pour les agences américaines et celles qui traitent avec des agences américaines.

La terminologie de la norme NIST 800-63 peut être un peu déroutante au début, surtout si vous n'êtes habitué qu'à l'authentification par nom d'utilisateur + mot de passe. Des progrès en matière d'authentification moderne sont nécessaires, nous devons donc introduire une terminologie qui deviendra courante à l'avenir, mais nous comprenons la difficulté de compréhension jusqu'à ce que l'industrie s'habitue à ces nouveaux termes. Nous avons fourni un glossaire à la fin de ce chapitre pour vous aider. Nous avons reformulé de nombreuses exigences afin de satisfaire l'intention, plutôt que sa forme. Par exemple, l'ASVS utilise le terme "mot de passe" alors que le NIST utilise "secret mémorisé" tout au long de cette norme.

L'authentification ASVS V2, la gestion de session V3 et, dans une moindre mesure, les contrôles d'accès V4 ont été adaptés pour constituer un sous-ensemble conforme de certains contrôles NIST 800-63b, axés sur les menaces communes et les faiblesses d'authentification couramment exploitées. Lorsque la conformité totale à la norme NIST 800-63 est requise, veuillez consulter la norme NIST 800-63.

Sélection d'un niveau NIST AAL approprié

La norme de vérification de la sécurité des applications a tenté de faire correspondre les exigences ASVS L1 à celles du NIST AAL1, L2 à AAL2, et L3 à AAL3. Cependant, l'approche du niveau 1 ASVS en tant que contrôles "essentiels" n'est pas nécessairement le niveau AAL correct pour vérifier une application ou une API. Par exemple, si l'application est une application de niveau 3 ou a des exigences réglementaires pour être AAL3, le niveau 3 doit être choisi dans les sections V2 et V3 Gestion des sessions. Le choix du niveau d'affirmation d'authentification (AAL) conforme aux normes NIST doit être effectué conformément aux directives NIST 800-63b, comme indiqué dans la section *Selecting AAL* du NIST 800-63b Section 6.2.

Légende

Les applications peuvent toujours dépasser les exigences du niveau actuel, surtout si l'authentification moderne figure sur la feuille de route d'une application. Auparavant, l'ASVS exigeait une AMF obligatoire. Le NIST n'exige pas d'AMF obligatoire. C'est pourquoi nous avons utilisé une désignation optionnelle dans ce chapitre pour indiquer les cas où l'ASVS encourage mais n'exige pas de contrôle. Les clés suivantes sont utilisées tout au long de cette norme :

Marque	Description
	Non requis
o	Recommandé, mais pas obligatoire
✓	Obligatoire

V2.1 Exigences en matière de sécurité des mots de passe

Les mots de passe, appelés "Memorized Secrets" par NIST 800-63, comprennent les mots de passe, les codes PIN, les motifs de déverrouillage, le choix du chaton correct ou d'un autre élément d'image, et les phrases de passe. Ils sont généralement considérés comme "quelque chose que vous savez" et sont souvent utilisés comme des authentificateurs à facteur unique. L'utilisation continue de l'authentification à un facteur unique pose des problèmes importants, notamment les milliards de noms d'utilisateur et de mots de passe valides divulgués sur l'internet, les mots de passe par défaut ou faibles, les tables arc-en-ciel et les dictionnaires ordonnés des mots de passe les plus courants.

Les applications devraient fortement encourager les utilisateurs à s'inscrire à l'authentification multi-facteurs, et devraient permettre aux utilisateurs de réutiliser les jetons qu'ils possèdent déjà, tels que les jetons FIDO ou U2F, ou de se relier à un fournisseur de services d'accréditation qui fournit une authentification multi-facteurs.

Les fournisseurs de services d'accréditation (CSP) fournissent une identité fédérée aux utilisateurs. Les utilisateurs auront souvent plus d'une identité avec plusieurs CSP, comme une identité d'entreprise utilisant Azure AD, Okta, Ping Identity ou Google, ou une identité de consommateur utilisant Facebook, Twitter, Google ou WeChat, pour ne citer que quelques alternatives courantes. Cette liste n'est pas une approbation de ces entreprises ou services, mais simplement un encouragement pour les développeurs à prendre en compte la réalité que de nombreux utilisateurs ont plusieurs identités établies. Les organisations devraient envisager de s'intégrer aux identités des utilisateurs existants, conformément au profil de risque de la force du CSP en matière de vérification d'identité. Par exemple, il est peu probable qu'une organisation gouvernementale accepte une identité de média social comme identifiant pour des systèmes sensibles, car il est facile de créer de fausses identités ou de jeter des identités, alors qu'une société de jeux pour mobiles pourrait bien avoir besoin de s'intégrer aux principales plateformes de médias sociaux pour accroître sa base de joueurs actifs.

#	Description	L1	L2	L3	CWE	NIST
2.1.1	Vérifiez que les mots de passe définis par l'utilisateur comportent au moins 12 caractères. (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Vérifiez que les mots de passe de 64 caractères ou plus sont autorisés. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Vérifiez que le mot de passe n'est pas tronqué. Toutefois, des espaces multiples consécutifs peuvent être remplacés par un seul espace. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Vérifiez que tout caractère Unicode imprimable, y compris les caractères neutres comme les espaces et les Emojis, sont autorisés dans les mots de passe.	✓	✓	✓	521	5.1.1.2
2.1.5	Vérifier que les utilisateurs peuvent changer leur mot de passe.	✓	✓	✓	620	5.1.1.2

#	Description	L1	L2	L3	CWE	NIST
2.1.6	Vérifiez que la fonctionnalité de changement de mot de passe nécessite le mot de passe actuel et le nouveau mot de passe de l'utilisateur.	✓	✓	✓	620	5.1.1.2
2.1.7	Vérifiez que les mots de passe soumis lors de l'enregistrement du compte, de la connexion et de la modification du mot de passe sont vérifiés par rapport à un ensemble de mots de passe non respectés, soit localement (comme les 1 000 ou 10 000 mots de passe les plus courants qui correspondent à la politique du système en matière de mots de passe), soit en utilisant une API externe. En cas d'utilisation d'une API, il convient d'utiliser un mécanisme de vérification de l'absence de connaissance ou un autre mécanisme pour s'assurer que le mot de passe en texte clair n'est pas envoyé ou utilisé pour vérifier l'état de violation du mot de passe. En cas de violation du mot de passe, l'application doit demander à l'utilisateur de définir un nouveau mot de passe non violé. (C6)	✓	✓	✓	521	5.1.1.2
2.1.8	Vérifiez qu'un compteur de force de mot de passe est fourni pour aider les utilisateurs à définir un mot de passe plus fort.	✓	✓	✓	521	5.1.1.2
2.1.9	Vérifiez qu'il n'existe pas de règles de composition des mots de passe limitant le type de caractères autorisés. Il ne doit pas y avoir restrictions sur l'utilisation de majuscules ou de minuscules, de chiffres ou de caractères spéciaux. (C6)	✓	✓	✓	521	5.1.1.2
2.1.10	Vérifiez qu'il n'y a pas d'exigences en matière de rotation périodique du mots de passe ou d'historique des mots de passe.	✓	✓	✓	263	5.1.1.2
2.1.11	Vérifiez que la fonction "coller", les aides de mot de passe du navigateur et les gestionnaires de mots de passe externes sont autorisés.	✓	✓	✓	521	5.1.1.2
2.1.12	Vérifiez que l'utilisateur peut choisir soit de visualiser temporairement la totalité du mot de passe masqué, soit de visualiser temporairement le dernier caractère tapé du mot de passe sur les plateformes qui n'ont pas cette fonctionnalité en natif.	✓	✓	✓	521	5.1.1.2

Note : L'objectif de permettre à l'utilisateur de visualiser son mot de passe ou de voir temporairement le dernier caractère est d'améliorer la facilité d'utilisation de la saisie des données d'identification, notamment en ce qui concerne l'utilisation de mots de passe plus longs, de phrases de passe et de gestionnaires de mots de passe. Une autre raison d'inclure cette exigence est de dissuader ou d'empêcher les rapports de test exigeant inutilement des organisations de passer outre le comportement du champ de mot de passe de la plate-forme native pour supprimer cette expérience de sécurité moderne et conviviale.

V2.2 Exigences générales relatives aux authentificateurs

L'agilité des authentificateurs est essentielle pour que les applications soient à l'épreuve du temps. Les vérificateurs d'applications Refactor permettent d'ajouter des authentificateurs supplémentaires en fonction des préférences de l'utilisateur, et de mettre à la retraite de manière ordonnée les authentificateurs obsolètes ou dangereux.

Le NIST considère le courrier électronique et les SMS comme des types d'authentificateurs "restreints", et il est probable qu'ils soient retirés du NIST 800-63 et donc de l'ASVS à un moment donné dans le futur. Les applications devraient prévoir une feuille de route qui ne nécessite pas l'utilisation du courrier électronique ou des SMS.

#	Description	L1	L2	L3	CWE	NIST
2.2.1	Vérifiez que les contrôles anti-automatisation sont efficaces pour atténuer les attaques par violation des tests d'accréditation, par énumération exhaustive (brute-force) et par verrouillage de compte. Ces contrôles comprennent le blocage des mots de passe les plus courants, les verrouillages progressifs, la limitation de débit, les CAPTCHA, les délais toujours plus longs entre les tentatives, les restrictions d'adresse IP ou les restrictions basées sur le risque telles que l'emplacement, la première connexion sur un appareil, les tentatives récentes de déverrouillage du compte, ou autres. Vérifiez qu'il n'y a pas plus de 100 tentatives infructueuses par heure sur un seul compte.	✓	✓	✓	307	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2
2.2.2	Vérifier que l'utilisation d'authentificateurs faibles (tels que les SMS et les e-mails) se limite à la vérification secondaire et à l'approbation des transactions et ne remplace pas les méthodes d'authentification plus sûres. Vérifiez que les méthodes plus fortes sont proposées avant les méthodes faibles, que les utilisateurs sont conscients des risques, ou que des mesures appropriées sont en place pour limiter les risques de compromission du compte.	✓	✓	✓	304	5.2.10
2.2.3	Vérifier que des notifications sécurisées sont envoyées aux utilisateurs après la mise à jour des détails d'authentification, tels que la réinitialisation de l'identifiant, le changement d'adresse électronique ou d'adresse, la connexion à partir d'un lieu inconnu ou risqué. L'utilisation de notifications "push" - plutôt que de SMS ou d'e-mail - est préférable, mais en l'absence de notifications "push", les SMS ou les e-mails sont acceptables tant qu'aucune information sensible n'est divulguée dans la notification.	✓	✓	✓	620	
2.2.4	Vérifier la résistance à l'usurpation d'identité contre le phishing, comme l'utilisation de l'authentification multi-facteurs, les dispositifs cryptographiques avec intention (comme les clés connectées avec un "push to authenticate"), ou à des niveaux AAL supérieurs, les certificats côté client.			✓	308	5.2.5
2.2.5	Vérifier que lorsqu'un fournisseur de services d'accréditation (CSP) et l'application vérifiant l'authentification sont séparés, un TLS mutuellement authentifié est en place entre les deux points terminaux.			✓	319	5.2.6

#	Description	L1	L2	L3	CWE	NIST
2.2.6	Vérifier la résistance au attaque de rejeu par l'utilisation obligatoire de dispositifs OTP, d'authentificateurs cryptographiques ou de codes de consultation.			✓	308	5.2.8
2.2.7	Vérifier l'intention d'authentification en exigeant l'entrée d'un jeton OTP ou une action initiée par l'utilisateur telle qu'une pression sur un bouton d'une clé matérielle FIDO.			✓	308	5.2.9

V2.3 Exigences relatives au cycle de vie des authentificateurs

Les authentificateurs sont les mots de passe, les jetons logiciels, les jetons matériels et les dispositifs biométriques. Le cycle de vie des authentificateurs est essentiel pour la sécurité d'une application - si quelqu'un peut s'enregistrer lui-même sur un compte sans preuve d'identité, il ne peut guère faire confiance à l'affirmation de son identité. Pour les sites de médias sociaux comme Reddit, c'est tout à fait normal. Pour les systèmes bancaires, il est essentiel de mettre davantage l'accent sur l'enregistrement et la délivrance de justificatifs d'identité et de dispositifs pour assurer la sécurité de l'application.

Remarque : les mots de passe ne doivent pas avoir une durée de vie maximale ni être soumis à une rotation. Les mots de passe doivent être vérifiés et non remplacés régulièrement.

#	Description	L1	L2	L3	CWE	NIST
2.3.1	Vérifier que les mots de passe ou codes d'activation initiaux générés par le système DOIVENT être générés de manière aléatoire et sécurisée, DOIVENT comporter au moins 6 caractères, PEUVENT contenir des lettres et des chiffres, et expirent après une courte période de temps. Ces secrets initiaux ne doivent pas être autorisés à devenir le mot de passe à long terme.	✓	✓	✓	330	5.1.1.2 / A.3
2.3.2	Vérifiez que l'inscription et l'utilisation de dispositifs d'authentification fournis par l'abonné sont prises en charge, comme les jetons U2F ou FIDO.		✓	✓	308	6.1.3
2.3.3	Vérifier que les instructions de renouvellement sont envoyées suffisamment tôt pour renouveler les authentificateurs à durée déterminée.		✓	✓	287	6.1.4

V2.4 Exigences en matière de stockage des identifiants

Les architectes et les développeurs doivent se conformer à cette section lorsqu'ils construisent ou remanient du code. Cette section ne peut être entièrement vérifiée qu'en utilisant la révision du code source ou par des tests unitaires ou d'intégration sécurisés. Les tests d'intrusions ne peuvent pas identifier l'un de ces problèmes.

La liste des fonctions de dérivation de clés à sens unique approuvées est détaillée dans la section 5.1.1.2 de la norme NIST 800-63 B, et dans BSI Kryptographische Verfahren : Empfehlungen und Schlussellängen (2018). Ces choix peuvent être remplacés par les derniers algorithmes nationaux ou régionaux et les dernières normes de longueur de clé.

Cette section ne peut pas être soumise à un test de pénétration, les contrôles ne sont donc pas marqués comme L1. Cependant, cette section est d'une importance vitale pour la sécurité des données d'identifications en cas de vol. Si vous bifurquez l'ASVS pour une directive sur l'architecture ou le codage ou une liste de contrôle pour l'examen du code source, veuillez replacer ces contrôles en L1 dans votre version privée.

#	Description	L1	L2	L3	CWE	NIST
2.4.1	Vérifiez que les mots de passe sont stockés sous une forme qui résiste aux attaques hors ligne. Les mots de passe DOIVENT être salés et hachés en utilisant une fonction approuvée à sens unique ou de hachage de mot de passe. Les fonctions de dérivation de clé et de hachage de mot de passe prennent un mot de passe, un sel et un facteur de coût (ex. : nombre d'itération algorithmique) comme intrants lors de la génération d'un hachage de mot de passe. (C6)		✓	✓	916	5.1.1.2
2.4.2	Vérifiez que le sel a une longueur d'au moins 32 bits et qu'il est choisi arbitrairement pour minimiser les collisions de la valeur du sel parmi les hashs stockés. Pour chaque authentifiant, une valeur de sel unique et le hachage qui en résulte DOIVENT être stockés. (C6)		✓	✓	916	5.1.1.2
2.4.3	Vérifiez que si le PBKDF2 est utilisé, le nombre d'itérations DOIT être aussi important que les performances du serveur de vérification le permettent, généralement au moins 100 000 itérations. (C6)		✓	✓	916	5.1.1.2
2.4.4	Vérifiez que si bcrypt est utilisé, le facteur de travail DOIT être aussi important que les performances du serveur de vérification le permettent, généralement au moins 13. (C6)		✓	✓	916	5.1.1.2
2.4.5	Vérifier qu'une itération supplémentaire d'une fonction de dérivation clé est effectuée, en utilisant une valeur de sel qui est secrète et connue uniquement du vérificateur. Générer la valeur de sel en utilisant un générateur de bits aléatoires approuvé [SP 800-90Ar1] et fournir au moins la sécurité minimale spécifiée dans la dernière révision de la norme SP 800-131A. La valeur de sel secrète DOIT être stockée séparément des mots de passe hachés (par exemple, dans un dispositif spécialisé comme un module de sécurité matériel).		✓	✓	916	5.1.1.2

Lorsque des normes américaines sont mentionnées, une norme régionale ou locale peut être utilisée à la place ou en plus de la norme américaine, selon les besoins.

V2.5 Exigences en matière de récupération des identifiants

#	Description	L1	L2	L3	CWE	NIST
2.5.1	Vérifiez que si un secret d'activation initiale ou de récupération du système est envoyé à l'utilisateur, il est à usage unique, limité dans le temps et aléatoire. (C6)	✓	✓	✓	640	5.1.1.2
2.5.2	Vérifier que les indices de mot de passe ou l'authentification basée sur la connaissance (dites "questions secrètes") ne sont pas présents.	✓	✓	✓	640	5.1.1.2
2.5.3	Vérifiez que la récupération du mot de passe ne révèle en aucune façon le mot de passe actuel. (C6)	✓	✓	✓	640	5.1.1.2
2.5.4	Vérifiez que les comptes partagés ou par défaut ne sont pas présents (par exemple "root", "admin" ou "sa").	✓	✓	✓	16	5.1.1.2 / A.3
2.5.5	Vérifier que si un facteur d'authentification est modifié ou remplacé, l'utilisateur est informé de cet événement.	✓	✓	✓	304	6.1.2.3
2.5.6	Vérifier les mots de passe oubliés, et les autres chemins de récupération utilisent un mécanisme de récupération sécurisé, tel que TOTP ou autre soft token, mobile push, ou un autre mécanisme de récupération hors ligne. (C6)	✓	✓	✓	640	5.1.1.2
2.5.7	Vérifier qu'en cas de perte des facteurs d'authentification OTP ou multi-facteurs, la preuve d'identité est effectuée au même niveau que lors de l'inscription.		✓	✓	308	6.1.2.3

V2.6 Exigences relatives aux vérificateurs des secrets

Les tables d'authentifications secrètes sont des listes pré-générées de codes secrets, similaires aux numéros d'autorisation de transaction (TAN), aux codes de récupération des médias sociaux ou à une grille contenant un ensemble de valeurs aléatoires. Ils sont distribués aux utilisateurs en toute sécurité. Ces codes de recherche sont utilisés une fois, et une fois qu'ils sont tous utilisés, la liste secrète de recherche est jetée. Ce type d'authentificateur est considéré comme "quelque chose que vous avez".

#	Description	L1	L2	L3	CWE	NIST
2.6.1	Vérifiez que les secrets de la table d'authentification ne peuvent être utilisés qu'une seule fois.		✓	✓	308	5.1.2.2
2.6.2	Vérifiez que les secrets de la table d'authentification ont un caractère aléatoire suffisant (112 bits d'entropie) ou, s'ils ont moins de 112 bits d'entropie, qu'ils sont salés avec un sel unique et aléatoire de 32 bits et hachés avec un hachage unidirectionnel approuvé.		✓	✓	330	5.1.2.2
2.6.3	Vérifiez que les secrets de la table d'authentification résistent aux attaques hors ligne, comme les valeurs prévisibles.		✓	✓	310	5.1.2.2

V2.7 Exigences relatives aux vérificateurs hors bande

Dans le passé, un vérificateur hors bande courant aurait été un courriel ou un SMS contenant un lien de réinitialisation de mot de passe. Les attaquants utilisent ce faible mécanisme pour réinitialiser des comptes qu'ils ne contrôlent pas encore, par exemple en prenant le compte de courrier électronique d'une personne et en réutilisant tout lien de réinitialisation découvert. Il existe de meilleurs moyens de gérer la vérification hors bande.

Les authentificateurs hors bande sécurisés sont des dispositifs physiques qui peuvent communiquer avec le vérificateur par un canal secondaire sécurisé. Les notifications "push" vers les appareils mobiles en sont des exemples. Ce type d'authentificateur est considéré comme "quelque chose que vous avez". Lorsqu'un utilisateur souhaite s'authentifier, l'application de vérification envoie un message à l'authentificateur hors bande via une connexion à l'authentificateur directement ou indirectement par le biais d'un service tiers. Le message contient un code d'authentification (généralement un nombre aléatoire de six chiffres ou un dialogue d'approbation modale). L'application de vérification attend de recevoir le code d'authentification par le canal principal et compare le hachage de la valeur reçue au hachage du code d'authentification original. En cas de correspondance, le vérificateur hors bande peut supposer que l'utilisateur s'est authentifié.

L'ASVS suppose que seuls quelques développeurs développeront de nouveaux authentificateurs hors bande, tels que les notifications "push", et donc que les contrôles suivants de l'ASVS s'appliquent aux vérificateurs, tels que l'API d'authentification, les applications et les mises en œuvre de l'authentification unique. Si vous développez un nouvel authentificateur hors bande, veuillez vous référer à NIST 800-63B 5.1.3.1.

Les authentificateurs hors bande dangereux tels que le courrier électronique et la voix sur IP ne sont pas autorisés. Les authentifications PSTN et SMS sont actuellement "restreintes" par le NIST et devraient être interdites au profit des notifications "push" ou similaires. Si vous devez utiliser l'authentification hors bande par téléphone ou SMS, veuillez consulter 5.1.3.3.

#	Description	L1	L2	L3	CWE	NIST
2.7.1	Vérifiez que les authentificateurs de texte en clair hors bande (NIST "restreint"), tels que les SMS ou le PSTN, ne sont pas proposés par défaut, et que des alternatives plus fortes, telles que les notifications "push", sont proposées en premier lieu.	✓	✓	✓	287	5.1.3.2
2.7.2	Vérifiez que le vérificateur hors bande expire les demandes d'authentification, les codes ou les jetons hors bande après 10 minutes.	✓	✓	✓	287	5.1.3.2
2.7.3	Vérifiez que les demandes d'authentification, codes ou jetons hors bande du vérificateur ne sont utilisables qu'une seule fois, et uniquement pour la demande d'authentification originale.	✓	✓	✓	287	5.1.3.2
2.7.4	Vérifier que l'authentificateur et le vérificateur hors bande communiquent sur un canal indépendant sécurisé.	✓	✓	✓	523	5.1.3.2
2.7.5	Vérifiez que le vérificateur hors bande ne conserve qu'une version hachée du code d'authentification.		✓	✓	256	5.1.3.2
2.7.6	Vérifiez que le code d'authentification initial est généré par un générateur de nombres aléatoires sécurisé, contenant au moins 20 bits d'entropie (en général, un nombre aléatoire de six chiffres est suffisant).		✓	✓	310	5.1.3.2

V2.8 Exigences relatives aux vérificateurs uniques à facteur unique ou à facteurs multiples

Les mots de passe à usage unique (OTP) sont des jetons physiques ou logiciels qui affichent un défi pseudo-aléatoire à usage unique en constante évolution. Ces dispositifs rendent le phishing (usurpation d'identité) difficile, mais pas impossible. Ce type d'authentifiant est considéré comme "quelque chose que vous avez". Les jetons multi-facteurs sont similaires aux OTP à facteur unique, mais nécessitent un code PIN valide, un déverrouillage biométrique, une insertion USB ou un appariement NFC ou une valeur supplémentaire (comme les calculatrices de signature de transaction) à saisir pour créer l'OTP final.

#	Description	L1	L2	L3	CWE	NIST
2.8.1	Vérifier que les OTP basées sur le temps ont une durée de vie définie avant d'expirer.	✓	✓	✓	613	5.1.4.2 / 5.1.5.2
2.8.2	Vérifier que les clés symétriques utilisées pour vérifier les OTP soumises sont hautement protégées, par exemple en utilisant un module de sécurité matériel ou un stockage de clés basé sur un système d'exploitation sécurisé.		✓	✓	320	5.1.4.2 / 5.1.5.2
2.8.3	Vérifier que des algorithmes cryptographiques approuvés sont utilisés dans la génération, dans la préparation et dans la vérification des OTP.		✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	Vérifiez que l'OTP basé sur le temps ne peut être utilisé qu'une seule fois pendant la période de validité.		✓	✓	287	5.1.4.2 / 5.1.5.2
2.8.5	Vérifier que si un jeton OTP multi-facteur basé sur le temps est réutilisé pendant la période de validité, il est enregistré et rejeté avec des notifications sécurisées envoyées au détenteur du dispositif.		✓	✓	287	5.1.5.2
2.8.6	Vérifier que le générateur OTP physique à facteur unique peut être révoqué en cas de vol ou autre perte. S'assurer que la révocation est immédiatement effective pour toutes les sessions connectées, quel que soit le lieu.		✓	✓	613	5.2.1
2.8.7	Vérifiez que les authentificateurs biométriques sont limités à une utilisation en tant que facteurs secondaires en conjonction avec quelque chose que vous avez et quelque chose que vous connaissez.		o	✓	308	5.2.3

V2.9 Exigences relatives aux vérificateurs de logiciels et d'appareils cryptographiques

Les clés de sécurité cryptographiques sont des cartes à puce ou des clés FIDO, où l'utilisateur doit brancher ou apparier le dispositif cryptographique à l'ordinateur pour compléter l'authentification. Les vérificateurs envoient un défi aux dispositifs ou logiciels cryptographiques, et le dispositif ou le logiciel calcule une réponse basée sur une clé cryptographique stockée en toute sécurité.

Les exigences relatives aux dispositifs et logiciels cryptographiques à facteur unique et aux dispositifs et logiciels cryptographiques à facteurs multiples sont les mêmes, car la vérification de l'authentificateur cryptographique prouve la possession du facteur d'authentification.

#	Description	L1	L2	L3	CWE	NIST
2.9.1	Vérifier que les clés cryptographiques utilisées dans la vérification sont stockées de manière sûre et protégées contre la divulgation, par exemple en utilisant un TPM ou un HSM, ou un service OS qui peut utiliser ce stockage sécurisé.		✓	✓	320	5.1.7.2
2.9.2	Vérifiez que le nonce de défi est d'une longueur d'au moins 64 bits, et statistiquement unique ou non pendant la durée de vie du dispositif cryptographique.		✓	✓	330	5.1.7.2
2.9.3	Vérifier que des algorithmes cryptographiques approuvés sont utilisés dans la génération, la préparation et la vérification.		✓	✓	327	5.1.7.2

V2.10 Exigences d'authentification des services

Cette section n'est pas testable par tests d'intrusions, et n'a donc aucune exigence de L1. Toutefois, si elle est utilisée dans une architecture, un codage ou une révision de code sécurisé, veuillez supposer que le logiciel (tout comme le Java Key Store) est l'exigence minimale de la L1. Le stockage de secrets en texte clair n'est en aucun cas acceptable.

#	Description	L1	L2	L3	CWE	NIST
2.10.1	Vérifiez que les secrets d'intégration ne reposent pas sur des mots de passe immuables, tels que les clés API ou les comptes privilégiés partagés.		OS assisté	HSM	287	5.1.1.1
2.10.2	Vérifiez que si des mots de passe sont requis, l'authentification ne soit pas avec un compte par défaut.		OS assisté	HSM	255	5.1.1.1
2.10.3	Vérifiez que les mots de passe sont stockés avec une protection suffisante pour empêcher les attaques de récupération hors ligne, y compris l'accès au système local.		Assistance du système d'exploitation	HSM	522	5.1.1.1
2.10.4	Vérifier que les mots de passe, les intégrations avec les bases de données et les systèmes tiers, les seeds et les secrets internes, ainsi que les clés API sont gérés de manière sécurisée et ne sont pas inclus dans le code source ou stockés dans des dépôts de code source. Ce type de stockage DEVRAIT résister aux attaques hors ligne. L'utilisation d'un stockage de clés logiciel sécurisé (L1), d'un module de plate-forme de confiance (TPM) ou d'un module de sécurité matériel (L3) est recommandée pour le stockage des mots de passe.		OS assisté	HSM	798	

Exigences supplémentaires des agences américaines

Les agences américaines ont des exigences obligatoires concernant le NIST 800-63. La norme de vérification de la sécurité des applications a toujours été d'environ 80% des contrôles qui s'appliquent à près de 100% des applications, et non les derniers 20% des contrôles avancés ou ceux qui ont une applicabilité limitée. En tant que telle, l'ASVS est un sous-ensemble strict de la norme NIST 800-63, en particulier pour les classifications IAL1/2 et AAL1/2, mais elle n'est pas suffisamment complète, notamment en ce qui concerne les classifications IAL3/AAL3.

Nous demandons instamment aux agences gouvernementales américaines de revoir et de mettre en œuvre la norme NIST 800-63 dans son intégralité.

Glossaire des termes

Terme	Signification
CSP	Credential Service Provider également appelé fournisseur d'identité
Authenticator	Code qui authentifie un mot de passe, un jeton, un MFA, une affirmation fédérée, etc.
Vérificateur	"Entité qui vérifie l'identité du demandeur en vérifiant la possession et le contrôle par le demandeur d'un ou deux authentificateurs au moyen d'un protocole d'authentification. Pour ce faire, le vérificateur peut également avoir besoin de valider les références qui lient le ou les authentificateurs à l'identifiant de l'abonné et vérifier leur statut"
OTP	Mot de passe unique
SFA	Les authentificateurs à facteur unique, tels que quelque chose que vous connaissez (secrets mémorisés, mots de passe, phrases de passe, codes PIN), quelque chose que vous êtes (biométrie, empreintes digitales, scanners du visage), ou quelque chose que vous avez (jetons OTP, un dispositif cryptographique tel qu'une carte à puce),
MFA	Authentification multi-facteurs, qui comprend deux ou plusieurs facteurs uniques

Références

Pour plus d'informations, voir aussi :

- NIST 800-63 - Digital Identity Guidelines
- NIST 800-63 A - Enrollment and Identity Proofing
- NIST 800-63 B - Authentication and Lifecycle Management
- NIST 800-63 C - Federation and Assertions
- NIST 800-63 FAQ
- OWASP Testing Guide 4.0: Testing for Authentication
- OWASP Cheat Sheet - Password storage
- OWASP Cheat Sheet - Forgot password
- OWASP Cheat Sheet - Choosing and using security questions

V3 : Exigences de vérification de la gestion des sessions

Objectif de contrôle

L'une des composantes essentielles de toute application web ou API à état est le mécanisme par lequel elle contrôle et maintient l'état pour un utilisateur ou un dispositif qui interagit avec elle. La gestion de session transforme un protocole sans état en protocole avec état, ce qui est essentiel pour différencier les différents utilisateurs ou appareils.

Assurez-vous qu'une application vérifiée satisfait aux exigences de gestion de session de haut niveau suivantes :

- Les sessions sont uniques à chaque individu et ne peuvent être devinées ou partagées.
- Les sessions sont invalidées lorsqu'elles ne sont plus nécessaires et sont interrompues pendant les périodes d'inactivité.

Comme indiqué précédemment, ces exigences ont été adaptées pour constituer un sous-ensemble conforme de contrôles NIST 800-63b sélectionnés, axés sur les menaces communes et les faiblesses d'authentification couramment exploitées. Les exigences de vérification précédentes ont été supprimées, réduites ou, dans la plupart des cas, adaptées pour être fortement alignées sur l'intention des exigences obligatoires NIST 800-63b.

Exigences de vérification de sécurité

V3.1 Exigences fondamentales en matière de gestion des sessions

#	Description	L1	L2	L3	CWE	NIST
3.1.1	Vérifiez que l'application ne révèle jamais les jetons de session dans les paramètres d'URL ou les messages d'erreur.	✓	✓	✓	598	

V3.2 Exigences contraignantes de la session

#	Description	L1	L2	L3	CWE	NIST
3.2.1	Vérifiez que l'application génère un nouveau jeton de session sur l'authentification de l'utilisateur. (C6)	✓	✓	✓	384	7.1
3.2.2	Vérifiez que les jetons de session possèdent au moins 128 bits d'entropie. (C6)	✓	✓	✓	331	7.1
3.2.3	Vérifiez que l'application ne stocke que des jetons de session dans le navigateur en utilisant des méthodes sûres telles que les cookies correctement sécurisés (voir section 3.4) ou le stockage de session HTML 5.	✓	✓	✓	539	7.1
3.2.4	Vérifiez que les jetons de session sont générés à l'aide d'algorithmes cryptographiques approuvés. (C6)		✓	✓	331	7.1

Le TLS ou un autre canal de transport sécurisé est obligatoire pour la gestion des sessions. Cette question est traitée dans le chapitre sur la sécurité des communications.

V3.3 Exigences en matière de déconnexion et de temporisation des sessions

Les durées de session ont été alignées sur la norme NIST 800-63, qui autorise des durées de session beaucoup plus longues que celles traditionnellement autorisées par les normes de sécurité. Les organisations doivent examiner le tableau ci-dessous, et si un délai plus long est souhaitable en fonction du risque de l'application, la valeur NIST doit être la limite supérieure des délais d'inactivité de la session.

Dans ce contexte, la valeur L1 est IAL1/AAL1, la valeur L2 est IAL2/AAL3, la valeur L3 est IAL3/AAL3. Pour IAL2/AAL2 et IAL3/AAL3, le délai d'inactivité le plus court est la limite inférieure des délais d'inactivité pour être déconnecté ou ré-authentifié pour reprendre la session.

#	Description	L1	L2	L3	CWE	NIST
3.3.1	Vérifiez que la déconnexion et l'expiration invalident le jeton de session. (C6)	✓	✓	✓	613	7.1
3.3.2	Si les authentificateurs permettent aux utilisateurs de rester connectés, vérifiez que la ré-authentification a lieu périodiquement, que ce soit en cas d'utilisation active ou après une période d'inactivité. (C6)	30 jours	12 heures ou 30 minutes d'inactivité, 2FA facultatif	12 heures ou 15 minutes d'inactivité, avec 2FA	613	7.2
3.3.3	Vérifiez que l'application offre la possibilité de mettre fin à toutes les autres sessions actives après un changement de mot de passe réussi (y compris le changement par réinitialisation/récupération du mot de passe), et que cette option est effective dans toute l'application, la connexion fédérée (si elle existe) et toute partie qui se fie à elle.		✓	✓	613	
3.3.4	Vérifiez que les utilisateurs sont en mesure de consulter et (après avoir saisi à nouveau leurs identifiants de connexion) de se déconnecter d'une ou de toutes les sessions et de tous les dispositifs actuellement actifs.		✓	✓	613	7.1

V3.4 Gestion de session basée sur les cookies

#	Description	L1	L2	L3	CWE	NIST
3.4.1	Vérifiez que les jetons de session basés sur des cookies ont l'attribut "Secure". (C6)	✓	✓	✓	614	7.1.1
3.4.2	Vérifiez que les jetons de session basés sur des cookies ont l'attribut "HttpOnly". (C6)	✓	✓	✓	1004	7.1.1
3.4.3	Vérifiez que les jetons de session basés sur des cookies utilisent l'attribut "SameSite" pour limiter l'exposition aux attaques de contrefaçon par requête intersite. (C6)	✓	✓	✓	16	7.1.1
3.4.4	Vérifiez que les jetons de session basés sur les cookies utilisent le préfixe "__Host" (voir références) pour assurer la confidentialité des cookies de session.	✓	✓	✓	16	7.1.1

#	Description	L1	L2	L3	CWE	NIST
3.4.5	Vérifiez que si la demande est publiée sous un nom de domaine avec d'autres applications qui définissent ou utilisent des cookies de session susceptibles de les remplacer ou de les divulguer, définissez l'attribut de chemin dans les jetons de session basés sur les cookies en utilisant le chemin le plus précis possible. (C6)	✓	✓	✓	16	7.1.1

V3.5 Gestion de session à jetons

La gestion des sessions basée sur des jetons comprend les clés JWT, OAuth, SAML et API. Parmi celles-ci, les clés API sont connues pour être faibles et ne doivent pas être utilisées dans un nouveau code.

#	Description	L1	L2	L3	CWE	NIST
3.5.1	Vérifiez que l'application ne valide pas les jetons OAuth et refresh -- par eux-même -- comme la présence de l'abonné et permet aux utilisateurs de mettre fin aux relations de confiance avec les applications liées.		✓	✓	290	7.1.2
3.5.2	Vérifiez que l'application utilise des jetons de session plutôt que des secrets et des clés d'API statiques, sauf dans le cas d'anciennes implémentations(legacy).		✓	✓	798	
3.5.3	Vérifiez que les jetons de session sans état utilisent les signatures numériques, le cryptage et d'autres contre-mesures pour se protéger contre les attaques par altération, mise sous enveloppe, rediffusion, chiffrement nul et substitution de clé.		✓	✓	345	

V3.6 Re-authentification d'une fédération ou d'une assertion

Cette section concerne les personnes qui écrivent le code de la partie de relais (RP) ou du fournisseur de services d'accréditation (CSP). Si vous comptez sur un code mettant en œuvre ces caractéristiques, assurez-vous que ces questions sont traitées correctement.

#	Description	L1	L2	L3	CWE	NIST
3.6.1	Vérifier que les parties qui se fient à la procédure précisent le délai maximal d'authentification aux fournisseurs de services d'authentification (CSP) et que ces derniers ré-authentifient l'abonné s'ils n'ont pas utilisé de session pendant cette période.			✓	613	7.2.1
3.6.2	Vérifier que les fournisseurs de services d'accréditation (CSP) informent les parties ayant fait confiance au dernier événement d'authentification, afin de permettre aux RP de déterminer s'ils doivent ré-authentifier l'utilisateur.			✓	613	7.2.1

V3.7 Défenses contre l'exploitation de la gestion des sessions

Il existe un petit nombre d'attaques de gestion de session, dont certaines sont liées à l'expérience utilisateur (UX) des sessions. Auparavant, sur la base des exigences de la norme ISO 27002, l'ASVS exigeait le blocage de plusieurs sessions simultanées. Le blocage de sessions simultanées n'est plus approprié, non seulement parce que les utilisateurs modernes disposent de nombreux appareils ou que l'application est une API sans session de navigateur, mais aussi parce que dans la plupart de ces implémentations, le dernier authentificateur gagne, qui est souvent l'attaquant. Cette section fournit des conseils de premier plan sur la dissuasion, le retard et la détection des attaques de gestion de session à l'aide de code.

Description de l'attaque semi-ouverte

Au début de 2018, plusieurs institutions financières ont été compromises par ce que les attaquants ont appelé des "attaques à demi ouvertes". Ce terme est resté dans l'industrie. Les attaquants ont frappé plusieurs institutions avec des bases de code propriétaires différentes, et en effet il semble que les bases de code soient différentes au sein des mêmes institutions. L'attaque semi-ouverte exploite un défaut de conception communément rencontré dans de nombreux systèmes d'authentification, de gestion de session et de contrôle d'accès existants.

Les attaquants lancent une attaque à demi-ouverte en essayant de verrouiller, de réinitialiser ou de récupérer un justificatif d'identité. Un modèle de gestion de session très répandu réutilise les objets/modèles de session de profil utilisateur entre le code non authentifié, semi-authentifié (réinitialisation du mot de passe, nom d'utilisateur oublié) et entièrement authentifié. Ce modèle remplit un objet de session ou un jeton valide contenant le profil de la victime, y compris les hachages de mots de passe et les rôles. Si les contrôles d'accès dans les contrôleurs ou les routeurs ne vérifient pas correctement que l'utilisateur est bien connecté, l'attaquant pourra agir en tant qu'utilisateur. Les attaques peuvent inclure la modification du mot de passe de l'utilisateur à une valeur connue, la mise à jour de l'adresse électronique pour effectuer une réinitialisation de mot de passe valide, la désactivation de l'authentification multifactorielle ou l'inscription d'un nouveau dispositif d'AMF, la révélation ou la modification des clés API, etc.

#	Description	L1	L2	L3	CWE	NIST
3.7.1	Vérifier que l'application garantit une session de connexion complète et valide ou exige une nouvelle authentification ou une vérification secondaire avant d'autoriser toute transaction sensible ou modification de compte.	✓	✓	✓	306	

Références

Pour plus d'informations, voir aussi :

- OWASP Testing Guide 4.0: Session Management Testing
- OWASP Session Management Cheat Sheet
- Set-Cookie __Host- prefix details

V4 : Exigences de vérification du contrôle d'accès

Objectif de contrôle

L'autorisation est le concept qui consiste à ne permettre l'accès aux ressources qu'à ceux qui sont autorisés à les utiliser. Assurez-vous qu'une application vérifiée satisfait aux exigences de haut niveau suivantes :

- Les personnes qui accèdent aux ressources possèdent une autorisation valide pour le faire.
- Les utilisateurs sont associés à un ensemble bien défini de rôles et de privilèges.
- Les métadonnées relatives aux rôles et aux autorisations sont protégées contre toute rediffusion ou altération.

Exigences de vérification de la sécurité

V4.1 Conception générale du contrôle d'accès

#	Description	L1	L2	L3	CWE
4.1.1	Vérifiez que l'application applique les règles de contrôle d'accès sur une couche de service de confiance, en particulier si le contrôle d'accès côté client est présent et pourrait être contourné.	✓	✓	✓	602
4.1.2	Vérifier que tous les attributs des utilisateurs et des données et les informations sur les politiques utilisées par les contrôles d'accès ne peuvent être manipulés par les utilisateurs finaux, sauf autorisation spécifique.	✓	✓	✓	639
4.1.3	Vérifier que le principe du moindre privilège existe - les utilisateurs ne doivent pouvoir accéder qu'aux fonctions, fichiers de données, URL, contrôleurs, services et autres ressources pour lesquels ils possèdent une autorisation spécifique. Cela implique une protection contre l'usurpation et l'élévation des privilèges. (C7)	✓	✓	✓	285
4.1.4	Vérifiez que le principe de refus par défaut existe, selon lequel les nouveaux utilisateurs/rôles commencent avec des autorisations minimales ou nulles et les utilisateurs/rôles ne reçoivent pas l'accès aux nouvelles fonctionnalités tant que l'accès n'est pas explicitement attribué. (C7)	✓	✓	✓	276
4.1.5	Vérifier que les contrôles d'accès échouent de manière sûre, y compris lorsqu'une exception se produit. (C10)	✓	✓	✓	285

V4.2 Contrôle d'accès au niveau des opérations

#	Description	L1	L2	L3	CWE
4.2.1	Vérifier que les données sensibles et les API sont protégées contre les attaques par référence directe à un objet (IDOR) non sécurisées visant la création, la lecture, la mise à jour et la suppression d'enregistrements, telles que la création ou la mise à jour de l'enregistrement de quelqu'un d'autre, la consultation de tous les enregistrements ou la suppression de tous les enregistrements.	✓	✓	✓	639
4.2.2	Vérifiez que l'application ou le cadriciel applique un mécanisme anti-CSRF fort pour protéger les fonctionnalités authentifiées, et qu'une anti-automation ou un anti-CSRF efficace protège les fonctionnalités non authentifiées.	✓	✓	✓	352

V4.3 Autres considérations relatives au contrôle d'accès

#	Description	L1	L2	L3	CWE
4.3.1	Vérifier que les interfaces administratives utilisent une authentification multifactorielle appropriée pour empêcher toute utilisation non autorisée.	✓	✓	✓	419
4.3.2	Vérifiez que la navigation dans les répertoires est désactivée, sauf si vous le souhaitez délibérément. En outre, les applications ne doivent pas permettre la découverte ou la divulgation de métadonnées de fichiers ou de répertoires, tels que les dossiers Thumbs.db, .DS_Store, .git ou .svn.	✓	✓	✓	548
4.3.3	Vérifier que la demande dispose d'une autorisation supplémentaire (telle qu'une authentification renforcée ou adaptative) pour les systèmes à faible valeur, et/ou d'une séparation des tâches pour les demandes à valeur élevée afin de faire appliquer les contrôles anti-fraude en fonction du risque de fraude de la demande et de la fraude passée.		✓	✓	732

Références

Pour plus d'informations, voir aussi :

- OWASP Testing Guide 4.0: Authorization
- OWASP Cheat Sheet: Access Control
- OWASP CSRF Cheat Sheet
- OWASP REST Cheat Sheet

V5 : Exigences de validation, d'assainissement et de vérification de l'encodage

Objectif de contrôle

La faiblesse la plus courante en matière de sécurité des applications web est l'incapacité à valider correctement les données provenant du client ou de l'environnement avant de les utiliser directement sans aucun encodage de sortie. Cette faiblesse est à l'origine de presque toutes les vulnérabilités importantes des applications web, telles que le Cross-Site Scripting (XSS), l'injection SQL, l'injection d'interpréteur, les attaques locales/Unicode, les attaques de système de fichiers et les débordements de mémoire tampon.

Assurez-vous qu'une application vérifiée satisfait aux exigences de haut niveau suivantes :

- La validation des entrées et l'architecture de codage des sorties ont un pipeline convenu pour prévenir les attaques par injection.
- Les données d'entrée sont fortement typées, validées, vérifiées en plage ou en longueur ou, au pire, aseptisées ou filtrées.
- Les données de sortie sont codées ou échappées selon le contexte des données, aussi près que possible de l'interpréteur.

Avec l'architecture moderne des applications web, l'encodage des données de sortie est plus important que jamais. Il est difficile de fournir une validation robuste des entrées dans certains scénarios, de sorte que l'utilisation d'API plus sûres telles que les requêtes paramétrées, les cadres de modélisation à évasion automatique ou un encodage de sortie soigneusement choisi est essentiel pour la sécurité de l'application.

V5.1 Exigences de validation des entrées

Des contrôles de validation des entrées correctement mis en œuvre, utilisant une liste d'autorisation positive et un fort typage des données, peuvent éliminer plus de 90 % de toutes les attaques par injection. Les contrôles de longueur et de portée peuvent encore réduire ce phénomène. Il est nécessaire de mettre en place une validation d'entrée sécurisée pendant l'architecture de l'application, les sprints de conception, le codage et les tests unitaires et d'intégration. Bien que nombre de ces éléments ne puissent être trouvés dans les tests de pénétration, les résultats de leur non-implantation se trouvent généralement dans la version 5.3 - Exigences en matière de codage de sortie et de prévention des injections. Il est recommandé aux développeurs et aux réviseurs de codes sécurisés de traiter cette section comme si la L1 était requise pour tous les éléments afin de prévenir les injections.

#	Description	L1	L2	L3	CWE
5.1.1	Vérifiez que l'application dispose de défenses contre les attaques de pollution des paramètres HTTP, en particulier si le cadre de l'application ne fait aucune distinction quant à la source des paramètres de la requête (GET, POST, cookies, en-têtes ou variables d'environnement).	✓	✓	✓	235
5.1.2	Vérifiez que les cadriciels protègent contre les attaques par assignation massive de paramètres, ou que l'application dispose de contre-mesures pour protéger contre l'assignation dangereuse de paramètres, comme le marquage des champs privés ou similaires. (C5)	✓	✓	✓	915
5.1.3	Vérifiez que toutes les entrées (champs de formulaire HTML, demandes REST, paramètres URL, en-têtes HTTP, cookies, fichiers batch, flux RSS, etc) sont validées par une validation positive (liste d'autorisation). (C5)	✓	✓	✓	20

#	Description	L1	L2	L3	CWE
5.1.4	Vérifier que les données structurées sont fortement typées et validées par rapport à un schéma défini comprenant les caractères, la longueur et le modèle autorisés (par exemple, numéros de carte de crédit ou de téléphone, ou valider que deux champs connexes sont raisonnables, comme vérifier la correspondance entre la banlieue et le code postal). (C5)	✓	✓	✓	20
5.1.5	Vérifiez que les redirections et les transferts d'URL n'autorisent que les destinations prévu, ou affichez un avertissement lors d'une redirection vers un contenu potentiellement non fiable.	✓	✓	✓	601

V5.2 Exigences en matière d'assainissement et de « bac à sable »

#	Description	L1	L2	L3	CWE
5.2.1	Vérifiez que toutes les entrées HTML non fiables provenant d'éditeurs WYSIWYG ou similaires sont correctement assainit avec une bibliothèque ou une fonction de framework de nettoyage HTML. (C5)	✓	✓	✓	116
5.2.2	Vérifiez que les données non structurées sont assainit afin d'appliquer les mesures de sécurité telles que les caractères et la longueur autorisés.	✓	✓	✓	138
5.2.3	Vérifiez que l'application assainit les entrées de l'utilisateur avant de passer aux systèmes de messagerie pour protéger contre l'injection SMTP ou IMAP.	✓	✓	✓	147
5.2.4	Vérifiez que l'application évite l'utilisation de eval() ou d'autres fonctions d'exécution de code dynamique. Lorsqu'il n'y a pas d'alternative, toute entrée utilisateur incluse doit être assainit ou mise en sandbox avant d'être exécutée.	✓	✓	✓	95
5.2.5	Vérifiez que l'application protège contre les attaques par injection de modèles en veillant à ce que toute entrée de l'utilisateur incluse soit aseptisée ou mise en bac à sable.	✓	✓	✓	94
5.2.6	Vérifier que l'application protège contre les attaques SSRF, en validant ou en assainissant les données non fiables ou les métadonnées de fichiers HTTP, comme les noms de fichiers et les champs de saisie d'URL, utiliser la liste d'autorisation des protocoles, domaines, chemins et ports.	✓	✓	✓	918
5.2.7	Vérifiez que l'application assainit, désactive ou met en place une isolation (sandbox) pour le contenu scriptable fourni par l'utilisateur (Scalable Vector Graphics - SVG), en particulier en ce qui concerne les XSS résultant de scripts natif au code présent et foreignObject.	✓	✓	✓	159
5.2.8	Vérifiez que l'application assainit, désactive ou met en sandbox le contenu des scripts ou des modèles d'expression fournis par l'utilisateur, tels que les feuilles de style Markdown, CSS ou XSL, le BBCode ou autres.	✓	✓	✓	94

V5.3 Exigences en matière d'encodage de sortie et de prévention des injections

L'encodage de la sortie à proximité de l'interprète utilisé est essentiel pour la sécurité de toute application. Généralement, l'encodage de sortie n'est pas persistant, mais utilisé pour rendre la sortie sûre dans le contexte de sortie approprié pour une utilisation immédiate. Le défaut d'encodage de la sortie entraînera une application non sécurisée, injectable et dangereuse.

#	Description	L1	L2	L3	CWE
5.3.1	Vérifiez que l'encodage de sortie est pertinent pour l'interprète et le contexte requis. Par exemple, utilisez des encodeurs spécifiques pour les valeurs HTML, les attributs HTML, JavaScript, les paramètres URL, les en-têtes HTTP, SMTP et autres selon le contexte, en particulier à partir d'entrées non fiables (par exemple les noms avec Unicode ou apostrophes, comme ねこ ou O'Hara). (C4)	✓	✓	✓	116
5.3.2	Vérifiez que l'encodage de sortie préserve le jeu de caractères et la langue choisis par l'utilisateur, de sorte que tout point de caractère Unicode soit valide et traité en toute sécurité. (C4)	✓	✓	✓	176
5.3.3	Vérifiez que l'encodage des sorties en fonction du contexte, de préférence automatisé - ou au pire, manuel - protège contre le XSS réfléchi, stocké et basé sur le DOM. (C4)	✓	✓	✓	79
5.3.4	Vérifier que la sélection de données ou les requêtes de base de données (par exemple SQL, HQL, ORM, NoSQL) utilisent des requêtes paramétrées, des ORM, des cadres d'entités, ou sont autrement protégées contre les attaques par injection de base de données. (C3)	✓	✓	✓	89
5.3.5	Vérifiez que, lorsque des mécanismes paramétrés ou plus sûrs ne sont pas présents, un encodage de sortie spécifique au contexte est utilisé pour se protéger contre les attaques par injection, comme l'utilisation de l'échappement SQL pour se protéger contre l'injection SQL. (C3, C4)	✓	✓	✓	89
5.3.6	Vérifiez que l'application protège contre les attaques par injection de JavaScript ou de JSON, y compris pour les attaques d'évaluation, les includes JavaScript distants, les contournements de la politique de sécurité du contenu (CSP), les DOM XSS et l'évaluation des expressions JavaScript. (C4)	✓	✓	✓	830
5.3.7	Vérifiez que l'application protège contre les vulnérabilités de LDAP Injection, ou que des contrôles de sécurité spécifiques pour empêcher des injections LDAP ont été mis en place. (C4)	✓	✓	✓	90
5.3.8	Vérifiez que l'application protège contre l'injection de commandes du système d'exploitation et que les appels du système d'exploitation utilisent des requêtes paramétrées du système d'exploitation ou utilisent l'encodage contextuel de la sortie de la ligne de commande. (C4)	✓	✓	✓	78
5.3.9	Vérifiez que l'application protège contre les attaques par inclusion de fichier local (LFI) ou par inclusion de fichier distant (RFI).	✓	✓	✓	829
5.3.10	Vérifiez que l'application protège contre les attaques par injection XPath ou par injection XML. (C4)	✓	✓	✓	643

Note : L'utilisation de requêtes paramétrées ou l'échappement du SQL n'est pas toujours suffisant ; les noms de tables et de colonnes, ORDER BY, etc. ne peuvent pas être échappés. L'inclusion de données échappées fournies par l'utilisateur dans ces champs entraîne l'échec des requêtes ou de l'injection SQL.

Note : Le format SVG autorise explicitement le script ECMA dans presque tous les contextes, de sorte qu'il peut ne pas être possible de bloquer complètement tous les vecteurs XSS SVG. Si un téléchargement SVG est

nécessaire, nous recommandons fortement soit de servir ces fichiers téléchargés en tant que texte/plain, soit d'utiliser un domaine de contenu distinct fourni par l'utilisateur pour empêcher que le XSS ne prenne le relais de l'application.

V5.4 Exigences en matière de mémoire, de chaînes de caractères et de code non géré

Les exigences suivantes ne s'appliquent que lorsque l'application utilise un langage système ou un code non géré.

#	Description	L1	L2	L3	CWE
5.4.1	Vérifiez que l'application utilise une chaîne de caractères à mémoire sécurisée, une copie mémoire sécurisée et l'arithmétique des pointeurs pour détecter ou empêcher les débordements de pile, de mémoire tampon ou de tas.		✓	✓	120
5.4.2	Vérifiez que les chaînes de format ne prennent pas d'entrée potentiellement hostile, et sont constantes.		✓	✓	134
5.4.3	Vérifiez que les techniques de validation des signes, des plages et des entrées sont utilisées pour éviter les débordements d'entiers.		✓	✓	190

V5.5 Exigences de prévention de la désérialisation

#	Description	L1	L2	L3	CWE
5.5.1	Vérifiez que les objets sérialisés utilisent des contrôles d'intégrité ou sont cryptés pour empêcher la création d'objets hostiles ou la falsification de données. (C5)	✓	✓	✓	502
5.5.2	Vérifiez que l'application restreint correctement les analyseurs XML pour n'utiliser que la configuration la plus restrictive possible et pour s'assurer que les fonctions dangereuses telles que la résolution d'entités externes sont désactivées pour empêcher les XXE.	✓	✓	✓	611
5.5.3	Vérifiez que la désérialisation des données non fiables est évitée ou protégée à la fois dans le code personnalisé et les bibliothèques tierces (comme les analyseurs JSON, XML et YAML).	✓	✓	✓	502
5.5.4	Vérifiez que lors de l'analyse de JSON dans les navigateurs ou les backends basés sur JavaScript, JSON.parse est utilisé pour analyser le document JSON. N'utilisez pas eval() pour analyser JSON.	✓	✓	✓	95

Références

Pour plus d'informations, voir aussi :

- OWASP Testing Guide 4.0: Input Validation Testing
- OWASP Cheat Sheet: Input Validation
- OWASP Testing Guide 4.0: Testing for HTTP Parameter Pollution
- OWASP LDAP Injection Cheat Sheet
- OWASP Testing Guide 4.0: Client Side Testing
- OWASP Cross Site Scripting Prevention Cheat Sheet
- OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet
- OWASP Java Encoding Project

- OWASP Mass Assignment Prevention Cheat Sheet
- DOMPurify - Client-side HTML Sanitization Library
- XML External Entity (XXE) Prevention Cheat Sheet

Pour plus d'informations sur l'évasion automatique, veuillez consulter

- Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems
- AngularJS Strict Contextual Escaping
- AngularJS ngBind
- Angular Sanitization
- Angular Template Security
- ReactJS Escaping
- Improperly Controlled Modification of Dynamically-Determined Object Attributes

Pour plus d'informations sur la désérialisation, veuillez consulter

- OWASP Deserialization Cheat Sheet
- OWASP Deserialization of Untrusted Data Guide

V6 : Exigences de vérification de la cryptographie stockée

Objectif de contrôle

Assurez-vous qu'une application vérifiée satisfait aux exigences de haut niveau suivantes :

- Tous les modules cryptographiques échouent de manière sécurisée et que les erreurs sont traitées correctement.
- Un générateur de nombres aléatoires approprié est utilisé.
- L'accès aux clés est géré de manière sécurisée.

V6.1 Classification des données

L'actif le plus important est constitué par les données traitées, stockées ou transmises par une application. Il faut toujours procéder à une évaluation de l'impact sur la vie privée afin de classer correctement les besoins en matière de protection des données de toute donnée stockée.

#	Description	L1	L2	L3	CWE
6.1.1	Vérifier que les données privées réglementées sont stockées sous forme cryptée pendant le repos, comme les informations d'identification personnelle (IIP), les informations personnelles sensibles ou les données considérées comme susceptibles d'être soumises à la GDPR de l'UE.		✓	✓	311
6.1.2	Vérifier que les données de santé réglementées sont stockées de manière cryptée pendant le repos, comme les dossiers médicaux, les détails des dispositifs médicaux ou les dossiers de recherche désanonymisés.		✓	✓	311
6.1.3	Vérifiez que les données financières réglementées sont stockées de manière cryptée lorsqu'elles sont au repos, telles que les comptes financiers, les défauts ou les antécédents de crédit, les dossiers fiscaux, l'historique des salaires, les bénéficiaires ou les dossiers de marché ou de recherche désanonymisés.		✓	✓	311

V6.2 Algorithmes

Les récents progrès de la cryptographie signifient que des algorithmes et des longueurs de clé auparavant sûrs ne sont plus sûrs ou suffisants pour protéger les données. Il devrait donc être possible de modifier les algorithmes.

Bien que cette section ne soit pas facilement testée lors des tests d'intrusions, les développeurs devraient considérer toute cette section comme obligatoire même si la L1 est absente de la plupart des éléments.

#	Description	L1	L2	L3	CWE
6.2.1	Vérifiez que tous les modules cryptographiques échouent en toute sécurité, et que les erreurs sont traitées de manière à ne pas permettre les attaques de type "Padding Oracle".	✓	✓	✓	310
6.2.2	Vérifiez que des algorithmes, des bibliothèques cryptographiques et des modes éprouvés par l'industrie ou approuvés par le gouvernement sont utilisés, au lieu de la cryptographie codée sur mesure. (C8)		✓	✓	327
6.2.3	Vérifiez que le vecteur d'initialisation du chiffrement, la configuration du chiffrement et les modes de blocage sont configurés de manière sécurisée en utilisant les derniers conseils.		✓	✓	326

#	Description	L1	L2	L3	CWE
6.2.4	Vérifiez que les algorithmes de chiffrement ou de hachage, les longueurs de clé, le nombre de rondes, les chiffrements ou les modes, peuvent être reconfigurés, mis à niveau ou échangés à tout moment, pour se protéger contre les failles cryptographiques. (C8)		✓	✓	326
6.2.5	Vérifiez que les modes de blocs non sécurisés connus (c'est-à-dire ECB, etc.), les modes de remplissage (c'est-à-dire PKCS#1 v1.5, etc.), les chiffrements avec des blocs de petites tailles (c'est-à-dire Triple-DES, Blowfish, etc.) et les algorithmes de hachage faibles (c'est-à-dire MD5, SHA1, etc.) ne sont pas utilisés, sauf si cela est nécessaire pour la rétrocompatibilité.		✓	✓	326
6.2.6	Vérifiez que les nonces, vecteurs d'initialisation et autres numéros à usage unique ne doivent pas être utilisés plus d'une fois avec une clé de cryptage donnée. La méthode de génération doit être appropriée à l'algorithme utilisé.		✓	✓	326
6.2.7	Vérifier que les données cryptées sont authentifiées par des signatures, des modes de chiffrement authentifiés ou le HMAC pour s'assurer que le texte chiffré n'est pas altéré par une partie non autorisée.			✓	326
6.2.8	Vérifiez que toutes les opérations cryptographiques sont à temps constant, sans opérations de "court-circuit" dans les comparaisons, les calculs ou les retours, afin d'éviter les fuites d'informations.			✓	385

V6.3 Valeurs aléatoires

La véritable génération de nombres pseudo-aléatoires (PRNG) est incroyablement difficile à réaliser. En général, les bonnes sources d'entropie au sein d'un système seront rapidement épuisées si elles sont trop utilisées, mais des sources moins aléatoires peuvent conduire à des clés et des secrets prévisibles.

#	Description	L1	L2	L3	CWE
6.3.1	Vérifiez que tous les nombres aléatoires, noms de fichiers aléatoires, GUIDs aléatoires et chaînes aléatoires sont générés en utilisant le générateur de nombres aléatoires sécurisé cryptographiquement approuvé par le module cryptographique lorsque ces valeurs aléatoires sont destinées à ne pas être devinées par un attaquant.		✓	✓	338
6.3.2	Vérifiez que les GUID aléatoires sont créés en utilisant l'algorithme GUID v4, et un générateur de nombres pseudo-aléatoires sécurisé cryptographiquement (CSPRNG). Les GUID créés à l'aide d'autres générateurs de nombres pseudo-aléatoires peuvent être prévisibles.		✓	✓	338
6.3.3	Vérifiez que les nombres aléatoires sont créés avec une entropie correcte même lorsque l'application est soumise à une forte charge, ou que l'application se dégrade gracieusement dans de telles circonstances.			✓	338

V6.4 Gestion du secret

Bien que cette section ne soit pas facilement testée, les développeurs devraient considérer toute cette section comme obligatoire même si la L1 est absente de la plupart des éléments.

#	Description	L1	L2	L3	CWE
6.4.1	Vérifiez qu'une solution de gestion des secrets, telle qu'un coffre fort de clés, est utilisé pour créer, stocker, contrôler l'accès aux secrets et les détruire en toute sécurité. (C8)		✓	✓	798
6.4.2	Vérifiez que le matériel clé ne soit pas exposé à l'application mais utilise plutôt un module de sécurité isolé comme un coffre-fort pour les opérations cryptographiques. (C8)		✓	✓	320

Références

Pour plus d'informations, voir aussi :

- OWASP Testing Guide 4.0: Testing for weak Cryptography
- OWASP Cheat Sheet: Cryptographic Storage
- FIPS 140-2

V7 : Traitement des erreurs et exigences de vérification de l'enregistrement

Objectif de contrôle

L'objectif premier du traitement et de la journalisation des erreurs est de fournir des informations utiles à l'utilisateur, aux administrateurs et aux équipes de réponse aux incidents. L'objectif n'est pas de créer des quantités massives de journaux, mais des journaux de haute qualité, avec plus de signal que de bruit rejeté.

Les journaux de haute qualité contiennent souvent des données sensibles et doivent être protégés conformément aux lois ou directives locales en matière de confidentialité des données. Cela devrait inclure :

- Ne pas collecter ou enregistrer des informations sensibles, sauf si cela est spécifiquement requis.
- Veiller à ce que toutes les informations enregistrées soient traitées de manière sûre et protégées conformément à leur classification.
- Veiller à ce que les journaux ne soient pas conservés éternellement, mais qu'ils aient une durée de vie absolue aussi courte que possible.

Si les journaux contiennent des données privées ou sensibles, dont la définition varie d'un pays à l'autre, les journaux deviennent parmis les informations les plus sensibles détenues par l'application et donc très attrayantes pour les attaquants en soi.

Il est également important de s'assurer que l'application échoue en toute sécurité et que les erreurs ne divulguent pas d'informations inutiles.

V7.1 Exigences relatives au contenu des journaux

L'enregistrement d'informations sensibles est dangereux : les journaux deviennent eux-mêmes classifiés, ce qui signifie qu'ils doivent être cryptés, faire l'objet de politiques de conservation et être divulgués lors d'audits de sécurité. Assurez-vous que seules les informations nécessaires sont conservées dans les journaux, et certainement pas les paiements, les justificatifs d'identité (y compris les jetons de session), les informations sensibles ou identifiables personnellement.

V7.1 couvre le Top 10 de l'OWASP 2017:A10. Comme 2017:A10 et cette section ne sont pas testables tests d'intrusions, il est important pour :

- Les développeurs de s'assurer de la conformité totale avec cette section, comme si tous les éléments étaient marqués comme L1
- Tests de pénétration de valider la conformité totale de tous les éléments de la V7.1 par le biais d'un entretien, de captures d'écran ou d'une affirmation

#	Description	L1	L2	L3	CWE
7.1.1	Vérifiez que la demande n'enregistre pas les références ou les détails de paiement. Les jetons de session ne doivent être stockés dans les journaux que sous une forme hachée et irréversible. (C9, C10)	✓	✓	✓	532
7.1.2	Vérifiez que l'application n'enregistre pas d'autres données sensibles telles que définies par les lois locales sur la protection de la vie privée ou la politique de sécurité pertinente. (C9)	✓	✓	✓	532
7.1.3	Vérifiez que l'application enregistre les événements pertinents pour la sécurité, y compris les événements d'authentification réussis et échoués, les échecs de contrôle d'accès, les échecs de désérialisation et les échecs de validation des entrées. (C5, C7)		✓	✓	778

#	Description	L1	L2	L3	CWE
7.1.4	Vérifiez que chaque événement consigné dans le journal contient les informations nécessaires pour permettre une enquête détaillée sur la chronologie de l'événement. (C9)		✓	✓	778

V7.2 Exigences de traitement des journaux

Il est essentiel d'enregistrer en temps utile les événements de vérification, le triage et l'escalade. Assurez-vous que les journaux de l'application sont clairs et peuvent être facilement surveillés et analysés, soit localement, soit envoyés à un système de surveillance à distance.

V7.2 couvre le Top 10 de l'OWASP 2017:A10. Comme 2017:A10 et cette section ne sont pas testables, il est important pour :

- Les développeurs de s'assurer de la conformité totale avec cette section, comme si tous les éléments étaient marqués comme L1
- Tests de pénétration de valider la conformité totale de tous les éléments de la V7.2 par le biais d'un entretien, de captures d'écran ou d'une affirmation

#	Description	L1	L2	L3	CWE
7.2.1	Vérifiez que toutes les décisions d'authentification sont consignées, sans stocker d'identifiants de session ou de mots de passe sensibles. Cela devrait inclure les demandes avec les métadonnées pertinentes nécessaires aux enquêtes de sécurité.		✓	✓	778
7.2.2	Vérifiez que toutes les décisions de contrôle d'accès peuvent être enregistrées et que toutes les décisions qui ont échoué sont enregistrées. Cela devrait inclure les demandes avec les métadonnées pertinentes nécessaires aux enquêtes de sécurité.		✓	✓	285

V7.3 Exigences en matière de protection des journaux

Les journaux qui peuvent être trivialement modifiés ou supprimés sont inutiles pour les enquêtes et les poursuites. La divulgation des journaux peut révéler des détails internes sur l'application ou les données qu'elle contient. Il convient de prendre des précautions pour protéger les journaux contre toute divulgation, modification ou suppression non autorisée.

#	Description	L1	L2	L3	CWE
7.3.1	Vérifiez que l'application encode correctement les données fournies par l'utilisateur pour éviter l'injection de logs. (C9)		✓	✓	117
7.3.2	Vérifiez que tous les événements sont protégés contre l'injection lorsqu'ils sont visualisés dans le logiciel de visualisation des journaux. (C9)		✓	✓	117
7.3.3	Vérifiez que les journaux de sécurité sont protégés contre tout accès et toute modification non autorisés. (C9)		✓	✓	200
7.3.4	Vérifiez que les sources de temps sont synchronisées avec l'heure et le fuseau horaire corrects. Envisager sérieusement de n'enregistrer les données qu'en UTC si les systèmes sont globaux pour faciliter l'analyse criminalistique post-incident. (C9)		✓	✓	

Remarque : L'encodage des journaux (7.3.1) est difficile à tester et à examiner à l'aide d'outils dynamiques automatisés et de tests de pénétration, mais les architectes, les développeurs et les réviseurs de code source devraient le considérer comme une exigence de niveau 1.

V7.4 Traitement des erreurs

L'objectif du traitement des erreurs est de permettre à l'application de fournir des événements pertinents pour la sécurité en vue de la surveillance, du triage et de l'escalade. L'objectif n'est pas de créer des journaux. Lorsque vous enregistrez des événements liés à la sécurité, assurez-vous que le journal a un but et qu'il peut être distingué par le SIEM ou un logiciel d'analyse.

#	Description	L1	L2	L3	CWE
7.4.1	Vérifiez qu'un message générique s'affiche lorsqu'une erreur inattendue ou sensible à la sécurité se produit, éventuellement avec un identifiant unique que le personnel de soutien peut utiliser pour enquêter. (C10)	✓	✓	✓	210
7.4.2	Vérifiez que le traitement des exceptions est utilisé dans toute le code source pour tenir compte des conditions d'erreur prévues et imprévues. (C10)		✓	✓	544
7.4.3	Vérifiez qu'un gestionnaire d'erreurs de "dernier recours" est défini, qui prendra en compte toutes les exceptions non traitées. (C10)		✓	✓	431

Note : Certains langages, tels que Swift et Go - et selon la pratique courante de conception - de nombreux langages fonctionnels, ne prennent pas en charge les exceptions ou les gestionnaires d'événements de dernier recours. Dans ce cas, les architectes et les développeurs doivent utiliser un modèle, un langage ou un cadre convivial pour s'assurer que les applications peuvent gérer en toute sécurité des événements exceptionnels, inattendus ou liés à la sécurité.

Références

Pour plus d'informations, voir aussi :

- OWASP Testing Guide 4.0 content: Testing for Error Handling
- OWASP Authentication Cheat Sheet section about error messages

V8 : Exigences de vérification de la protection des données

Objectif de contrôle

Il y a trois éléments clés pour une bonne protection des données : Confidentialité, intégrité et disponibilité (CIA). Cette norme suppose que la protection des données est appliquée sur un système fiable, tel qu'un serveur, qui a été renforcé et dispose de protections suffisantes.

Les applications doivent supposer que tous les dispositifs des utilisateurs sont compromis d'une manière ou d'une autre. Lorsqu'une application transmet ou stocke des informations sensibles sur des dispositifs non sécurisés, tels que des ordinateurs, des téléphones et des tablettes partagés, l'application est chargée de s'assurer que les données stockées sur ces dispositifs sont cryptées et ne peuvent pas être facilement obtenues, modifiées ou divulguées de manière illicite.

Assurez-vous qu'une application vérifiée satisfait aux exigences de haut niveau suivantes en matière de protection des données :

- Confidentialité : Les données doivent être protégées contre toute observation ou divulgation non autorisée, tant pendant leur transit que lors de leur stockage.
- Intégrité : Les données doivent être protégées contre toute création, modification ou suppression malveillante par des attaquants non autorisés.
- Disponibilité : Les données doivent être accessibles aux utilisateurs autorisés, selon les besoins.

V8.1 Protection générale des données

#	Description	L1	L2	L3	CWE
8.1.1	Vérifiez que l'application protège les données sensibles contre la mise en cache dans des composants du serveur tels que les équilibreurs de charge et les caches d'applications.		✓	✓	524
8.1.2	Vérifier que toutes les copies en cache ou temporaires de données sensibles stockées sur le serveur sont protégées contre tout accès non autorisé ou purgées/invalidées après que l'utilisateur autorisé a accédé aux données sensibles.		✓	✓	524
8.1.3	Vérifier que l'application minimise le nombre de paramètres dans une requête, tels que les champs cachés, les variables Ajax, les cookies et les valeurs d'en-tête.		✓	✓	233
8.1.4	Vérifier que l'application peut détecter et alerter sur un nombre anormal de demandes, par exemple par IP, par utilisateur, par total par heure ou par jour, ou tout ce qui a un sens pour l'application.		✓	✓	770
8.1.5	Vérifiez que des sauvegardes régulières des données importantes sont effectuées et que des tests de restauration des données sont effectués.			✓	19
8.1.6	Vérifiez que les sauvegardes sont stockées en toute sécurité pour éviter que les données ne soient volées ou corrompues.			✓	19

V8.2 Protection des données côté client

#	Description	L1	L2	L3	CWE
8.2.1	Vérifiez que l'application définit suffisamment d'en-têtes anticaching pour que les données sensibles ne soient pas mises en cache dans les navigateurs modernes.	✓	✓	✓	525

#	Description	L1	L2	L3	CWE
8.2.2	Vérifiez que les données stockées dans le stockage côté client (telles que le stockage local HTML5, le stockage de session, IndexedDB, les cookies réguliers ou les cookies Flash) ne contiennent pas de données sensibles ou d'IIP.	✓	✓	✓	922
8.2.3	Vérifiez que les données authentifiées sont effacées du stockage du client, tel que le DOM du navigateur, après la fin du client ou de la session.	✓	✓	✓	922

V8.3 Données privées sensibles

Cette section permet de protéger les données sensibles contre la création, la lecture, la mise à jour ou la suppression sans autorisation, notamment en cas de grandes quantités.

Le respect de cette section implique le respect du contrôle d'accès V4, et en particulier V4.2. Par exemple, la protection contre les mises à jour ou la divulgation non autorisées d'informations personnelles sensibles nécessite le respect de la V4.2.1. Veuillez vous conformer à cette section et à V4 pour une couverture complète.

Note : Les réglementations et les lois relatives à la protection de la vie privée, telles que "Australian Privacy Principles" APP-11 ou GDPR, ont une incidence directe sur la manière dont les applications doivent aborder la mise en œuvre du stockage, de l'utilisation et de la transmission des informations personnelles sensibles. Cela va de sanctions sévères à de simples conseils. Veuillez consulter vos lois et règlements locaux et, le cas échéant, un spécialiste ou un avocat qualifié en matière de protection de la vie privée.

#	Description	L1	L2	L3	CWE
8.3.1	Vérifiez que les données sensibles sont envoyées au serveur dans le corps ou les en-têtes du message HTTP, et que les paramètres de la chaîne de requête de tout verbe HTTP ne contiennent pas de données sensibles.	✓	✓	✓	319
8.3.2	Vérifier que les utilisateurs disposent d'une méthode pour supprimer ou exporter leurs données sur demande.	✓	✓	✓	212
8.3.3	Vérifier que les utilisateurs disposent d'un langage clair concernant la collecte et l'utilisation des informations personnelles fournies et que les utilisateurs ont donné leur consentement pour l'utilisation de ces données avant qu'elles ne soient utilisées de quelque manière que ce soit.	✓	✓	✓	285
8.3.4	Vérifier que toutes les données sensibles créées et traitées par l'application ont été identifiées, et s'assurer qu'une politique est en place sur la manière de traiter les données sensibles. (C8)	✓	✓	✓	200
8.3.5	Vérifier que l'accès aux données sensibles est contrôlé (sans enregistrer les données sensibles elles-mêmes), si les données sont collectées en vertu des directives pertinentes sur la protection des données ou si l'enregistrement de l'accès est nécessaire.		✓	✓	532
8.3.6	Vérifiez que les informations sensibles contenues dans la mémoire sont écrasées dès qu'elles ne sont plus nécessaires pour atténuer les attaques de vidage de la mémoire, en utilisant des zéros ou des données aléatoires.		✓	✓	226
8.3.7	Vérifier que les informations sensibles ou privées qui doivent être cryptées, le sont à l'aide d'algorithmes approuvés qui assurent à la fois la confidentialité et l'intégrité. (C8)		✓	✓	327

#	Description	L1	L2	L3	CWE
8.3.8	Vérifier que les informations personnelles sensibles font l'objet d'une classification de conservation des données, de sorte que les données anciennes ou périmées soient supprimées automatiquement, selon un calendrier ou selon la situation.		✓	✓	285

Lorsqu'on envisage la protection des données, il faut avant tout tenir compte de l'extraction ou de la modification de masse ou de l'utilisation excessive. Par exemple, de nombreux systèmes de médias sociaux ne permettent aux utilisateurs que d'ajouter 100 nouveaux amis par jour, mais le système d'où proviennent ces demandes n'a pas d'importance. Une plateforme bancaire peut souhaiter bloquer plus de 5 transactions par heure en transférant plus de 1000 euros de fonds vers des institutions externes. Les exigences de chaque système sont susceptibles d'être très différentes, de sorte que la décision d'être "anormal" doit tenir compte du modèle de menace et du risque commercial. Les critères importants sont la capacité de détecter, de dissuader ou, de préférence, de bloquer ces actions anormales de masse.

Références

Pour plus d'informations, voir aussi :

- Consider using Security Headers website to check security and anti-caching headers
- OWASP Secure Headers project
- OWASP Privacy Risks Project
- OWASP User Privacy Protection Cheat Sheet
- European Union General Data Protection Regulation (GDPR) overview
- European Union Data Protection Supervisor - Internet Privacy Engineering Network

V9 : Exigences de vérification des communications

Objectif de contrôle

Assurez-vous qu'une demande vérifiée satisfait aux exigences de haut niveau suivantes :

- Le TLS ou le cryptage fort est toujours utilisé, quelle que soit la sensibilité des données transmises
- Les conseils de configuration les plus récents et les plus importants sont utilisés pour activer et ordonner les algorithmes et les chiffres préférés
- Les algorithmes et les chiffres faibles ou bientôt obsolètes sont commandés en dernier recours
- Les algorithmes et les chiffres non sécurisés, dépréciés ou connus, sont désactivés.

Les principaux conseils de l'industrie sur la configuration sécurisée de TLS changent fréquemment, souvent en raison de ruptures catastrophiques dans les algorithmes et les chiffres existants. Utilisez toujours les versions les plus récentes des outils de révision de la configuration TLS (tels que SSLyze ou d'autres scanners TLS) pour configurer l'ordre et la sélection d'algorithme préférés. La configuration doit être vérifiée périodiquement pour s'assurer que la configuration des communications sécurisées est toujours présente et efficace.

V9.1 Exigences de sécurité des communications des clients

Toutes les communications avec les clients ne doivent avoir lieu que sur des voies de communication cryptées. En particulier, l'utilisation de TLS 1.2 ou d'une version ultérieure est pratiquement obligatoire pour les navigateurs et les moteurs de recherche modernes. La configuration doit être régulièrement revue à l'aide d'outils en ligne afin de s'assurer que les dernières pratiques de pointe sont en place.

#	Description	L1	L2	L3	CWE
9.1.1	Vérifiez que le TLS sécurisé est utilisé pour toutes les connexions des clients et ne revient pas à des protocoles non sécurisés ou non chiffrés. (C8)	✓	✓	✓	319
9.1.2	Vérifiez à l'aide d'outils de test TLS en ligne ou actualisés que seuls les algorithmes, les chiffrages et les protocoles puissants sont activés, les algorithmes et les chiffrages les plus puissants étant définis de préférence.	✓	✓	✓	326
9.1.3	Vérifiez que les anciennes versions des protocoles SSL et TLS, des algorithmes, des chiffres et de la configuration sont désactivées, comme SSLv2, SSLv3, ou TLS 1.0 et TLS 1.1. La dernière version de TLS doit être la suite de chiffrement préférée.	✓	✓	✓	326
9.1.4	Pour les applications de client lourd, vérifiez que l'application utilise son propre magasin de certificats, ou qu'elle épingle le certificat ou la clé publique du terminal, et qu'elle n'établira pas de connexion avec des terminaux qui offrent un certificat ou une clé différente, même si elle est signée par une AC de confiance.			✓	295

V9.2 Exigences de sécurité des communications du serveur

Les communications entre serveurs ne se limitent pas à HTTP. Des connexions sécurisées vers et depuis d'autres systèmes, tels que les systèmes de surveillance, les outils de gestion, l'accès à distance et ssh, les intergiciels, les bases de données, les ordinateurs centraux, les systèmes partenaires ou sources externes -- doivent être en place. Toutes ces connexions doivent être cryptées pour éviter "d'être difficiles à l'extérieur et trivialement faciles à intercepter à l'intérieur".

#	Description	L1	L2	L3	CWE
9.2.1	Vérifiez que les connexions vers et depuis le serveur utilisent des certificats TLS de confiance. Lorsque des certificats générés en interne ou auto-signés sont utilisés, le serveur doit être configuré pour ne faire confiance qu'à des AC internes spécifiques et à des certificats auto-signés spécifiques. Tous les autres doivent être rejetés.		✓	✓	295
9.2.2	Vérifier que les communications cryptées telles que TLS sont utilisées pour toutes les connexions entrantes et sortantes, y compris pour les ports de gestion, la surveillance, l'authentification, les appels d'API ou de service web, les connexions de base de données, de nuage, sans serveur, d'ordinateur central, externes et de partenaires. Le serveur ne doit pas se rabattre sur des protocoles non sécurisés ou non chiffrés.		✓	✓	319
9.2.3	Vérifiez que toutes les connexions cryptées à des systèmes externes qui impliquent des informations ou des fonctions sensibles sont authentifiées.		✓	✓	287
9.2.4	Vérifiez que la révocation de certification appropriée, telle que le protocol OCSP (Online Certificate Status Protocol), est activée et configurée.		✓	✓	299
9.2.5	Vérifiez que les échecs de connexion TLS en arrière-plan sont enregistrés.			✓	544

Références

Pour plus d'informations, voir aussi :

- OWASP – TLS Cheat Sheet
- OWASP - Pinning Cheat Sheet
- Remarques sur les « modes approuvés de TLS ». Dans le passé, l'ASVS faisait référence à la norme américaine FIPS 140-2, mais en tant que norme mondiale, l'application des normes américaines peut être difficile, contradictoire ou déroutante à appliquer. Une meilleure méthode pour atteindre la conformité avec 9.1.3 consisterait à examiner des guides tels que Mozilla's Server Side TLS ou generate known good configurations, et utiliser des outils d'évaluation TLS connus, tels que sslyze, divers scanners de vulnérabilité ou des services d'évaluation TLS en ligne fiables pour obtenir le niveau de sécurité souhaité. En général, nous constatons que la non-conformité de cette section est l'utilisation de chiffrements et d'algorithmes obsolètes ou non sécurisés, le manque de secret de transmission parfait, les protocoles SSL obsolètes ou non sécurisés, les chiffrements préférés faibles, etc.

V10 : Exigences de vérification des codes malveillants

Objectif de contrôle

Assurez-vous que le code satisfait aux exigences de haut niveau suivantes :

- L'activité malveillante est traitée de manière sûre et appropriée pour ne pas affecter le reste de l'application.
- Il n'y a pas de bombes à retardement ou d'autres attaques basées sur le temps.
- Ne pas "téléphoner à la maison" vers des destinations malveillantes ou non autorisées.
- Il n'y a pas de portes dérobées, d'oeufs de Pâques, d'attaques au salami, de rootkits ou de code non autorisé pouvant être contrôlé par un attaquant.

Trouver un code malveillant est une preuve du négatif, qu'il est impossible de valider complètement. Il convient de tout mettre en œuvre pour s'assurer que le code ne comporte pas de code malveillant inhérent ou de fonctionnalité indésirable.

V10.1 Contrôles de l'intégrité du code

La meilleure défense contre les codes malveillants est de "faire confiance, mais vérifier". L'introduction d'un code non autorisé ou malveillant dans un code est souvent une infraction pénale dans de nombreuses juridictions. Les politiques et les procédures doivent clairement définir les sanctions applicables aux codes malveillants.

Les principaux développeurs doivent régulièrement examiner les vérifications de code, en particulier celles qui peuvent concerner le temps d'accès, les E/S ou les fonctions réseau.

#	Description	L1	L2	L3	CWE
10.1.1	Vérifiez qu'un outil d'analyse de code est utilisé pour détecter les codes potentiellement malveillants, tels que les fonctions temporelles, les opérations de fichiers et les connexions réseau non sécurisées.			✓	749

V10.2 Recherche de code malveillant

Les codes malveillants sont extrêmement rares et difficiles à détecter. L'examen manuel ligne par ligne du code peut aider à rechercher des bombes logiques, mais même le plus expérimenté des examinateurs de code aura du mal à trouver un code malveillant même s'il sait qu'il existe.

Il n'est pas possible de se conformer à cette section sans un accès complet au code source, y compris aux bibliothèques de tiers.

#	Description	L1	L2	L3	CWE
10.2.1	Vérifiez que le code source de l'application et les bibliothèques tierces ne contiennent pas de "téléphone à la maison" ou de capacités de collecte de données non autorisées. Lorsque de telles fonctionnalités existent, obtenez l'autorisation de l'utilisateur pour leur fonctionnement avant de collecter des données.		✓	✓	359
10.2.2	Vérifiez que l'application ne demande pas d'autorisations inutiles ou excessives pour les caractéristiques ou capteurs liés à la vie privée, tels que les contacts, les caméras, les microphones ou l'emplacement.		✓	✓	272

#	Description	L1	L2	L3	CWE
10.2.3	Vérifiez que le code source de l'application et les bibliothèques tierces ne contiennent pas de portes dérobées, telles que des comptes ou des clés codées en dur ou supplémentaires non documentées, des obscurcissements de code, des blobs binaires non documentés, des rootkits, ou des fonctions de débogage anti-débogage, non sécurisées, ou encore des fonctionnalités obsolètes, non sécurisées ou cachées qui pourraient être utilisées de manière malveillante si elles étaient découvertes.			✓	507
10.2.4	Vérifiez que le code source de l'application et les bibliothèques tierces ne contiennent pas de bombes à retardement en recherchant les fonctions liées à la date et à l'heure.			✓	511
10.2.5	Vérifiez que le code source de l'application et les bibliothèques tierces ne contiennent pas de code malveillant, tel que des attaques de type salami, des contournements logiques ou des bombes logiques.			✓	511
10.2.6	Vérifiez que le code source de l'application et les bibliothèques tierces ne contiennent pas d'œufs de Pâques ou toute autre fonctionnalité potentiellement indésirable.			✓	507

V10.3 Contrôles d'intégrité des applications déployées

Une fois qu'une application est déployée, un code malveillant peut encore être inséré. Les applications doivent se protéger contre les attaques courantes, telles que l'exécution de code non signé provenant de sources non fiables et les rachats de sous-domaines.

La conformité à cette section est susceptible d'être opérationnelle et continue.

#	Description	L1	L2	L3	CWE
10.3.1	Vérifiez que si l'application dispose d'une fonction de mise à jour automatique du client ou du serveur, les mises à jour doivent être obtenues par des canaux sécurisés et signées numériquement. Le code de mise à jour doit valider la signature numérique de la mise à jour avant l'installation ou l'exécution de la mise à jour.	✓	✓	✓	16
10.3.2	Vérifiez que l'application utilise des protections d'intégrité, telles que la signature de code ou l'intégrité des sous-ressources. L'application ne doit pas charger ou exécuter du code provenant de sources non fiables, comme des includes de chargement, des modules, des plugins, du code ou des bibliothèques provenant de sources non fiables ou de l'Internet.	✓	✓	✓	353
10.3.3	Vérifiez que l'application est protégée contre les reprises de sous-domaines si elle repose sur des entrées DNS ou des sous-domaines DNS, tels que des noms de domaine expirés, des pointeurs DNS ou CNAME obsolètes, des projets expirés dans des dépôts de code source publics, ou des API de nuages transitoires, des fonctions sans serveur, ou des espaces de stockage (*autogen-bucket-id*.cloud.example.com) ou similaires. Les protections peuvent consister à s'assurer que les noms DNS utilisés par les applications sont régulièrement vérifiés pour détecter toute expiration ou modification.	✓	✓	✓	350

Références

- Hostile Subdomain Takeover, Detectify Labs
- Hijacking of abandoned subdomains part 2, Detectify Labs

V11 : Exigences de vérification de la logique d'entreprise

Objectif de contrôle

Assurez-vous qu'une demande vérifiée satisfait aux exigences de haut niveau suivantes :

- Le flux logique de l'entreprise est séquentiel, traité dans l'ordre, et ne peut être contourné.
- La logique métier comprend des limites pour détecter et prévenir les attaques automatisées, comme les petits transferts de fonds continus, ou l'ajout d'un million d'amis un à la fois, etc.
- Les flux de logique commerciale de grande valeur ont pris en compte les cas d'abus et les acteurs malveillants, et disposent de protections contre l'usurpation, l'altération, la répudiation, la divulgation d'informations et les attaques par élévation de privilèges.

V11.1 Exigences de sécurité de la logique d'entreprise

La sécurité de la logique commerciale est tellement individuelle à chaque demande qu'aucune liste de contrôle ne s'appliquera jamais. La sécurité de la logique d'entreprise doit être conçue pour protéger contre les menaces externes probables - elle ne peut pas être ajoutée en utilisant des pare-feu d'applications web ou des communications sécurisées. Nous recommandons l'utilisation de la modélisation des menaces lors des sprints de conception, par exemple en utilisant l'OWASP Cornucopia ou des outils similaires.

#	Description	L1	L2	L3	CWE
11.1.1	Vérifier que l'application traitera seulement les flux de logique métier pour un utilisateur dans l'ordre séquentiel des étapes et sans sauter d'étapes.	✓	✓	✓	841
11.1.2	Vérifier que l'application traitera seulement les flux de logiques métier, toutes les étapes étant traitées en temps humain réaliste, c'est-à-dire que les transactions ne sont pas soumises trop rapidement (effectuer par un robot).	✓	✓	✓	799
11.1.3	Vérifiez que l'application comporte des limites appropriées pour des actions ou des transactions commerciales spécifiques qui sont correctement exécutées par utilisateur.	✓	✓	✓	770
11.1.4	Vérifiez que l'application dispose de contrôles anti-automatisation suffisants pour détecter et protéger contre l'exfiltration de données, les demandes excessives de logique métiers, les téléchargements excessifs de fichiers ou les attaques par déni de service.	✓	✓	✓	770
11.1.5	Vérifier que l'application a des limites ou une validation de la logique métier pour se protéger contre les risques ou les menaces commerciales probables, identifiés à l'aide de la modélisation des menaces ou de méthodologies similaires.	✓	✓	✓	841
11.1.6	Vérifiez que la demande ne souffre pas de problèmes de "temps de contrôle au moment de l'utilisation" (TOCTOU) ou d'autres situation de compétition (race condition) pour les opérations sensibles.		✓	✓	367
11.1.7	Vérifiez que les moniteurs de demande ne présentent pas d'événements ou d'activités inhabituels du point de vue de la logique métier. Par exemple, des tentatives d'effectuer des actions hors service ou des actions qu'un utilisateur normal ne tenterait jamais. (C9)		✓	✓	754
11.1.8	Vérifiez que l'application dispose d'alertes configurables lorsque des attaques automatisées ou une activité inhabituelle sont détectées.		✓	✓	390

Références

Pour plus d'informations, voir aussi :

- OWASP Testing Guide 4.0: Business Logic Testing
- Anti-automation can be achieved in many ways, including the use of OWASP AppSensor and OWASP Automated Threats to Web Applications
- OWASP AppSensor can also help with Attack Detection and Response.
- OWASP Cornucopia

V12 : Exigences de vérification des dossiers et des ressources

Objectif de contrôle

Assurez-vous qu'une application vérifiée satisfait aux exigences de haut niveau suivantes :

- Les données des fichiers non fiables doivent être traitées en conséquence et de manière sécurisée.
- Les données de fichiers non fiables obtenues à partir de sources non fiables sont stockées en dehors de la racine web et avec des permissions limitées.

V12.1 Exigences pour le téléchargement de fichiers

Bien que les bombes zip soient facilement testables à l'aide de techniques de test de pénétration, elles sont considérées comme L2 et au-dessus pour encourager la prise en compte de la conception et du développement avec des tests manuels minutieux, et pour éviter que les tests de pénétration manuels ou automatisés engendre une condition de déni de service.

#	Description	L1	L2	L3	CWE
12.1.1	Vérifiez que la demande n'accepte pas de fichiers volumineux qui pourraient remplir l'espace de stockage ou provoquer un déni de service.	✓	✓	✓	400
12.1.2	Vérifiez que les fichiers compressés sont contrôlés pour détecter les "bombes zip" - de petits fichiers d'entrée qui se décompresseront en fichiers énormes, épuisant ainsi les limites de stockage des fichiers.		✓	✓	409
12.1.3	Vérifiez qu'un quota de taille de fichier et un nombre maximum de fichiers par utilisateur sont appliqués pour s'assurer qu'un seul utilisateur ne peut pas remplir le stockage avec trop de fichiers, ou des fichiers excessivement gros.		✓	✓	770

V12.2 Exigences en matière d'intégrité des fichiers

#	Description	L1	L2	L3	CWE
12.2.1	Vérifiez que les fichiers obtenus de sources non fiables sont validés comme étant du type attendu en fonction du contenu du fichier.		✓	✓	434

V12.3 Exigences relatives à l'exécution des fichiers

#	Description	L1	L2	L3	CWE
12.3.1	Vérifiez que les métadonnées de nom de fichier soumises par l'utilisateur ne sont pas utilisées directement par les systèmes de fichiers du système ou du cadre et qu'une API URL est utilisée pour protéger contre la traversée du chemin (path traversal).	✓	✓	✓	22
12.3.2	Vérifier que les métadonnées de nom de fichier soumises par l'utilisateur sont validées ou ignorées pour empêcher la divulgation, la création, la mise à jour ou la suppression de fichiers locaux (LFI).	✓	✓	✓	73
12.3.3	Vérifier que les métadonnées de nom de fichier soumises par l'utilisateur sont validées ou ignorées pour empêcher la divulgation ou l'exécution de fichiers distants (RFI), qui peuvent également conduire à des SSRF.	✓	✓	✓	98

#	Description	L1	L2	L3	CWE
12.3.4	Vérifiez que l'application protège contre le téléchargement de fichiers réfléchis (RFD) en validant ou en ignorant les noms de fichiers soumis par les utilisateurs dans un paramètre JSON, JSONP ou URL, l'en-tête Content-Type de la réponse doit être défini sur text/plain, et l'en-tête Content-Disposition doit avoir un nom de fichier fixe.	✓	✓	✓	641
12.3.5	Vérifier que les métadonnées de fichiers non fiables ne sont pas utilisées directement avec l'API système ou les bibliothèques, pour se protéger contre l'injection de commandes du système d'exploitation.	✓	✓	✓	78
12.3.6	Vérifiez que l'application n'inclut pas et n'exécute pas de fonctionnalités provenant de sources non fiables, telles que des réseaux de distribution de contenu non vérifiés, des bibliothèques JavaScript, des bibliothèques node npm ou des DLL côté serveur.		✓	✓	829

V12.4 Exigences en matière de stockage des fichiers

#	Description	L1	L2	L3	CWE
12.4.1	Vérifiez que les fichiers obtenus de sources non fiables sont stockés en dehors de la racine web, avec des permissions limitées, de préférence avec une validation forte.	✓	✓	✓	922
12.4.2	Vérifiez que les fichiers obtenus de sources non fiables sont analysés par des scanners antivirus pour empêcher le téléchargement de contenus malveillants connus.	✓	✓	✓	509

V12.5 Exigences de téléchargement des fichiers

#	Description	L1	L2	L3	CWE
12.5.1	Vérifiez que l'application web est configuré pour ne servir que les fichiers ayant des extensions de fichier spécifiques afin d'éviter les informations involontaires et les fuites de code source. Par exemple, les fichiers de sauvegarde (par exemple .bak), les fichiers de travail temporaires (par exemple .swp), les fichiers compressés (.zip, .tar.gz, etc) et les autres extensions couramment utilisées par les éditeurs doivent être bloqués, sauf si cela est nécessaire.	✓	✓	✓	552
12.5.2	Vérifiez que les demandes directes aux fichiers téléchargés ne seront jamais exécutées en tant que contenu HTML/JavaScript.	✓	✓	✓	434

V12.6 Exigences de protection des SSRF

#	Description	L1	L2	L3	CWE
12.6.1	Vérifiez que le serveur web ou d'application est configuré avec une liste d'autorisation de ressources ou de systèmes à partir desquels le serveur peut envoyer des requêtes ou charger des données/fichiers.	✓	✓	✓	918

Références

Pour plus d'informations, voir aussi :

- File Extension Handling for Sensitive Information
- Reflective file download by Oren Hafif

- OWASP Third Party JavaScript Management Cheat Sheet

V13 : Exigences de vérification des API et des services Web

Objectif de contrôle

Veiller à ce qu'une application vérifiée qui utilise des API de service de confiance (utilisant généralement JSON ou XML ou GraphQL) tel :

- Une authentification, une gestion de session et une autorisation adéquates de tous les services web.
- Une validation d'entrée de tous les paramètres qui passent d'un niveau de confiance inférieur à un niveau supérieur.
- Des contrôles de sécurité efficaces pour tous les types d'API, y compris les API en nuage et les API sans serveur

Veuillez lire ce chapitre en combinaison avec tous les autres chapitres à ce même niveau ; nous ne dupliquons plus les problèmes d'authentification ou de gestion de session API.

V13.1 Exigences génériques de vérification de la sécurité des services web

#	Description	L1	L2	L3	CWE
13.1.1	Vérifiez que tous les composants de l'application utilisent les mêmes encodages et analyseurs pour éviter les attaques par analyse qui exploitent des comportements différents d'URI ou d'analyse de fichiers qui pourraient être utilisés dans les attaques SSRF et RFI.	✓	✓	✓	116
13.1.2	Vérifiez que l'accès aux fonctions d'administration et de gestion est limité aux administrateurs autorisés.	✓	✓	✓	419
13.1.3	Vérifier que les URL des API n'exposent pas d'informations sensibles, telles que la clé API, les jetons de session, etc.	✓	✓	✓	598
13.1.4	Vérifier que les décisions d'autorisation sont prises à la fois à l'URI, appliquées par la sécurité programmatique ou déclarative au niveau du contrôleur ou du routeur, et au niveau des ressources, appliquées par des autorisations basées sur des modèles de permission.		✓	✓	285
13.1.5	Vérifiez que les demandes contenant des types de contenu inattendus ou manquants sont rejetées avec les en-têtes appropriés (statut de réponse HTTP 406 Inacceptable ou 415 Type de support non pris en charge).		✓	✓	434

V13.2 Exigences de védification pour les services web de type RESTful

La validation du schéma JSON en est à un stade préliminaire de normalisation (voir références). Lorsque vous envisagez d'utiliser la validation de schéma JSON, qui est la meilleure pratique pour les services web RESTful, pensez à utiliser ces stratégies de validation de données supplémentaires en combinaison avec la validation de schéma JSON :

- Validation de l'objet JSON, par exemple s'il y a des éléments manquants ou en trop.
- Validation des valeurs de l'objet JSON en utilisant des méthodes de validation d'entrée standard, telles que le type de données, le format de données, la longueur, etc.
- et validation formelle du schéma JSON.

Une fois que la norme de validation du schéma JSON sera formalisée, l'ASVS mettra à jour ses conseils dans ce domaine. Surveillez attentivement toutes les bibliothèques de validation de schémas JSON utilisées, car elles devront être mises à jour régulièrement jusqu'à ce que la norme soit formalisée et que les bogues soient éliminés des implémentations de référence.

#	Description	L1	L2	L3	CWE
13.2.1	Vérifiez que les méthodes HTTP RESTful activées sont un choix valable pour l'utilisateur ou une action, comme par exemple empêcher les utilisateurs normaux d'utiliser le verbe DELETE ou PUT sur des API ou des ressources protégées.	✓	✓	✓	650
13.2.2	Vérifiez que les requêtes HTTP utilisant le verbe HEAD, OPTIONS, TRACE ou GET ne modifient aucune structure de données dorsale ni n'effectuent aucune action de changement d'état. Ces requêtes sont des méthodes sûres et ne devraient donc pas avoir d'effets secondaires.	✓	✓	✓	650
13.2.3	Vérifier que la validation du schéma JSON est en place et vérifiée avant d'accepter la saisie.	✓	✓	✓	20
13.2.4	Vérifiez que les services web RESTful qui utilisent des cookies sont protégés contre la falsification des requêtes intersites par l'utilisation d'au moins un ou plusieurs des éléments suivants : modèle de cookie à triple ou double soumission (voir références), nonces CSRF ou vérification de l'en-tête de la requête d'origine.	✓	✓	✓	352
13.2.5	Vérifiez que les services REST disposent de contrôles anti-automatisation pour se protéger contre les appels excessifs, surtout si l'API n'est pas authentifiée.		✓	✓	770
13.2.6	Vérifiez que les services REST vérifient explicitement que le type de contenu entrant est bien celui attendu, par exemple application/xml ou application/json.		✓	✓	436
13.2.7	Vérifiez que les en-têtes et la données utiles sont dignes de confiance et intègre. Exiger un cryptage fort pour le transport (TLS uniquement) peut être suffisant dans de nombreux cas, car il assure à la fois la protection de la confidentialité et de l'intégrité. Les messages signés peuvent fournir une assurance supplémentaire en plus des protections de transport pour les applications de haute sécurité, mais elles entraînent une complexité et des risques supplémentaires à comparer avec les avantages.		✓	✓	345

V13.3 Exigences de vérification du service web SOAP

#	Description	L1	L2	L3	CWE
13.3.1	Vérifier que la validation du schéma XSD a lieu pour garantir un document XML correctement formé, suivie de la validation de chaque champ de saisie avant tout traitement de ces données.	✓	✓	✓	20
13.3.2	Vérifier que la charge utile du message est signée en utilisant WS-Security pour assurer un transport fiable entre le client et le service.		✓	✓	345

Note : En raison de problèmes liés aux attaques XXE contre les DTD, la validation des DTD ne doit pas être utilisée, et l'évaluation des DTD cadre doit être désactivée conformément aux exigences définies dans la configuration V14.

V13.4 GraphQL et autres exigences de sécurité de la couche de données des services Web

#	Description	L1	L2	L3	CWE
13.4.1	Vérifiez qu'un méchanisme de limitation d'allocation de resource ou de limitation de complexité soit en place pour prévenir les dénis de services.		✓	✓	770
13.4.2	Vérifiez que la logique d'autorisation de GraphQL ou d'une autre couche de données doit être mise en œuvre au niveau de la couche de logique d'entreprise au lieu de la couche GraphQL.		✓	✓	285

Références

Pour plus d'informations, voir aussi :

- OWASP Serverless Top 10
- OWASP Serverless Project
- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing
- OWASP Cross-Site Request Forgery cheat sheet
- OWASP XML External Entity Prevention Cheat Sheet - General Guidance
- JSON Web Tokens (and Signing)
- REST Security Cheat Sheet
- JSON Schema
- XML DTD Entity Attacks
- Orange Tsai - A new era of SSRF Exploiting URL Parser In Trending Programming Languages

V14 : Exigences de vérification de la configuration

Objectif de contrôle

Assurez-vous qu'une application vérifiée satisfait :

- Un environnement de construction sécurisé, reproductible et automatisable.
- Une gestion des dépendances étroite et une configuration renforcée, de sorte que les composants obsolètes ou non sécurisés ne soient pas inclus dans l'application.
- Une configuration sécurisée par défaut, de sorte que les administrateurs et les utilisateurs doivent affaiblir la sécurité par défaut.

La configuration de l'application "out of the box" doit être sûre pour être sur Internet, ce qui signifie une configuration "out of the box".

V14.1 Exigences sur les constructions

Les pipelines de construction sont la base d'une sécurité reproductible : chaque fois qu'un élément non sécurisé est découvert, il peut être résolu dans le code source, les scripts de construction ou de déploiement, et testé automatiquement. Nous encourageons fortement l'utilisation de pipelines de compilation avec des contrôles de sécurité et de dépendance automatiques qui avertissent ou interrompent la compilation afin d'éviter que des problèmes de sécurité connus ne soient déployés en production. Les étapes manuelles effectuées de manière irrégulière conduisent directement à des erreurs de sécurité évitables.

Alors que l'industrie se dirige vers un modèle DevSecOps, il est important de garantir la disponibilité et l'intégrité continues du déploiement et de la configuration pour atteindre un état "known good". Dans le passé, si un système était piraté, il fallait des jours, voire des mois, pour prouver qu'aucune autre intrusion n'avait eu lieu. Aujourd'hui, avec l'avènement des infrastructures définies par logiciel, des déploiements A/B rapides sans aucun temps d'arrêt, et des constructions automatisées conteneurisées, il est possible de construire, de durcir et de déployer automatiquement et en continu un "known good" en remplacement de tout système compromis.

Si les modèles traditionnels sont toujours en place, des mesures manuelles doivent être prises pour renforcer et sauvegarder cette configuration afin de permettre le remplacement rapide des systèmes compromis par des systèmes à haute intégrité et sans compromis, et ce dans les meilleurs délais.

La conformité à cette section nécessite un système de construction automatisé et l'accès à des scripts de construction et de déploiement.

#	Description	L1	L2	L3	CWE
14.1.1	Vérifier que les processus de construction et de déploiement des applications sont effectués de manière sûre et répétable, comme l'automatisation des CI / CD, la gestion automatisée de la configuration et les scripts de déploiement automatisés.		✓	✓	
14.1.2	Vérifiez que les drapeaux du compilateur sont configurés pour activer toutes les protections et les avertissements disponibles contre les débordements de mémoire tampon, y compris la randomisation de la pile, la prévention de l'exécution des données, et pour casser la compilation si un pointeur, une mémoire, une chaîne de format, un entier ou une chaîne de caractères dangereux sont trouvés.		✓	✓	120
14.1.3	Vérifiez que la configuration du serveur est durcie conformément aux recommandations du serveur d'application et des cadres utilisés.		✓	✓	16

#	Description	L1	L2	L3	CWE
14.1.4	Vérifier que l'application, la configuration et toutes les dépendances peuvent être redéployées à l'aide de scripts de déploiement automatisés, construites à partir d'un runbook documenté et testé dans un délai raisonnable, ou restaurées à partir de sauvegardes en temps utile.		✓	✓	
14.1.5	Vérifier que les administrateurs autorisés peuvent vérifier l'intégrité de toutes les configurations pertinentes pour la sécurité afin de détecter les altérations.			✓	

V14.2 Exigences sur les dépendances

La gestion des dépendances est essentielle au bon fonctionnement de toute application, quel que soit son type. L'incapacité à se tenir à jour avec des dépendances obsolètes ou peu sûres est la cause première des attaques les plus importantes et les plus coûteuses à ce jour.

Remarque : au niveau 1, la conformité à la norme 14.2.1 concerne les observations ou les détections de bibliothèques et de composants côté client et autres, plutôt que l'analyse statique du code de construction ou l'analyse des dépendances, plus précise. Ces techniques plus précises pourraient être découvertes par des entretiens, le cas échéant.

#	Description	L1	L2	L3	CWE
14.2.1	Vérifiez que tous les composants sont à jour, de préférence en utilisant un vérificateur de dépendances pendant le temps de construction ou de compilation. (C2)	✓	✓	✓	1026
14.2.2	Vérifiez que toutes les fonctionnalités, la documentation, les échantillons et les configurations inutiles sont supprimés, tels que les exemples d'applications, la documentation de la plate-forme et les utilisateurs par défaut ou les exemples.	✓	✓	✓	1002
14.2.3	Vérifier que si les actifs d'application, tels que les bibliothèques JavaScript, les feuilles de style CSS ou les polices web, sont hébergés en externe sur un réseau de diffusion de contenu (CDN) ou un fournisseur externe, l'intégrité des sous-ressources (SRI) est utilisée pour valider l'intégrité de l'actif.	✓	✓	✓	829
14.2.4	Vérifier que les composants tiers proviennent de dépôts prédéfinis, fiables et continuellement entretenus. (C2)		✓	✓	829
14.2.5	Vérifier qu'un catalogue d'inventaire de toutes les bibliothèques tierces en service est tenu à jour. (C2)		✓	✓	
14.2.6	Vérifiez que la surface d'attaque est réduite en mettant en bac à sable ou en encapsulant des bibliothèques tierces pour n'exposer que le comportement requis dans l'application. (C2)		✓	✓	265

V14.3 Exigences de divulgation involontaire de renseignements sur la sécurité

Les configurations de production devraient être renforcées pour se protéger contre les attaques courantes, telles que les consoles de débogage, relever la barre pour les attaques de type "cross-site scripting" (XSS) et "remote file inclusion" (RFI), et pour éliminer les "vulnérabilités" triviales de découverte d'informations qui sont la marque indésirable de nombreux rapports de tests de pénétration. Nombre de ces problèmes sont rarement considérés comme un risque important, mais ils sont liés à d'autres vulnérabilités. Si ces problèmes ne sont pas présents par défaut, elle place la barre plus haute avant que la plupart des attaques puissent réussir.

#	Description	L1	L2	L3	CWE
14.3.1	Vérifiez que les messages d'erreur du serveur web ou d'application et du cadre sont configurés pour fournir des réponses personnalisées et exploitables par l'utilisateur afin d'éliminer toute divulgation involontaire de sécurité.	✓	✓	✓	209
14.3.2	Vérifier que les modes de débogage du serveur web ou d'application et du cadre d'application sont désactivés en production afin d'éliminer les fonctionnalités de débogage, les consoles de développement et les divulgations de sécurité non intentionnelles.	✓	✓	✓	497
14.3.3	Vérifiez que les en-têtes HTTP ou toute partie de la réponse HTTP n'exposent pas d'informations détaillées sur la version des composants du système.	✓	✓	✓	200

V14.4 Exigences relatives aux en-têtes de sécurité HTTP

#	Description	L1	L2	L3	CWE
14.4.1	Vérifiez que chaque réponse HTTP contient un en-tête Content-Type. Les types de contenu text/*, /+xml et application/xml doivent également spécifier un jeu de caractères sûr (par exemple, UTF-8, ISO-8859-1).	✓	✓	✓	173
14.4.2	Vérifiez que toutes les réponses de l'API contiennent Content-Disposition : attachment ; filename="api.json" (ou tout autre nom de fichier approprié pour le type de contenu).	✓	✓	✓	116
14.4.3	Vérifier qu'une politique de sécurité du contenu (CSP) est en place pour aider à atténuer l'impact des attaques XSS comme les vulnérabilités d'injection HTML, DOM, JSON et JavaScript.	✓	✓	✓	1021
14.4.4	Vérifiez que toutes les réponses contiennent X-Content-Type-Options: nosniff.	✓	✓	✓	116
14.4.5	Vérifiez que l'en-tête Strict-Transport-Security est inclus dans toutes les réponses et pour tous les sous-domaines, comme Strict-Transport-Security : max-age=15724800 ; includeSubdomains.	✓	✓	✓	523
14.4.6	Vérifiez qu'un en-tête "Referrer-Policy" approprié est inclus, tel que "no-referrer" ou "same-origin".	✓	✓	✓	116
14.4.7	Vérifier que le contenu d'une application web ne peut pas être intégré par défaut dans un site tiers et que l'intégration des ressources exactes n'est autorisée que si nécessaire en utilisant un en-tête approprié tel "Content-Security-Policy: frame-ancestors" ou "X-Frame-Options".	✓	✓	✓	346

V14.5 Exigences sur la validation des en-têtes de requête HTTP

#	Description	L1	L2	L3	CWE
14.5.1	Vérifiez que le serveur d'application accepte seulement les méthodes HTTP utilisées par l'application/API (incluant les requetes de type OPTIONS), et les journalise/alertes sur toutes les demandes qui sont invalades pour le contexte de l'application.	✓	✓	✓	749
14.5.2	Vérifiez que l'en-tête Origin fourni n'est pas utilisé pour les décisions d'authentification ou de contrôle d'accès, car l'en-tête Origin peut facilement être modifié par un attaquant.	✓	✓	✓	346
14.5.3	Vérifiez que l'en-tête "Cross-Origin Resource Sharing" (CORS) Access-Control-Allow-Origin utilise une liste d'autorisation stricte de domaines et sous-domaines de confiance pour la comparaison avec l'origine "null" et ne la prend pas en charge.	✓	✓	✓	346
14.5.4	Vérifiez que les en-têtes HTTP ajoutés par un proxy de confiance ou des dispositifs SSO, tels qu'un jeton au porteur, sont authentifiés par l'application.		✓	✓	306

Références

Pour plus d'informations, voir aussi :

- OWASP Testing Guide 4.0: Testing for HTTP Verb Tampering
- Adding Content-Disposition to API responses helps prevent many attacks based on misunderstanding on the MIME type between client and server, and the "filename" option specifically helps prevent Reflected File Download attacks.
- Content Security Policy Cheat Sheet
- Exploiting CORS misconfiguration for BitCoins and Bounties
- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing
- Sandboxing third party components

Annexe A : Glossaire

- **2FA** - L'authentification à deux facteurs(2FA) ajoute un deuxième niveau d'authentification à la connexion d'un compte.
- **Address Space Layout Randomization (ASLR)** - Une technique pour rendre plus difficile l'exploitation des bugs de corruption de la mémoire.
- **Sécurité des applications** - La sécurité au niveau des applications se concentre sur l'analyse des composants qui constituent la couche application du modèle de référence d'interconnexion des systèmes ouverts (modèle OSI), plutôt que de se concentrer par exemple sur le système d'exploitation sous-jacent ou les réseaux connectés.
- **Vérification de la sécurité des applications** - L'évaluation technique d'une application par rapport à l'ASVS de l'OWASP.
- **Rapport de vérification de la sécurité des applications** - Rapport qui documente les résultats globaux et l'analyse à l'appui produite par le vérificateur pour une application particulière.
- **Authentification** - La vérification de l'identité déclarée d'un utilisateur d'application.
- **Vérification automatisée** - Utilisation d'outils automatisés (soit des outils d'analyse dynamique, soit des outils d'analyse statique, soit les deux) qui utilisent les signatures de vulnérabilité pour trouver des problèmes.
- **Test de la boîte noire** - Méthode de test de logiciels qui consiste à examiner la fonctionnalité d'une application sans examiner ses structures ou son fonctionnement internes.
- **Composant** - unité de code autonome, avec des interfaces disque et réseau associées, qui communique avec d'autres composants.
- **Scripting intersite** (XSS) - Faille de sécurité que l'on trouve généralement dans les applications web et qui permet l'injection de scripts côté client dans le contenu.
- **Module cryptographique** - Matériel, logiciel et/ou microprogramme qui met en œuvre des algorithmes cryptographiques et/ou génère des clés cryptographiques.
- **CWE** - Common Weakness Enumeration (CWE) est une liste développée par la communauté des faiblesses communes de sécurité logicielle. Elle sert de langage commun, de mesure pour les outils de sécurité des logiciels, et de base pour l'identification des faiblesses, l'atténuation et les efforts de prévention.
- **DAST** - Les technologies de test dynamique de la sécurité des applications (DAST) sont conçues pour détecter les conditions indiquant une vulnérabilité de sécurité dans une application en cours d'exécution.
- **Vérification de la conception** - Évaluation technique de l'architecture de sécurité d'une application.
- **Vérification dynamique** - L'utilisation d'outils automatisés qui utilisent les signatures de vulnérabilité pour trouver des problèmes pendant l'exécution d'une application.
- **Globally Unique Identifier** (GUID) - Numéro de référence unique utilisé comme identifiant dans un logiciel.
- **Hyper Text Transfer Protocol** (HTTPS) - Un protocole d'application pour les systèmes d'information hypermédia distribués et collaboratifs. Il constitue la base de la communication de données pour le World Wide Web.
- **Clé codée en dur** - Clé cryptographique qui est stockée sur le système de fichiers, que ce soit dans le code, les commentaires ou les fichiers.
- **Validation des entrées** - La canonisation et la validation des entrées des utilisateurs non fiables.

- **Code malveillant** - Code introduit dans une application au cours de son développement à l'insu du propriétaire de l'application, qui contourne la politique de sécurité prévue pour l'application. Ce n'est pas la même chose qu'un logiciel malveillant tel qu'un virus ou un ver !
- **Malware** - Code exécutable qui est introduit dans une application pendant son exécution à l'insu de l'utilisateur ou de l'administrateur de l'application.
- **Open Web Application Security Project** (OWASP) - L'Open Web Application Security Project (OWASP) est une communauté mondiale libre et ouverte qui vise à améliorer la sécurité des logiciels d'application. Notre mission est de rendre la sécurité des applications "visible", afin que les personnes et les organisations puissent prendre des décisions éclairées sur les risques liés à la sécurité des applications. Voir : https://www.owasp.org/
- **Informations d'identification personnelle** (IIP) - sont des informations qui peuvent être utilisées seules ou avec d'autres informations pour identifier, contacter ou localiser une seule personne, ou pour identifier un individu dans son contexte.
- **PIE** - L'exécutable indépendant de la position (PIE) est un corps de code machine qui, placé quelque part dans la mémoire primaire, s'exécute correctement quelle que soit son adresse absolue.
- **PKI** - Public Key Infrastructure (PKI) est un arrangement qui lie les clés publiques avec les identités respectives des entités. La liaison est établie par un processus d'enregistrement et de délivrance de certificats auprès et par une autorité de certification (CA).
- **SAST** - Le test statique de sécurité des applications (SAST) est un ensemble de technologies conçues pour analyser le code source des applications, le code d'octet et les binaires pour le codage et les conditions de conception qui sont indicatives des vulnérabilités de sécurité. Les solutions SAST analysent une application de "l'intérieur vers l'extérieur" dans un état de non-exécution.
- **SDLC** - Cycle de vie du développement logiciel.
- **Security Architecture** - Une abstraction de la conception d'une application qui identifie et décrit où et comment les contrôles de sécurité sont utilisés, et identifie et décrit également l'emplacement et la sensibilité des données de l'utilisateur et de l'application.
- **Configuration de la sécurité** - La configuration d'exécution d'une application qui affecte la manière dont la sécurité est assurée.
- **Contrôle de sécurité** - Une fonction ou un composant qui effectue un contrôle de sécurité (par exemple, un contrôle d'accès) ou qui, lorsqu'il est appelé, produit un effet de sécurité (par exemple, en générant un enregistrement d'audit).
- **Injection SQL (SQLi)** - Technique d'injection de code utilisée pour attaquer des applications orientées données, dans laquelle des instructions SQL malveillantes sont insérées dans un point d'entrée.
- **SSO Authentification** - Le Single Sign On (SSO) se produit lorsqu'un utilisateur se connecte à une application et est ensuite automatiquement connecté à d'autres applications sans avoir à se ré-authentifier. Par exemple, lorsque vous vous connectez à Google, lorsque vous accédez à d'autres services Google tels que YouTube, Google Docs et Gmail, vous serez automatiquement connecté.
- **SVG** - Graphiques vectoriels évolutifs
- **Modélisation de la menace** - Technique consistant à développer des architectures de sécurité de plus en plus raffinées pour identifier les agents de menace, les zones de sécurité, les contrôles de sécurité et les actifs techniques et commerciaux importants.
- **Transport Layer Security** - Protocoles cryptographiques qui assurent la sécurité des communications sur une connexion réseau
- **URI/URL/URL fragments** - Un Uniform Resource Identifier est une chaîne de caractères utilisée pour identifier un nom ou une ressource web. Un Uniform Resource Locator est souvent utilisé comme référence à une ressource.

- **Vérificateur** - La personne ou l'équipe qui examine une demande par rapport aux exigences de l'OWASP ASVS.
- **liste d'autorisation** - Une liste de données ou d'opérations autorisées, par exemple une liste de caractères qui sont autorisés à effectuer une validation d'entrée.
- **Certificat X.509** - Un certificat X.509 est un certificat numérique qui utilise la norme internationale largement acceptée d'infrastructure à clé publique (PKI) X.509 pour vérifier qu'une clé publique appartient à l'utilisateur, à l'ordinateur ou à l'identité du service contenu dans le certificat.

Annexe B : Références

Les projets suivants de l'OWASP sont les plus susceptibles d'être utiles aux utilisateurs/adopteurs de cette norme :

Projets de base de l'OWASP

1. OWASP Top 10 Project: https://owasp.org/www-project-top-ten/
2. OWASP Testing Guide: https://owasp.org/www-project-web-security-testing-guide/
3. OWASP Proactive Controls: https://owasp.org/www-project-proactive-controls/
4. OWASP Security Knowledge Framework: https://owasp.org/www-project-security-knowledge-framework/
5. OWASP Software Assurance Maturity Model (SAMM): https://owasp.org/www-project-samm/

Projet OWASP Cheat Sheet Series

This project has a number of cheat sheets which will be relevant for different topics in the ASVS.

Vous trouverez ici une correspondance avec l'ASVS :
https://github.com/OWASP/CheatSheetSeries/blob/master/IndexASVS.md

Projets liés à la sécurité mobile

1. OWASP Mobile Security Project: https://owasp.org/www-project-mobile-security/
2. OWASP Mobile Top 10 Risks: https://owasp.org/www-project-mobile-top-10/
3. OWASP Mobile Security Testing Guide: https://owasp.org/www-project-mobile-security-testing-guide/

Projets liés à l'Internet des objets de l'OWASP

1. Projet OWASP "Internet of Things" :
https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

Projets OWASP sans serveur

1. OWASP Serverless Project: https://owasp.org/www-project-serverless-top-10/

Autres

De même, les sites web suivants sont les plus susceptibles d'être utiles aux utilisateurs/adopteurs de cette norme

1. SecListes Github : https://github.com/danielmiessler/SecLists
2. MITRE Dénombrement des faiblesses communes : https://cwe.mitre.org/
3. Conseil des normes de sécurité PCI : https://www.pcisecuritystandards.org
4. Norme de sécurité des données PCI (DSS) v3.2.1 Exigences et procédures d'évaluation de la sécurité :
https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf
5. PCI Software Security Framework - Exigences et procédures d'évaluation des logiciels sécurisés :
https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf
6. Exigences et procédures d'évaluation du PCI Secure Software Lifecycle (Secure SLC) :
https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf

Annexe C : Exigences de vérification de l'Internet des objets

Cette section était à l'origine dans la branche principale, mais avec le travail effectué par l'équipe IoT OWASP, il n'est pas logique de maintenir deux fils de discussion différents sur le sujet. Pour la version 4.0, nous la déplaçons vers l'annexe, et nous invitons tous ceux qui le souhaitent à utiliser plutôt la branche principale OWASP IoT project

Objectif de contrôle

Les dispositifs embarqués/implantés devraient :

- Avoir le même niveau de contrôle de sécurité dans le dispositif que dans le serveur, en appliquant les contrôles de sécurité dans un environnement de confiance.
- Les données sensibles stockées sur l'appareil doivent l'être de manière sécurisée en utilisant un stockage sauvegardé par du matériel tel que des éléments sécurisés.
- Toutes les données sensibles transmises par le dispositif doivent utiliser la couche de transport de sécurité.

Exigences de vérification de la sécurité

#	Description	L1	L2	L3	Depuis
C.1	Vérifiez que les interfaces de débogage de la couche application telles que les interfaces USB, UART et autres variantes série sont désactivées ou protégées par un mot de passe complexe.	✓	✓	✓	4.0
C.2	Vérifier que les clés et certificats cryptographiques sont uniques à chaque appareil.	✓	✓	✓	4.0
C.3	Vérifiez que les contrôles de protection de la mémoire tels que ASLR et DEP sont activés par le système d'exploitation embarqué/OT, le cas échéant.	✓	✓	✓	4.0
C.4	Vérifiez que les interfaces de débogage sur puce telles que JTAG ou SWD sont désactivées ou que le mécanisme de protection disponible est activé et configuré de manière appropriée.	✓	✓	✓	4.0
C.5	Vérifiez que l'exécution de confiance est mise en œuvre et activée, si elle est disponible sur le SoC ou le CPU de l'appareil.	✓	✓	✓	4.0
C.6	Vérifier que les données sensibles, les clés privées et les certificats sont stockés de manière sécurisée dans un élément sécurisé, TPM, TEE (Trusted Execution Environment), ou protégés par une cryptographie forte.	✓	✓	✓	4.0
C.7	Vérifiez que les applications du microprogramme protègent les données en transit en utilisant la sécurité de la couche transport.	✓	✓	✓	4.0
C.8	Vérifiez que les applications du microprogramme valident la signature numérique des connexions au serveur.	✓	✓	✓	4.0
C.9	Vérifiez que les communications sans fil sont mutuellement authentifiées.	✓	✓	✓	4.0
C.10	Vérifiez que les communications sans fil sont envoyées sur un canal crypté.	✓	✓	✓	4.0
C.11	Vérifier que toute utilisation de fonctions C interdites est remplacée par les fonctions équivalentes sûres appropriées.	✓	✓	✓	4.0
C.12	Vérifiez que chaque microprogramme tient à jour une nomenclature des logiciels cataloguant les composants tiers, le versionnage et les vulnérabilités publiées.	✓	✓	✓	4.0

#	Description	L1	L2	L3	Depuis
C.13	Vérifier que tous les codes, y compris les binaires de tiers, les bibliothèques, les cadres sont examinés pour les références codées en dur (backdoors).	✓	✓	✓	4.0
C.14	Vérifiez que les composants de l'application et du micrologiciel ne sont pas susceptibles de recevoir l'OS Command Injection en invoquant des enveloppes de commandes shell, des scripts, ou que les contrôles de sécurité empêchent l'OS Command Injection.	✓	✓	✓	4.0
C.15	Vérifiez que les applications du microprogramme fixent la signature numérique à un ou plusieurs serveurs de confiance.		✓	✓	4.0
C.16	Vérifier la présence de dispositifs de résistance et/ou de détection de l'altération.		✓	✓	4.0
C.17	Vérifiez que toutes les technologies de protection de la propriété intellectuelle disponibles fournies par le fabricant de la puce sont activées.		✓	✓	4.0
C.18	Vérifiez que des contrôles de sécurité sont en place pour empêcher l'ingénierie inverse des microprogrammes (par exemple, suppression des symboles de débogage verbeux).		✓	✓	4.0
C.19	Vérifiez que le périphérique valide la signature de l'image de démarrage avant le chargement.		✓	✓	4.0
C.20	Vérifiez que le processus de mise à jour du microprogramme n'est pas vulnérable aux attaques par heure de contrôle ou par heure d'utilisation.		✓	✓	4.0
C.21	Vérifiez que l'appareil utilise la signature de code et valide les fichiers de mise à jour du micrologiciel avant l'installation.		✓	✓	4.0
C.22	Vérifiez que l'appareil ne peut pas être rétrogradé vers d'anciennes versions (anti-rollback) de firmware valide.		✓	✓	4.0
C.23	Vérifier l'utilisation d'un générateur de nombres pseudo-aléatoires cryptographiquement sécurisé sur un dispositif intégré (par exemple, en utilisant des générateurs de nombres aléatoires fournis par des puces).		✓	✓	4.0
C.24	Vérifiez que le microprogramme peut effectuer des mises à jour automatiques selon un calendrier prédéfini.		✓	✓	4.0
C.25	Vérifiez que l'appareil efface le micrologiciel et les données sensibles dès la détection d'une altération ou la réception d'un message non valide.			✓	4.0
C.26	Vérifiez que seuls les microcontrôleurs qui prennent en charge la désactivation des interfaces de débogage (par exemple JTAG, SWD) sont utilisés.			✓	4.0
C.27	Vérifiez que seuls les microcontrôleurs qui offrent une protection substantielle contre le décapuchonnage et les attaques des canaux latéraux sont utilisés.			✓	4.0
C.28	Vérifier que les traces sensibles ne sont pas exposées aux couches extérieures du circuit imprimé.			✓	4.0
C.29	Vérifiez que la communication entre les puces est cryptée (par exemple, communication de carte principale à carte fille).			✓	4.0

#	Description	L1	L2	L3	Depuis
C.30	Vérifiez que l'appareil utilise la signature de code et valide le code avant l'exécution.			✓	4.0
C.31	Vérifiez que les informations sensibles conservées en mémoire sont écrasées par des zéros dès qu'elles ne sont plus nécessaires.			✓	4.0
C.32	Vérifiez que les applications de microprogrammes utilisent des conteneurs de noyau pour l'isolation entre les applications.			✓	4.0
C.33	Vérifiez que les drapeaux de compilateurs sécurisés tels que -fPIE, -fstack-protector-all, -Wl,-z,noexecstack, -Wl,-z,noexecheap sont configurés pour les constructions de microprogrammes.			✓	4.0
C.34	Vérifiez que les microcontrôleurs sont configurés avec une protection par code (le cas échéant).			✓	4.0

Références

Pour plus d'informations, voir aussi :

- OWASP Internet of Things Top 10
- OWASP Embedded Application Security Project
- OWASP Internet of Things Project
- Trudy TCP Proxy Tool