Référentiel de vérification de la sécurité des applications (ASVS)

Version 5.0.0

Mai 2025

Table of Contents

Frontispice

À propos du référentiel

Le Référentiel de vérification de la sécurité des applications (Application Security Verification Standard) est une liste d'exigences de sécurité des applications que les architectes, les développeurs, les testeurs, les professionnels de la sécurité, les fournisseurs d'outils et les consommateurs peuvent utiliser pour définir, créer, tester et vérifier des applications sécurisées.

Copyright et licence

Version 5.0.0, Mai 2025

Chefs de projet

Jim Manico	Daniel Cuthbert
Josh C Grossman	Elar Lang

Groupes de travail

Tobias Ahnoff	Ralph Andalis	Ryan Armstrong	Gabriel Corona
Meghan Jacquot	Shanni Prutchi	Iman Sharafaldin	Eden Yardeni

Autres contributeurs majeurs

Sjoerd Langkemper	Isaac Lewis
Mark Carney	Sandro Gauci

Contributeurs et relecteurs

Nous avons inclus une liste des autres contributeurs dans l'annexe E.

S'il manque un crédit dans la liste des crédits 5.0, veuillez enregistrer un ticket sur GitHub pour être reconnu dans les futures mises à jour.

Le référentiel de vérification de la sécurité des applications repose sur les épaules des personnes concernées, de ASVS 1.0 en 2008 à 4.0 en 2019. 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 Andrew van der Stock, Mike Boberski, Jeff Williams et/ou 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.

Préface

Bienvenue dans la version 5.0 du référentiel de vérification de la sécurité des applications (ASVS).

Introduction

Lancé à l'origine en 2008 grâce à une collaboration communautaire mondiale, l'ASVS définit un ensemble exhaustif d'exigences de sécurité pour la conception, le développement et le test d'applications et de services Web modernes.

Après la sortie d'ASVS 4.0 en 2019 et sa mise à jour mineure (v4.0.3) en 2021, la version 5.0 représente une étape importante, modernisée pour refléter les dernières avancées en matière de sécurité logicielle.

ASVS 5.0 est le résultat de nombreuses contributions des chefs de projet, des membres du groupe de travail et de la communauté OWASP au sens large pour mettre à jour et améliorer ce référentiel important.

Principes derrière la version 5.0

Cette révision majeure a été élaborée en gardant à l'esprit plusieurs principes clés :

- Portée et orientation affinées : Cette version du référentiel a été conçue pour s'aligner plus directement sur les piliers fondamentaux de son nom : Application, Sécurité, Vérification et Référentiel. Les exigences ont été réécrites afin de mettre l'accent sur la prévention des failles de sécurité plutôt que d'imposer des implémentations techniques spécifiques. Les textes des exigences sont explicites et expliquent leur raison d'être.
- Prise en charge des décisions de sécurité documentées : ASVS 5.0 introduit des exigences de documentation des décisions de sécurité clés. Cela améliore la traçabilité et prend en charge les implémentations qui dépendent du contexte, permettant aux organisations d'adapter leur posture de sécurité à leurs besoins et risques spécifiques.
- Niveaux mis à jour : Bien qu'ASVS conserve son modèle à trois niveaux, les définitions des niveaux ont évolué pour faciliter son adoption. Le niveau 1 est conçu comme la première étape de l'adoption d'ASVS, fournissant la première couche de défense. Le niveau 2 offre une vue d'ensemble des pratiques de sécurité standard, et le niveau 3 répond aux exigences avancées de haute assurance.
- Contenu restructuré et enrichi : ASVS 5.0 comprend environ 350 exigences réparties en 17 chapitres. Les chapitres ont été réorganisés pour plus de clarté et de praticité. Un mappage bidirectionnel entre les versions 4.0 et 5.0 est fourni pour faciliter la migration.

Regard vers l'avenir

Tout comme la sécurisation d'une application n'est jamais vraiment achevée, l'ASVS ne l'est pas non plus. Bien que la version 5.0 soit une version majeure, le développement se poursuit. Cette version permet à la communauté de bénéficier des améliorations et des ajouts accumulés, et pose également les bases de futures améliorations. Cela pourrait inclure des efforts communautaires pour créer des guides de mise en œuvre et de vérification, basés sur les exigences de base.

ASVS 5.0 est conçu pour servir de base fiable au développement de logiciels sécurisés. La communauté est invitée à adopter, contribuer et développer ce référentiel afin de faire progresser collectivement la sécurité des applications.

Qu'est-ce que l'ASVS ?

Le Référentiel de vérification de la sécurité des applications (ASVS) définit des exigences de sécurité pour les applications et services web. Elle constitue une ressource précieuse pour quiconque souhaite concevoir, développer et maintenir des applications sécurisées ou évaluer leur sécurité.

Ce chapitre décrit les aspects essentiels de l'utilisation de l'ASVS, notamment son champ d'application, la structure de ses niveaux de priorité et ses principaux cas d'utilisation.

Portée de l'ASVS

La portée de l'ASVS est définie par son nom : Application, Sécurité, Vérification et Référentiel. Elle établit les exigences incluses ou exclues, avec pour objectif principal d'identifier les principes de sécurité à respecter. Elle prend également en compte les exigences de documentation, qui servent de base aux exigences de mise en œuvre.

Il n'existe aucune portée pour les attaquants. Par conséquent, les exigences de l'ASVS doivent être évaluées parallèlement aux recommandations relatives aux autres aspects du cycle de vie des applications, notamment les processus CI/CD, l'hébergement et les activités opérationnelles.

Application

ASVS définit une « application » comme le produit logiciel en cours de développement, dans lequel des contrôles de sécurité doivent être intégrés. ASVS ne prescrit pas les activités du cycle de développement ni la manière dont l'application doit être construite via un pipeline CI/CD ; il spécifie plutôt les résultats de sécurité à atteindre au sein même du produit.

Les composants qui traitent, modifient ou valident le trafic HTTP, tels que les pares-feux d'applications web (WAF), les équilibreurs de charge ou les proxys, peuvent être considérés comme faisant partie de l'application à ces fins spécifiques, car certains contrôles de sécurité en dépendent directement ou peuvent être implémentés par leur intermédiaire. Ces composants doivent être pris en compte pour les exigences liées aux réponses mises en cache, à la limitation du débit ou à la restriction des connexions entrantes et sortantes en fonction de la source et de la destination.

À l'inverse, ASVS exclut généralement les exigences qui ne concernent pas directement l'application ou dont la configuration ne relève pas de sa responsabilité. Par exemple, les problèmes DNS sont généralement gérés par une équipe ou une fonction distincte.

De même, si l'application est responsable de la manière dont elle consomme les entrées et produit les sorties, l'interaction d'un processus externe avec l'application ou ses données est considérée comme hors du champ d'application d'ASVS. Par exemple, la sauvegarde de l'application ou de ses données relève généralement de la responsabilité d'un processus externe et n'est pas contrôlée par l'application ou ses développeurs.

Sécurité

Chaque exigence doit avoir un impact démontrable sur la sécurité. L'absence d'exigence doit entraîner une application moins sécurisée, et sa mise en œuvre doit réduire la probabilité ou l'impact d'un risque de sécurité.

Toutes les autres considérations, telles que les aspects fonctionnels, le style de code ou les exigences de politique, sont hors du champ d'application.

Vérification

L'exigence doit être vérifiable et la vérification doit aboutir à une décision d'échec ou de réussite.

Référentiel

L'ASVS est conçue comme un recueil d'exigences de sécurité à mettre en œuvre pour se conformer au référentiel. Cela signifie que les exigences se limitent à la définition de l'objectif de sécurité à atteindre. D'autres informations connexes peuvent être intégrées à l'ASVS ou reliées par des mappings.

Plus précisément, l'OWASP comporte de nombreux projets, et l'ASVS évite délibérément tout chevauchement avec le contenu d'autres projets. Par exemple, les développeurs peuvent se demander : « Comment implémenter une exigence particulière dans ma technologie ou mon environnement ? » Cette question devrait être traitée par le projet « Cheat-Sheets ». Les vérificateurs peuvent se demander : « Comment tester cette exigence dans cet environnement ? » Cette question devrait être traitée par le projet « Web Security Testing Guide ».

Bien que l'ASVS ne soit pas uniquement destiné aux experts en sécurité, il attend du lecteur des connaissances techniques pour comprendre le contenu ou la capacité à rechercher des concepts spécifiques.

Exigence

Le terme « exigence » est utilisé spécifiquement dans l'ASVS car il décrit ce qui doit être accompli pour la satisfaire. L'ASVS ne contient que des exigences (doit) et ne contient pas de recommandations (devrait) comme condition principale.

En d'autres termes, les recommandations, qu'elles constituent une solution parmi d'autres ou des considérations de style de code, ne répondent pas à la définition d'une exigence.

Les exigences de l'ASVS visent à répondre à des principes de sécurité spécifiques sans être trop spécifiques à une implémentation ou à une technologie, tout en étant explicites quant à leur raison d'être. Cela signifie également que les exigences ne sont pas construites autour d'une méthode de vérification ou d'une implémentation particulière.

Décisions de sécurité documentées

En matière de sécurité logicielle, planifier la conception de la sécurité et les mécanismes à utiliser en amont permettra une implémentation plus cohérente et fiable du produit fini ou de la fonctionnalité.

De plus, pour certaines exigences, la mise en œuvre sera complexe et très spécifique aux besoins de l'application. Parmi les exemples courants, on peut citer les autorisations, la validation des entrées et les contrôles de protection autour de différents niveaux de données sensibles.

Pour tenir compte de cela, plutôt que de formuler des affirmations générales telles que « toutes les données doivent être chiffrées » ou de tenter de couvrir tous les cas d'utilisation possibles dans une exigence, des exigences de documentation ont été incluses, exigeant que l'approche et la configuration du développeur d'applications pour ces types de contrôles soient documentées. Ceci peut ensuite être vérifié pour en vérifier la pertinence, puis la mise en œuvre réelle peut être comparée à la documentation afin d'évaluer si elle répond aux attentes.

Ces exigences visent à documenter les décisions prises par l'organisation développant l'application concernant la mise en œuvre de certaines exigences de sécurité.

Les exigences de documentation figurent toujours dans la première section d'un chapitre (bien que tous les chapitres n'en disposent pas) et sont toujours associées à une exigence de mise en œuvre, où les décisions documentées doivent être effectivement mises en œuvre. L'essentiel est de distinguer la vérification de la documentation et de la mise en œuvre effective.

L'inclusion de ces exigences repose sur deux facteurs clés. Le premier est qu'une exigence de sécurité implique souvent l'application de règles, par exemple concernant les types de fichiers autorisés à être téléchargés, les contrôles métier à appliquer et les caractères autorisés pour un champ particulier. Ces règles diffèrent d'une application à l'autre ; par conséquent, l'ASVS ne peut pas les définir de manière prescriptive, et un aide-mémoire ou une réponse plus détaillée ne serait d'aucune utilité dans ce cas précis. De même, sans documentation de ces décisions, il sera impossible de vérifier les exigences qui les mettent en œuvre.

Le deuxième facteur est que, pour certaines exigences, il est important d'offrir au développement d'applications une certaine flexibilité quant à la manière de répondre à des défis de sécurité particuliers. Par exemple, dans les versions précédentes d'ASVS, les règles d'expiration de session étaient très strictes. En pratique, de nombreuses applications, notamment celles destinées aux utilisateurs, ont des règles beaucoup plus souples et préfèrent mettre en œuvre d'autres contrôles d'atténuation. Les exigences de documentation permettent donc explicitement cette flexibilité.

Il est clair que les développeurs ne sont pas censés prendre et documenter ces décisions individuellement, mais que l'organisation dans son ensemble les prend et s'assure qu'elles sont communiquées aux développeurs, qui s'assurent ensuite de les respecter.

Fournir aux développeurs des spécifications et des conceptions pour de nouvelles fonctionnalités est une étape standard du développement logiciel. De même, les développeurs sont censés utiliser des composants et des mécanismes d'interface utilisateur communs plutôt que de prendre leurs propres décisions à chaque fois. Par conséquent, étendre ce principe à la sécurité ne devrait pas être perçu comme surprenant ou controversé.

La manière d'y parvenir est également flexible. Les décisions de sécurité peuvent être documentées dans un document littéral, auquel les développeurs sont tenus de se référer. Alternativement, elles peuvent être documentées et implémentées dans une bibliothèque de code commune que tous les développeurs sont tenus d'utiliser. Dans les deux cas, le résultat souhaité est atteint.

Niveaux de vérification de la sécurité des applications

L'ASVS définit trois niveaux de vérification de la sécurité, chaque niveau augmentant en profondeur et en complexité. L'objectif général est que les organisations commencent par le premier niveau pour répondre aux préoccupations de sécurité les plus critiques, puis progressent vers les niveaux supérieurs en fonction des besoins de l'organisation et des applications. Les niveaux peuvent être présentés comme L1, L2 et L3 dans le document et dans les textes d'exigences.

Chaque niveau ASVS indique les exigences de sécurité à atteindre à partir de ce niveau, les exigences des niveaux supérieurs restants étant des recommandations.

Afin d'éviter les doublons ou les exigences obsolètes aux niveaux supérieurs, certaines exigences s'appliquent à un niveau particulier, mais sont soumises à des conditions plus strictes pour les niveaux supérieurs.

Évaluation des niveaux

Les niveaux sont définis par une évaluation de priorité de chaque exigence, basée sur l'expérience de mise en œuvre et de test des exigences de sécurité. L'accent est mis principalement sur la balance entre la réduction des risques et les efforts nécessaires à la mise en œuvre de l'exigence. Un autre facteur clé est de maintenir une faible barrière à l'entrée.

La réduction des risques évalue dans quelle mesure l'exigence réduit le niveau de risque de sécurité au sein de l'application, en tenant compte des facteurs d'impact classiques de confidentialité, d'intégrité et de disponibilité, et en déterminant s'il s'agit d'une couche de défense principale ou d'une défense en profondeur.

Les discussions rigoureuses autour des critères et des décisions de nivellement ont abouti à une répartition qui devrait s'appliquer à la grande majorité des cas, tout en admettant qu'elle ne soit pas parfaitement adaptée à toutes les situations. Cela signifie que, dans certains cas, les organisations peuvent souhaiter prioriser les exigences d'un niveau supérieur en amont, en fonction de leurs propres considérations de risque.

Les types d'exigences à chaque niveau peuvent être caractérisés comme suit.

Niveau 1

Ce niveau contient les exigences minimales à prendre en compte pour sécuriser une application et constitue un point de départ essentiel. Il comprend environ 20 % des exigences ASVS. L'objectif est de réduire au minimum les exigences afin de réduire les obstacles à l'entrée.

Ces exigences sont généralement critiques ou basiques, et constituent la première couche de défense pour prévenir les attaques courantes. Elles ne nécessitent pas d'autres vulnérabilités ou conditions préalables pour être exploitables.

Outre les exigences de la première couche de défense, certaines exigences ont moins d'impact aux niveaux supérieurs, comme celles relatives aux mots de passe. Celles-ci sont plus importantes pour le Niveau 1, car à partir de ces niveaux, les exigences d'authentification multi facteur deviennent pertinentes.

Le Niveau 1 n'est pas nécessairement testable par un testeur externe sans accès interne à la documentation ou au code (comme les tests « boîte noire »), bien que le nombre réduit d'exigences devrait faciliter sa vérification.

Niveau 2

La plupart des applications devraient s'efforcer d'atteindre ce niveau de sécurité. Environ 50 % des exigences de l'ASVS sont de niveau 2, ce qui signifie qu'une application doit implémenter environ 70 % des exigences de l'ASVS (l'ensemble des exigences de niveau 1 et 2) pour être conforme à ce niveau.

Ces exigences concernent généralement des attaques moins courantes ou des protections plus complexes contre les attaques courantes. Elles peuvent constituer une première couche de défense ou nécessiter certaines conditions préalables pour que l'attaque réussisse.

Niveau 3

Ce niveau doit être l'objectif des applications souhaitant démontrer les plus hauts niveaux de sécurité et représente environ 30 % des exigences à respecter.

Les exigences de cette section concernent généralement des mécanismes de défense en profondeur ou d'autres contrôles utiles, mais difficiles à mettre en œuvre.

Quel niveau atteindre ?

Les niveaux de priorité visent à refléter la maturité de l'organisation et de l'application en matière de sécurité applicative. Plutôt que d'imposer un niveau de sécurité normatif pour une application, l'organisation doit analyser ses risques et déterminer le niveau qu'elle estime approprié, en fonction de la sensibilité de l'application et, bien sûr, des attentes de ses utilisateurs.

Par exemple, une jeune start-up qui ne collecte que des données sensibles limitées peut décider de se concentrer sur le niveau 1 pour ses objectifs de sécurité initiaux, tandis qu'une banque peut avoir du mal à justifier auprès de ses clients un niveau inférieur au niveau 3 pour son application de banque en ligne.

Comment utiliser l'ASVS ?

Structure de l'ASVS

L'ASVS comprend environ 350 exigences, réparties en 17 chapitres, chacun étant lui-même subdivisé en sections.

L'objectif de cette division est de simplifier la sélection ou le filtrage des chapitres et sections en fonction de leur pertinence pour l'application. Par exemple, pour une API machine-to-machine, les exigences du chapitre V3 relatives aux interfaces web ne seront pas pertinentes. Si OAuth ou WebRTC n'est pas utilisé, ces chapitres peuvent également être ignorés.

Stratégie de publication

Les publications ASVS suivent le modèle « Majeure.Mineure.Correctif » et les numéros indiquent les modifications apportées. Pour une publication majeure, le premier numéro change, pour une publication mineure, le deuxième, et pour une publication corrective, le troisième.

- Publication majeure : Réorganisation complète, presque tout peut avoir changé, y compris les numéros d'exigences. Une réévaluation de conformité sera nécessaire (par exemple, 4.0.3 -> 5.0.0).
- Publication mineure : Des exigences peuvent être ajoutées ou supprimées, mais la numérotation globale reste inchangée. Une réévaluation de conformité sera nécessaire, mais devrait être plus simple (par exemple, 5.0.0 -> 5.1.0).
- Publication corrective : Des exigences peuvent être supprimées (par exemple, si elles sont dupliquées ou obsolètes) ou assouplies, mais une application conforme à la publication précédente sera également conforme à la publication corrective (par exemple, 5.0.0 -> 5.0.1).

Les informations ci-dessus concernent spécifiquement les exigences de l'ASVS. Les modifications apportées au texte environnant et à d'autres contenus, tels que les annexes, ne seront pas considérées comme des modifications majeures.

Flexibilité avec l'ASVS

Plusieurs points décrits ci-dessus, tels que les exigences de documentation et le mécanisme de niveaux, permettent une utilisation de l'ASVS plus flexible et plus spécifique à l'organisation.

De plus, il est fortement encouragé aux organisations de créer un fork spécifique à l'organisation ou au domaine, afin d'ajuster les exigences en fonction des caractéristiques et des niveaux de risque de leurs applications. Cependant, il est important de maintenir une traçabilité afin que la conformité à l'exigence 4.1.1 soit identique pour toutes les versions.

Idéalement, chaque organisation devrait créer son propre ASVS personnalisé, en supprimant les sections non pertinentes (par exemple, GraphQL, WebSockets, SOAP, si elles ne sont pas utilisées). Une version ou un supplément ASVS spécifique à l'organisation est également un bon support pour fournir des conseils d'implémentation spécifiques à l'organisation, détaillant les bibliothèques ou les ressources à utiliser pour se conformer aux exigences.

Comment référencer les exigences ASVS ?

Chaque exigence possède un identifiant au format <CHAPITRE>.<SECTION>.<EXIGENCE>, où chaque élément est un numéro. Par exemple, 1.11.3.

- La valeur <CHAPITRE> correspond au chapitre d'où provient l'exigence ; par exemple, toutes les exigences 1.#.# proviennent du chapitre ‘Encodage et nettoyage’.
- La valeur <SECTION> correspond à la section de ce chapitre où apparaît l'exigence ; par exemple, toutes les exigences 1.2.# se trouvent dans la section ‘Prévention des injections’ du chapitre ‘Encodage et nettoyage’.
- La valeur <EXIGENCE> identifie l'exigence spécifique au sein du chapitre et de la section, par exemple, 1.2.5 qui, dans la version 5.0.0 de cette norme, est :

> 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 OS paramétrées ou un encodage contextuel de la sortie de ligne de commande.

Les identifiants pouvant varier d'une version à l'autre du référentiel, il est préférable, pour les autres documents, rapports ou outils, d'utiliser le format suivant : v<VERSION>-<CHAPITRE>.<SECTION>.<EXIGENCE> où « VERSION » correspond à la balise de version ASVS. Par exemple, « v5.0.0-1.2.5 » désigne spécifiquement la 5ème exigence de la section ‘Prévention des injections’ du chapitre ‘Encodage et nettoyage’ de la version 5.0.0. (Ceci pourrait être résumé par V<VERSION>-<IDENTIFIANT_EXIGENCE>.)

Remarque : Le V précédant le numéro de version dans le format doit toujours être en minuscule.

Si des identifiants sont utilisés sans inclure l'élément v<VERSION>, ils doivent être considérés comme faisant référence au contenu le plus récent du Référentiel de vérification de la sécurité des applications. À mesure que le référentiel évolue, cela devient problématique ; c'est pourquoi les rédacteurs et les développeurs doivent inclure l'élément « version ».

Les listes d'exigences de l'ASVS sont disponibles aux formats CSV, JSON et autres, utiles à des fins de référence ou d'utilisation programmatique.

Forker l'ASVS

Les organisations peuvent tirer profit de l'adoption d'ASVS en choisissant l'un des trois niveaux ou en créant un fork spécifique au domaine qui ajuste les exigences en fonction du niveau de risque applicatif. Ce type de fork est encouragé, à condition de garantir la traçabilité de l'exigence 4.1.1, garantissant ainsi la même cohérence pour toutes les versions.

Idéalement, chaque organisation devrait créer son propre ASVS personnalisé, en supprimant les sections non pertinentes (par exemple, GraphQL, Websockets, SOAP, si elles ne sont pas utilisées). Le fork doit commencer par le niveau 1 d'ASVS comme référence, puis progresser vers les niveaux 2 ou 3 en fonction du risque de l'application.

Cas d'utilisation de l'ASVS

L'ASVS peut être utilisé pour évaluer la sécurité d'une application, ce point étant approfondi dans le chapitre suivant. Cependant, plusieurs autres utilisations potentielles de l'ASVS (ou d'une version dérivée) ont été identifiées.

En tant que conseil détaillé sur l'architecture de sécurité

L'une des utilisations les plus courantes du Référentiel de vérification de la sécurité des applications (ASVS) est de servir de ressource aux architectes de sécurité. Les ressources disponibles pour construire une architecture applicative sécurisée sont limitées, notamment pour les applications modernes. L'ASVS peut combler ces lacunes en permettant aux architectes de sécurité de choisir de meilleurs contrôles pour les problèmes courants, tels que les modèles de protection des données et les stratégies de validation des entrées. Les exigences en matière d'architecture et de documentation seront particulièrement utiles à cet égard.

Référence spécialisée en codage sécurisé

L'ASVS peut servir de base à la préparation d'une référence de développement sécurisé lors du développement d'applications, aidant ainsi les développeurs à prendre en compte la sécurité lors de la création de logiciels. Bien que l'ASVS puisse servir de base, les organisations doivent élaborer leurs propres directives claires et unifiées, idéalement basées sur les recommandations des ingénieurs ou architectes de sécurité. De plus, les organisations sont encouragées, dans la mesure du possible, à préparer des mécanismes et bibliothèques de sécurité approuvés, référencés dans les directives et utilisables par les développeurs.

Guide pour les tests unitaires et d'intégration automatisés

L'ASVS est conçu pour être hautement testable. Certaines vérifications seront techniques, tandis que d'autres exigences (telles que les exigences d'architecture et de documentation) pourront nécessiter une revue de documentation ou d'architecture. En créant des tests unitaires et d'intégration qui testent et analysent par fuzzing des cas d'abus spécifiques et pertinents liés aux exigences, vérifiables par des moyens techniques, il devrait être plus facile de vérifier le bon fonctionnement de ces contrôles à chaque build. Par exemple, des tests supplémentaires peuvent être créés pour la suite de tests d'un contrôleur de connexion afin de tester la présence de valeur par défaut courante pour le paramètre username, l'énumération des comptes, le force brute, l'injection LDAP et SQL, et les attaques XSS. De même, un test sur le paramètre password doit inclure les mots

de passe courants, la longueur des mots de passe, l'injection d'octets nuls, la suppression du paramètre, les attaques XSS, etc.

Pour la formation au développement sécurisé

L'ASVS peut également servir à définir les caractéristiques d'un logiciel sécurisé. De nombreux cours de « codage sécurisé » se résument à des cours de piratage éthique, avec quelques conseils de codage. Cela n'aide pas forcément les développeurs à écrire du code plus sécurisé. Les cours de développement sécurisé peuvent plutôt utiliser l'ASVS en mettant l'accent sur ses mécanismes positifs, plutôt que sur les 10 principes négatifs à éviter. La structure de l'ASVS offre également une structure logique pour aborder les différents sujets liés à la sécurisation d'une application.

En tant que cadre pour l'approvisionnement des logiciels sécurisés

L'ASVS est un cadre idéal pour faciliter l'approvisionnement des logiciels sécurisés ou l'acquisition de services de développement de développement sur mesure. 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 à ce niveau.

Appliquer l'ASVS en pratique

Les menaces varient selon les motivations. Certains secteurs disposent de ressources informatiques et technologiques spécifiques et d'exigences de conformité réglementaire spécifiques à leur domaine.

Il est fortement recommandé aux organisations d'analyser en profondeur les caractéristiques de risque spécifiques à leur activité et de déterminer le niveau d'ASVS approprié en fonction de ces risques et des exigences métier.

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.

Comment vérifier la conformité ASVS

L'ASVS n'est volontairement pas prescriptif quant à la manière précise de vérifier la conformité au niveau d'un guide de tests. Il est toutefois important de souligner certains points clés.

Rapport de vérification

Les tests d'intrusion traditionnels signalent les problèmes « par exception », en ne listant que les échecs. Cependant, un rapport de certification ASVS doit inclure le périmètre, un résumé de toutes les exigences vérifiées, les exigences pour lesquelles des exceptions ont été constatées et des conseils pour résoudre les problèmes. Certaines exigences peuvent ne pas être applicables (par exemple, la gestion des sessions dans les API sans état), et cela doit être mentionné dans le rapport.

Portée de la vérification

Une organisation développant une application n'implémentera généralement pas toutes les exigences, certaines pouvant être non pertinentes ou moins importantes selon les fonctionnalités de l'application. Le vérificateur doit préciser le périmètre de la vérification, notamment le niveau que l'organisation souhaite atteindre et les exigences incluses. Il doit s'agir de ce qui a été inclus plutôt que de ce qui ne l'a pas été. Il doit également fournir un avis sur les raisons justifiant l'exclusion des exigences non implémentées.

Cela devrait permettre au consommateur d'un rapport de vérification de comprendre le contexte de la vérification et de prendre une décision éclairée sur le niveau de confiance qu'il peut accorder à l'application.

Les organismes de certification peuvent choisir leurs méthodes de test, mais devraient les indiquer dans le rapport, et elles devraient idéalement être reproductibles. Différentes méthodes, comme les tests d'intrusion manuels ou l'analyse du code source, peuvent être utilisées pour vérifier des aspects tels que la validation des entrées, selon l'application et les exigences.

Mécanismes de vérification

Plusieurs techniques différentes peuvent être utilisées pour vérifier une même exigence ASVS. Outre les tests d'intrusion (utilisant des "credentials" valides pour une couverture complète de l'application), la vérification des exigences ASVS peut nécessiter l'accès à la documentation, au code source, à la configuration et aux personnes impliquées dans le processus de développement. Ceci est particulièrement vrai pour la vérification des exigences L2 et L3. Il est courant de fournir des preuves solides des résultats au moyen d'une documentation détaillée, pouvant inclure des documents de travail, des captures d'écran, des scripts et des journaux de tests. L'exécution d'un outil automatisé sans tests approfondis ne suffit pas à obtenir la certification, car chaque exigence doit être testée de manière vérifiable.

L'utilisation de l'automatisation pour vérifier les exigences ASVS est un sujet qui suscite un intérêt constant. Il est donc important de clarifier certains points relatifs aux tests automatisés et en boîte noire.

Le rôle des outils de test de sécurité automatisés

Lorsque des outils de tests de sécurité automatisés, tels que les tests de sécurité dynamiques et statiques des applications (DAST et SAST), sont correctement implémentés dans le pipeline de développement, ils peuvent identifier des problèmes de sécurité qui ne devraient jamais exister. Cependant, sans une configuration et un réglage minutieux, ils n'offriront pas la couverture requise et leur niveau de bruit empêchera l'identification et la résolution des véritables problèmes de sécurité.

Bien que cela puisse couvrir certaines des exigences techniques les plus basiques et les plus simples, telles que celles relatives à l'encodage ou à l'assainissement de sortie, il est essentiel de noter que ces outils ne seront pas entièrement en mesure de vérifier bon nombre des exigences ASVS les plus complexes ou celles liées à la logique métier et au contrôle d'accès.

Pour les exigences moins simples, l'automatisation reste probablement envisageable, mais des vérifications spécifiques à l'application devront être écrites pour y parvenir. Ces vérifications peuvent être similaires aux tests unitaires et d'intégration déjà utilisés par l'organisation. Il est donc possible d'utiliser l'infrastructure d'automatisation des tests existante pour écrire ces tests spécifiques à ASVS. Bien que cela nécessite un investissement à court terme, les avantages à long terme de la vérification continue des exigences ASVS seront significatifs.

En résumé, testable en utilisant l'automatisation != exécuter un outil sur étagère.

Le rôle des tests de pénétration

Bien que L1 dans la version 4.0 ait été optimisé pour les tests de « boîte noire » (sans documentation et sans source), même dans ce cas, le standard était clair sur le fait qu'il ne s'agissait pas d'une activité d'assurance efficace et qu'elle devait être activement découragée.

Les tests sans accès aux informations supplémentaires nécessaires constituent un mécanisme sous-productifs et inefficaces de vérification de la sécurité, car ils ne permettent pas d'examiner le code source, d'identifier les menaces et les contrôles manquants, et d'effectuer un test beaucoup plus approfondi dans un délai plus court.

Il est fortement encouragé d'effectuer des tests d'intrusion basés sur la documentation ou le code source (hybrides), avec un accès complet aux développeurs et à la documentation de l'application, plutôt que des tests d'intrusion traditionnels. Cela sera certainement nécessaire pour vérifier de nombreuses exigences ASVS.

Modifications par rapport à la version 4.x

Introduction

Les utilisateurs familiarisés avec la version 4.x du référentiel trouveront utile de consulter les principales modifications apportées à la version 5.0, notamment les mises à jour de contenu, de portée et de philosophie sous-jacente.

Sur les 286 exigences de la version 4.0.3, seules 11 restent inchangées, tandis que 15 ont subi des ajustements grammaticaux mineurs sans altérer leur sens. Au total, 109 exigences (38 %) ne sont plus des exigences distinctes dans la version 5.0 : 50 ont été simplement supprimées, 28 ont été supprimées car doublons et 31 ont été fusionnées avec d'autres exigences. Les autres ont été révisées. Même les exigences n'ayant pas subi de modifications substantielles ont des identifiants différents à la suite d'une réorganisation ou une restructuration.

Pour faciliter l'adoption de la version 5.0, des documents de correspondance sont fournis pour aider les utilisateurs à identifier les correspondances entre les exigences de la version 4.x et celles de la version 5.0. Ces correspondances ne sont pas liées au numéro de version et peuvent être mises à jour ou clarifiées si nécessaire.

Philosophie des exigences

Portée et orientation

La version 4.x incluait des exigences qui ne correspondaient pas au périmètre prévu du référentiel ; celles-ci ont été supprimées. Les exigences qui ne répondaient pas aux critères du périmètre de la version 5.0 ou qui n'étaient pas vérifiables ont également été exclues.

Priorité aux objectifs de sécurité plutôt qu'aux mécanismes

Dans la version 4.x, de nombreuses exigences se concentraient sur des mécanismes spécifiques plutôt que sur les objectifs de sécurité sous-jacents. Dans la version 5.0, les exigences sont centrées sur les objectifs de sécurité, ne référençant des mécanismes particuliers que lorsqu'ils constituent la seule solution pratique, ou les fournissant à titre d'exemple ou de conseil complémentaire.

Cette approche reconnaît que plusieurs méthodes peuvent exister pour atteindre un objectif de sécurité donné et évite toute prescription inutile susceptible de limiter la flexibilité organisationnelle.

De plus, les exigences traitant d'une même préoccupation de sécurité ont été consolidées lorsque cela était opportun.

Décisions de sécurité documentées

Bien que le concept de décisions de sécurité documentées puisse paraître nouveau dans la version 5.0, il s'agit d'une évolution des exigences antérieures liées à l'application des politiques et à la modélisation des menaces dans la version 4.0. Auparavant, certaines exigences exigeaient implicitement une analyse pour éclairer la mise en œuvre des contrôles de sécurité, comme la détermination des connexions réseau autorisées.

Afin de garantir la disponibilité des informations nécessaires à la mise en œuvre et à la vérification, ces attentes sont désormais explicitement définies comme des exigences de documentation, ce qui les rend claires, actionnables et vérifiables.

Modifications structurelles et nouveaux chapitres

Plusieurs chapitres de la version 5.0 introduisent du contenu entièrement nouveau :

- OAuth et OIDC – Compte tenu de l'adoption généralisée de ces protocoles pour la délégation d'accès et l'authentification unique, des exigences spécifiques ont été ajoutées pour répondre aux divers scénarios rencontrés par les développeurs. Ce domaine pourrait à terme évoluer vers une norme autonome, similaire à la gestion des exigences mobiles et IoT dans les versions précédentes.

- WebRTC – Avec la popularité croissante de cette technologie, ses considérations et défis de sécurité spécifiques sont désormais abordés dans une section dédiée.

Des efforts ont également été déployés pour garantir que les chapitres et sections soient organisés autour d'ensembles cohérents d'exigences connexes.

Cette restructuration a conduit à la création de chapitres supplémentaires :

- Jetons autonomes – Auparavant regroupés sous la rubrique « gestion de session », les jetons autonomes sont désormais reconnus comme un mécanisme distinct et un élément fondamental de la communication sans état (comme dans OAuth et OIDC). En raison de leurs implications spécifiques en matière de sécurité, elles sont traitées dans un chapitre dédié, avec l'introduction de nouvelles exigences dans la version 5.x.
- Sécurité du frontend web – Avec la complexité croissante des applications basées sur un navigateur et l'essor des architectures exclusivement basées sur des API, les exigences de sécurité du frontend ont été séparées dans un chapitre dédié.
- Codage et architecture sécurisés – Les nouvelles exigences concernant les pratiques de sécurité générales qui ne cadraient pas avec les chapitres existants ont été regroupées ici.

D'autres modifications organisationnelles ont été apportées à la version 5.0 afin de clarifier l'intention. Par exemple, les exigences de validation des entrées ont été déplacées avec la logique métier, reflétant ainsi leur rôle dans l'application des règles métier, plutôt que d'être regroupées avec le nettoyage et l'encodage.

L'ancien chapitre « Architecture » de la version 1 a été supprimé. Sa section initiale contenait des exigences hors du périmètre, tandis que les sections suivantes ont été redistribuées aux chapitres concernés, les exigences étant dédupliquées et clarifiées si nécessaire.

Suppression des correspondances directes avec d'autres normes

Les correspondances directes avec d'autres normes ont été supprimées du corps du référentiel. L'objectif est de préparer une correspondance avec le projet OWASP Common Requirement Enumeration (CRE), qui reliera ASVS à divers projets OWASP et normes externes.

Les correspondances directes avec CWE et NIST ne sont plus maintenues, comme expliqué ci-dessous.

Couplage réduit avec les directives du NIST sur l'identité numérique

Les directives du NIST sur l'identité numérique (SP 800-63) servent depuis longtemps de référence pour les contrôles d'authentification et d'autorisation. Dans la version 4.x, certains chapitres étaient étroitement alignés sur la structure et la terminologie du NIST.

Si ces directives demeurent une référence importante, un alignement strict a engendré des difficultés, notamment une terminologie moins reconnue, la duplication d'exigences similaires et des correspondances incomplètes. La version 5.0 s'éloigne de cette approche pour plus de clarté et de pertinence.

S'éloigner de l'énumération des faiblesses communes (CWE)

L'Énumération des Faiblesses Communes (CWE) fournit une taxonomie utile des faiblesses de sécurité logicielle. Cependant, des défis tels que les CWE par catégorie, les difficultés de correspondance des exigences à une CWE unique et la présence de mappages imprécis dans la version 4.x ont conduit à la décision d'abandonner les mappages CWE directs dans la version 5.0.

Repenser les définitions de niveaux

La version 4.x décrivait les niveaux comme L1 (« Minimum »), L2 (« Standard ») et L3 (« Avancé »), ce qui implique que toutes les applications manipulant des données sensibles doivent au moins atteindre le niveau L2.

La version 5.0 corrige plusieurs problèmes liés à cette approche, décrits dans les paragraphes suivants.

En pratique, alors que la version 4.x utilisait des graduations pour les indicateurs de niveau, la version 5.x utilise un simple numéro pour tous les formats du référentiel, notamment Markdown, PDF, DOCX, CSV, JSON et XML. Pour des raisons de rétrocompatibilité, les anciennes versions des sorties CSV, JSON et XML utilisant encore des graduations sont également générées.

Niveau d'entrée simplifié

Les retours ont indiqué que le nombre important d'exigences de niveau 1 (environ 120), combiné à sa désignation comme niveau « minimum », insuffisant pour la plupart des applications, freinait l'adoption. La version 5.0 vise à lever cet obstacle en définissant le niveau 1 principalement autour des exigences de défense de première couche, ce qui se traduit par des exigences plus claires et moins nombreuses à ce niveau. À titre d'exemple, la version 4.0.3 comptait 128 exigences de niveau 1 sur un total de 278, soit 46 %. La version 5.0.0 compte 70 exigences de niveau 1 sur un total de 345, soit 20 %.

L'illusion de la testabilité

Un facteur clé dans le choix des contrôles de niveau 1 dans la version 4.x était leur aptitude à être évalués par des tests d'intrusion externes de type « boîte noire ». Cependant, cette approche n'était pas totalement conforme à l'objectif du niveau 1 en tant qu'ensemble minimal de contrôles de sécurité. Certains utilisateurs ont fait valoir que le niveau 1 était insuffisant pour sécuriser les applications, tandis que d'autres ont trouvé qu'il était trop difficile à tester.

S'appuyer sur la testabilité comme critère est à la fois relatif et parfois trompeur. Le fait qu'une exigence soit testable ne garantit pas qu'elle puisse être testée de manière automatisée ou simple. De plus, les exigences les plus facilement testables ne sont pas toujours celles qui ont le plus grand impact sur la sécurité ou les plus simples à mettre en œuvre.

Par conséquent, dans la version 5.0, les décisions de niveau ont été prises principalement en fonction de la réduction des risques et en tenant compte de l'effort de mise en œuvre.

Pas seulement une question de risque

L'utilisation de niveaux prescriptifs, basés sur les risques, imposant un niveau spécifique à certaines applications s'est avérée trop rigide. En pratique, la priorisation et la mise en œuvre des contrôles de sécurité dépendent de multiples facteurs, notamment la réduction des risques et les efforts nécessaires à leur mise en œuvre.

Par conséquent, les organisations sont encouragées à atteindre le niveau qu'elles estiment devoir atteindre en fonction de leur maturité et du message qu'elles souhaitent transmettre à leurs utilisateurs.

V1 Encodage et nettoyage

Objectif de contrôle

Ce chapitre aborde les failles de sécurité les plus courantes des applications web liées au traitement non sécurisé de données non fiables. Ces failles peuvent entraîner diverses vulnérabilités techniques, où les données non fiables sont interprétées selon les règles syntaxiques de l'interpréteur concerné.

Pour les applications web modernes, il est toujours préférable d'utiliser des API plus sûres, telles que les requêtes paramétrées, l'échappement automatique ou les Framework de création de modèles. Dans le cas contraire, un encodage, un échappement ou un nettoyage des sorties soigneusement effectués deviennent essentiels à la sécurité de l'application.

La validation des entrées constitue un mécanisme de défense en profondeur contre les contenus inattendus ou dangereux. Cependant, son objectif principal étant de garantir que le contenu entrant répond aux attentes fonctionnelles et métier, les exigences afférentes sont décrites dans le chapitre « Validation et logique métier ».

V1.1 Architecture d'encodage et de nettoyage

Les sections ci-dessous présentent les exigences spécifiques à la syntaxe ou à l'interpréteur pour traiter en toute sécurité les contenus non sécurisés afin d'éviter les failles de sécurité. Ces exigences précisent l'ordre et le lieu de traitement. Elles visent également à garantir que les données soient stockées dans leur état d'origine et ne soient pas stockées sous une forme encodée ou échappée (par exemple, encodage HTML), afin d'éviter les problèmes de double encodage.

#	Description	Niveau
1.1.1	Vérifiez que l'entrée est décodée ou non échappée vers une forme canonique une seule fois, elle n'est décodée que lorsque des données codées sous cette forme sont attendues, et que cela est fait avant de traiter davantage l'entrée, par exemple, elle n'est pas effectuée après la validation ou la désinfection de l'entrée.	2
1.1.2	Vérifiez que l'application effectue l'encodage et l'échappement de sortie soit comme étape finale avant d'être utilisée par l'interpréteur pour lequel elle est destinée, soit par l'interpréteur lui-même.	2

V1.2 Prévention des injections

L'encodage ou l'échappement de sortie, effectué à proximité d'un contexte potentiellement dangereux, est essentiel pour la sécurité de toute application. Généralement, l'encodage et l'échappement de sortie ne sont pas conservés, mais servent à sécuriser la sortie pour une utilisation immédiate dans l'interpréteur approprié. Une tentative d'exécution trop précoce peut entraîner une altération du contenu ou rendre l'encodage ou l'échappement inefficaces.

Dans de nombreux cas, les bibliothèques logicielles incluent des fonctions sûres ou plus sûres qui exécutent cela automatiquement, même s'il est nécessaire de s'assurer qu'elles sont correctes pour le contexte actuel.

#	Description	Niveau
1.2.1	Vérifiez que l'encodage de sortie d'une réponse HTTP, d'un document HTML ou d'un document XML est pertinent pour le contexte requis, comme l'encodage des caractères pertinents pour les éléments HTML, les attributs HTML, les commentaires HTML, les CSS ou les champs d'en-tête HTTP, pour éviter de modifier la structure du message ou du document.	1
1.2.2	Vérifiez que, lors de la création dynamique d'URL, les données non fiables sont encodées en fonction de leur contexte (par exemple, encodage d'URL ou encodage base64url pour	1

#	Description	Niveau
	les paramètres de requête ou de chemin). Assurez-vous que seuls les schémas d'URL sûrs sont autorisés (par exemple, interdire javascript: ou data:).	
1.2.3	Vérifiez que l'encodage ou l'échappement de sortie est utilisé lors de la création dynamique de contenu JavaScript (y compris JSON), pour éviter de modifier la structure du message ou du document (pour éviter l'injection JavaScript et JSON).	1
1.2.4	Vérifiez que la sélection de données ou les requêtes de base de données (par exemple, SQL, HQL, NoSQL, Cypher) utilisent des requêtes paramétrées, des ORM, des Framework d'entités ou sont protégées contre les injections SQL et autres attaques par injection de base de données. Ceci est également pertinent lors de l'écriture de procédures stockées.	1
1.2.5	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 du système d'exploitation paramétrées ou utilisent un encodage de sortie de ligne de commande contextuelle.	1
1.2.6	Vérifiez que l'application protège contre les vulnérabilités d'injection LDAP ou que des contrôles de sécurité spécifiques pour empêcher l'injection LDAP ont été mis en œuvre.	2
1.2.7	Vérifiez que l'application est protégée contre les attaques par injection XPath en utilisant la paramétrisation des requêtes ou des requêtes précompilées.	2
1.2.8	Vérifiez que les processeurs LaTeX sont configurés de manière sécurisée (par exemple, en n'utilisant pas l'indicateur « --shell-escape ») et qu'une liste blanche de commandes est utilisée pour empêcher les attaques par injection LaTeX.	2
1.2.9	Vérifiez que l'application échappe les caractères spéciaux dans les expressions régulières (généralement à l'aide d'une barre oblique inverse) pour éviter qu'ils ne soient mal interprétés comme des métacaractères.	2
1.2.10	Vérifiez que l'application est protégée contre les injections CSV et injections de formules. L'application doit respecter les règles d'échappement définies dans les sections 2.6 et 2.7 de la RFC 4180 lors de l'export de contenu CSV. De plus, lors de l'export au format CSV ou vers d'autres formats de tableur (tels que XLS, XLSX ou ODF), les caractères spéciaux (notamment « = », « + », « - », « @ », « \t » (tabulation) et « \0 » (caractère null)) doivent être protégés par une apostrophe s'ils apparaissent en premier dans une valeur de champ.	3

Remarque : L'utilisation de requêtes paramétrées ou l'échappement SQL ne sont pas toujours suffisants. Les parties de requête telles que les noms de table et de colonne (y compris les noms de colonne « ORDER BY ») ne peuvent pas être échappées. L'inclusion de données utilisateur échappées dans ces champs entraîne l'échec des requêtes ou une injection SQL.

V1.3 Nettoyage

La protection idéale contre l'utilisation de contenu non fiable dans un contexte dangereux est d'utiliser un encodage ou un échappement spécifique au contexte, qui conserve la même signification sémantique du contenu dangereux mais le rend sûr pour une utilisation dans ce contexte particulier, comme expliqué plus en détail dans la section précédente.

Lorsque cela n'est pas possible, un nettoyage devient nécessaire, supprimant les caractères ou contenus potentiellement dangereux. Dans certains cas, cela peut modifier la signification sémantique de la saisie, mais pour des raisons de sécurité, il peut n'y avoir aucune autre solution.

#	Description	Niveau
1.3.1	Vérifiez que toutes les entrées HTML non fiables provenant d'éditeurs WYSIWYG ou similaires sont nettoyées à l'aide d'une bibliothèque ou d'une fonctionnalité de Framework de nettoyage HTML bien connue et sécurisée.	1
1.3.2	Vérifiez que l'application évite d'utiliser eval() ou d'autres fonctionnalités d'exécution de code dynamique telles que Spring Expression Language (SpEL). En l'absence d'alternative, toute saisie utilisateur incluse doit être nettoyée avant d'être exécutée.	1
1.3.3	Vérifiez que les données transmises à un contexte potentiellement dangereux sont préalablement nettoyées pour appliquer des mesures de sécurité, telles qu'autoriser uniquement des caractères sûrs pour ce contexte et raccourcir des entrées trop longues.	2
1.3.4	Vérifiez que le contenu Scalable Vector Graphics (SVG) scriptable fourni par l'utilisateur est validé ou nettoyé pour contenir uniquement des balises et des attributs (tels que des graphiques de dessin) qui sont sûrs pour l'application, par exemple, ne contiennent pas de scripts et d'objets étrangers.	2
1.3.5	Vérifiez que l'application nettoie ou désactive le contenu du langage de modèle d'expression ou de script fourni par l'utilisateur, tel que les feuilles de style CSS, le Markdown ou XSL, BBCode ou similaire.	2
1.3.6	Vérifiez que l'application protège contre les attaques de falsification de requête côté serveur (SSRF), en validant les données non fiables par rapport à une liste blanche de protocoles, de domaines, de chemins et de ports et en nettoyant les caractères potentiellement dangereux avant d'utiliser les données pour appeler un autre service.	2
1.3.7	Vérifiez que l'application se protège contre les attaques par injection de modèles en interdisant la création de modèles basés sur des entrées non fiables. En l'absence d'alternative, toute entrée non fiable incluse dynamiquement lors de la création du modèle doit être nettoyée ou rigoureusement validée.	2
1.3.8	Vérifiez que l'application nettoie correctement les entrées non fiables avant utilisation dans les requêtes Java Naming and Directory Interface (JNDI) et que JNDI est configuré de manière sécurisée pour empêcher les attaques par injection JNDI.	2
1.3.9	Vérifiez que l'application nettoie le contenu avant qu'il ne soit envoyé à memcache pour éviter les attaques par injection.	2
1.3.10	Vérifiez que les chaînes de format susceptibles d'être résolues de manière inattendue ou malveillante lors de leur utilisation sont nettoyées avant d'être traitées.	2
1.3.11	Vérifiez que l'application nettoie les entrées utilisateur avant de les transmettre aux systèmes de messagerie pour se protéger contre l'injection SMTP ou IMAP.	2
1.3.12	Vérifiez que les expressions régulières sont exemptes d'éléments provoquant "backtracking" exponentiel et assurez-vous que les entrées non fiables sont nettoyées pour atténuer les attaques ReDoS ou Runaway Regex.	3

V1.4 Mémoire, chaine de caractères et code à mémoire non-managée

Les exigences suivantes traitent des risques associés à une utilisation non sécurisée de la mémoire, qui s'appliquent généralement lorsque l'application utilise un langage système ou un code à mémoire non-managée.

Dans certains cas, il peut être possible d'y parvenir en définissant des indicateurs de compilateur qui activent les protections contre les dépassements de tampon et les avertissements, y compris la randomisation de la pile et

la prévention de l'exécution des données, et qui interrompent la construction si des opérations de pointeur, de mémoire, de chaîne de format, d'entier ou de chaine de caractères non sûres sont trouvées.

#	Description	Niveau
1.4.1	Vérifiez que l'application utilise des chaînes memory-safe, une copie de mémoire et une arithmétique de pointeur renforcée pour détecter ou empêcher les débordements de pile, de tampon ou de tas.	2
1.4.2	Vérifiez que les techniques de validation du signe, de la plage et de l'entrée sont utilisées pour éviter les dépassements d'entiers.	2
1.3.3	Vérifiez que la mémoire et les ressources allouées dynamiquement sont libérées et que les références ou les pointeurs vers la mémoire libérée sont supprimés ou définis à null pour éviter les "dangling pointer" et les vulnérabilités d'utilisation après libération.	2

V1.5 Désérialisation sécurisée

La conversion de données stockées ou transmises en objets applicatifs (désérialisation) a toujours été à l'origine de diverses vulnérabilités par injection de code. Il est important d'effectuer ce processus avec précaution et de manière sécurisée pour éviter ce type de problèmes.

En particulier, certaines méthodes de désérialisation ont été identifiées par la documentation des langages de programmation ou des Framework comme étant non sécurisées et ne peuvent être sécurisées avec des données non fiables. Pour chaque mécanisme utilisé, une vérification rigoureuse doit être effectuée.

#	Description	Niveau
1.5.1	Vérifiez que l'application configure les "parser" XML pour utiliser une configuration restrictive et que les fonctionnalités non sûres telles que la résolution d'entités externes sont désactivées pour empêcher les attaques XML eXternal Entity (XXE).	1
1.5.2	Vérifiez que la désérialisation des données non fiables garantit une gestion sécurisée des entrées, par exemple en utilisant une liste blanche de types d'objets ou en limitant les types d'objets définis par le client, afin d'empêcher les attaques par désérialisation. Les mécanismes de désérialisation explicitement définis comme non sécurisés ne doivent pas être utilisés avec des entrées non fiables.	2
1.5.3	Vérifiez que les différents "parser" utilisés dans l'application pour le même type de données (par exemple, les "parser" JSON, les "parser" XML, les "parser" d'URL) effectuent l'analyse de manière cohérente et utilisent le même mécanisme d'encodage de caractères pour éviter des problèmes tels que les vulnérabilités d'interopérabilité JSON ou un comportement d'analyse d'URI ou de fichier différent exploité dans les attaques d'inclusion de fichiers à distance (RFI) ou de falsification de requête côté serveur (SSRF).	3

Références

Pour plus d'informations, voir également :

- OWASP LDAP Injection Prevention Cheat Sheet
- OWASP Cross Site Scripting Prevention Cheat Sheet
- OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet

- OWASP XML External Entity Prevention Cheat Sheet
- OWASP Web Security Testing Guide: Client-Side Testing
- OWASP Java Encoding Project
- DOMPurify - Client-side HTML Sanitization Library
- RFC4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files

Pour plus d'informations, notamment sur les problèmes de désérialisation ou d'analyse, veuillez consulter :

- OWASP Deserialization Cheat Sheet
- An Exploration of JSON Interoperability Vulnerabilities
- Orange Tsai - A New Era of SSRF Exploiting URL Parser In Trending Programming Languages

V2 Validation et logique métier

Objectif de contrôle

Ce chapitre vise à garantir qu'une application vérifiée répond aux objectifs de haut niveau suivants :

- Les entrées reçues par l'application correspondent aux attentes métier ou fonctionnelles.
- Le flux logique métier est séquentiel, traité dans l'ordre et ne peut pas être contourné.
- La logique métier inclut des limites et des contrôles pour détecter et prévenir les attaques automatisées, telles que les transferts continus de petits fonds ou l'ajout d'un million d'amis un par un.
- Les flux logiques métier à haute valeur ajoutée ont pris en compte les cas d'abus et les acteurs malveillants, et disposent de protections contre l'usurpation d'identité, la falsification, la divulgation d'informations et les attaques par élévation de privilèges.

V2.1 Validation et documentation de la logique métier

La documentation de validation et de logique métier doit définir clairement les limites de la logique métier, les règles de validation et la cohérence contextuelle des éléments de données combinés, afin que ce qui doit être implémenté dans l'application soit clair.

#	Description	Niveau
2.1.1	Vérifiez que la documentation de l'application définit des règles de validation des entrées pour vérifier la validité des éléments de données par rapport à la structure attendue. Il peut s'agir de formats de données courants tels que des numéros de carte de crédit, des adresses e-mail ou des numéros de téléphone, ou d'un format de données interne.	1
2.1.2	Vérifiez que la documentation de l'application définit comment valider la cohérence logique et contextuelle des éléments de données combinés, par exemple en vérifiant que la banlieue et le code postal correspondent.	2
2.1.3	Vérifiez que les attentes en matière de limites et de validations de la logique métier sont documentées, y compris par utilisateur et globalement dans l'application.	2

V2.2 Validation des entrées

Des contrôles de validation des entrées efficaces renforcent les attentes métier ou fonctionnelles concernant le type de données que l'application s'attend à recevoir. Cela garantit une bonne qualité des données et réduit la surface d'attaque. Cependant, cela ne supprime ni ne remplace la nécessité d'utiliser un codage, un paramétrage ou un nettoyage corrects lors de l'utilisation des données dans un autre composant ou pour leur présentation en sortie.

Dans ce contexte, les « entrées » peuvent provenir d'une grande variété de sources, notamment des champs de formulaire HTML, des requêtes REST, des paramètres d'URL, des champs d'en-tête HTTP, des cookies, des fichiers sur disque, des bases de données et des API externes.

Un contrôle de logique métier peut vérifier qu'une entrée particulière est un nombre inférieur à 100. Une attente fonctionnelle peut vérifier qu'un nombre est inférieur à un certain seuil, car ce nombre contrôle le nombre de fois qu'une boucle particulière aura lieu, et un nombre élevé pourrait entraîner un traitement excessif et une condition potentielle de déni de service.

Bien que la validation de schéma ne soit pas explicitement obligatoire, elle peut être le mécanisme le plus efficace pour une couverture de validation complète des API HTTP ou d'autres interfaces qui utilisent JSON ou XML.

Veuillez noter les points suivants concernant la validation du schéma :

- La « version publiée » de la spécification de validation du schéma JSON est considérée comme prête pour la production, mais pas à proprement parler « stable ». Lorsque vous utilisez la validation du schéma JSON, assurez-vous qu'il n'y a aucune lacune par rapport aux instructions des exigences ci-dessous.
- Toutes les bibliothèques de validation de schéma JSON utilisées doivent également être surveillées et mises à jour si nécessaire une fois la norme formalisée.
- La validation DTD ne doit pas être utilisée et l'évaluation DTD du Framework doit être désactivée pour éviter les problèmes liés aux attaques XXE contre les DTD.

#	Description	Niveau
2.2.1	Vérifier que les entrées sont validées afin de respecter les attentes métier ou fonctionnelles. Cette validation doit être basée sur une liste de valeurs, de modèles et de plages autorisées, ou sur la comparaison des entrées avec une structure attendue et des limites logiques selon des règles prédéfinies. Pour le niveau 1, cette validation peut se concentrer sur les entrées utilisées pour prendre des décisions métier ou de sécurité spécifiques. À partir du niveau 2, elle doit s'appliquer à toutes les entrées.	1
2.2.2	Vérifiez que l'application est conçue pour appliquer la validation des entrées au niveau d'une couche de service de confiance. Bien que la validation côté client améliore l'ergonomie et doit être encouragée, elle ne doit pas être considérée comme un contrôle de sécurité.	1
2.2.3	Vérifiez que l'application garantit que les combinaisons d'éléments de données associés sont raisonnables selon les règles prédéfinies.	2

V2.3 Sécurité de la logique métier

Cette section examine les exigences clés pour garantir que l'application applique les processus de logique métier de manière correcte et n'est pas vulnérable aux attaques qui exploitent la logique et le flux de l'application.

#	Description	Niveau
2.3.1	Vérifiez que l'application traitera uniquement les flux de logique métier pour le même utilisateur dans l'ordre séquentiel attendu et sans sauter d'étapes.	1
2.3.2	Vérifiez que les limites de la logique métier sont implémentées conformément à la documentation de l'application, afin d'éviter que des failles de la logique métier ne soient pas exploitées.	2
2.3.3	Vérifiez que les transactions sont utilisées au niveau de la logique métier de telle sorte qu'une opération de logique métier réussisse dans son intégralité ou qu'elle soit restaurée à l'état correct précédent.	2
2.3.4	Vérifiez que les mécanismes de verrouillage au niveau de la logique métier sont utilisés pour garantir que les ressources en quantité limitée (telles que les sièges de théâtre ou les créneaux de livraison) ne peuvent pas être réservées deux fois en manipulant la logique de l'application.	2
2.3.5	Vérifiez que les flux logiques métier à forte valeur ajoutée nécessitent l'approbation de plusieurs utilisateurs afin d'éviter toute action non autorisée ou accidentelle. Cela peut inclure, sans s'y limiter, les transferts monétaires importants, les approbations de contrats,	3

#	Description	Niveau
	l'accès à des informations classifiées ou les contournements de sécurité dans le secteur manufacturier.	

V2.4 Anti-automatisation

Cette section comprend des contrôles anti-automatisation pour garantir que les interactions de type humaine sont requises et que les demandes automatisées excessives sont évitées.

#	Description	Niveau
2.4.1	Vérifiez que des contrôles anti-automatisation sont en place pour protéger contre les appels excessifs aux fonctions d'application qui pourraient conduire à l'exfiltration de données, à la création de données inutiles, à l'épuisement des quotas, aux violations de limites de débit, au déni de service ou à la surutilisation de ressources coûteuses.	2
2.4.2	Vérifiez que les flux logiques métier nécessitent un timing humain réaliste, évitant ainsi des soumissions de transactions excessivement rapides.	3

Références

Pour plus d'informations, voir également :

- OWASP Web Security Testing Guide 4.2: Input Validation Testing
- OWASP Web Security Testing Guide 4.2: Business Logic Testing
- La lutte contre l'automatisation peut se faire de différentes manières, notamment par l'utilisation de l'OWASP Automated Threats to Web Applications
- OWASP Input Validation Cheat Sheet
- JSON Schema

V3 Sécurité du frontend Web

Objectif de contrôle

Cette catégorie se concentre sur les exigences visant à protéger contre les attaques exécutées via un frontend web. Ces exigences ne s'appliquent pas aux solutions "machine-to-machine".

V3.1 Documentation sur la sécurité du frontend Web

Cette section décrit les fonctionnalités de sécurité du navigateur qui doivent être spécifiées dans la documentation de l'application.

#	Description	Niveau
3.1.1	Vérifiez que la documentation de l'application indique les fonctionnalités de sécurité attendues que les navigateurs utilisant l'application doivent prendre en charge (telles que HTTPS, HTTP Strict Transport Security (HSTS), Content Security Policy (CSP) et autres mécanismes de sécurité HTTP pertinents). Elle doit également définir le comportement de l'application lorsque certaines de ces fonctionnalités ne sont pas disponibles (par exemple, avertir l'utilisateur ou bloquer l'accès).	3

V3.2 Interprétation non intentionnelle du contenu

Le rendu de contenu ou de fonctionnalité dans un contexte incorrect peut entraîner l'exécution ou l'affichage de contenu malveillant.

#	Description	Niveau
3.2.1	Vérifiez que des contrôles de sécurité sont en place pour empêcher les navigateurs d'afficher du contenu ou des fonctionnalités dans les réponses HTTP dans un contexte incorrect (par exemple, lorsqu'une API, un fichier téléchargé par l'utilisateur ou une autre ressource est demandée directement). Les contrôles possibles peuvent inclure : ne pas transmettre le contenu sauf si les champs d'en-tête de requête HTTP (tels que Sec-Fetch-*) indiquent qu'il s'agit du contexte correct, utiliser la directive sandbox du champ d'en-tête Content-Security-Policy ou utiliser le type de disposition de pièce jointe dans le champ d'en-tête Content-Disposition.	1
3.2.2	Vérifiez que le contenu destiné à être affiché sous forme de texte, plutôt que rendu sous forme de HTML, est géré à l'aide de fonctions de rendu sécurisées (telles que createTextNode ou textContent) pour empêcher l'exécution involontaire de contenu tel que HTML ou JavaScript.	1
3.2.3	Vérifiez que l'application évite l'écrasement du DOM lors de l'utilisation de JavaScript côté client en utilisant des déclarations de variables explicites, en effectuant une vérification de type stricte, en évitant de stocker des variables globales sur l'objet document et en implémentant l'isolation de l'espace de nommage.	3

V3.3 Configuration des cookies

Cette section décrit les exigences de configuration sécurisée des cookies sensibles afin de fournir un niveau d'assurance plus élevé qu'ils ont été créés par l'application elle-même et d'empêcher que leur contenu ne fuite ou ne soit modifié de manière inappropriée.

#	Description	Niveau
3.3.1	Vérifiez que les cookies ont l'attribut « Secure » défini et si le préfixe « __Host- » n'est pas utilisé pour le nom du cookie, le préfixe « __Secure- » doit être utilisé pour le nom du cookie.	1
3.3.2	Vérifiez que la valeur de l'attribut « SameSite » de chaque cookie est définie en fonction de l'objectif du cookie, afin de limiter l'exposition aux attaques de réparation de l'interface utilisateur et aux attaques de falsification de requêtes basées sur le navigateur, communément appelées falsification de requêtes intersites (CSRF).	2
3.3.3	Vérifiez que les cookies ont le préfixe « __Host- » pour le nom du cookie, sauf s'ils sont explicitement conçus pour être partagés avec d'autres hôtes.	2
3.3.4	Vérifiez que si la valeur d'un cookie n'est pas censée être accessible aux scripts côté client (comme un jeton de session), le cookie doit avoir l'attribut « HttpOnly » défini et la même valeur (par exemple, un jeton de session) ne doit être transférée au client que via le champ d'en-tête « Set-Cookie ».	2
3.3.5	Vérifiez que lorsque l'application écrit un cookie, la longueur combinée du nom et de la valeur du cookie ne dépasse pas 4 096 octets. Les cookies trop volumineux ne seront pas stockés par le navigateur et ne seront donc pas envoyés avec les requêtes, empêchant ainsi l'utilisateur d'utiliser les fonctionnalités de l'application qui reposent sur ce cookie.	3

V3.4 En-têtes du mécanisme de sécurité du navigateur

Cette section décrit les en-têtes de sécurité qui doivent être définis sur les réponses HTTP pour activer les fonctionnalités et restrictions de sécurité du navigateur lors du traitement des réponses de l'application.

#	Description	Niveau
3.4.1	Vérifiez qu'un champ d'en-tête Strict-Transport-Security est inclus dans toutes les réponses afin d'appliquer une politique HTTP Strict Transport Security (HSTS). Une durée maximale d'au moins un an doit être définie, et pour les niveaux L2 et supérieurs, la politique doit également s'appliquer à tous les sous-domaines.	1
3.4.2	Vérifiez que le champ d'en-tête Access-Control-Allow-Origin du partage de ressources cross-origine (CORS) est une valeur fixe de l'application ou, si la valeur du champ d'en-tête de requête HTTP Origin est utilisée, qu'elle est validée par rapport à une liste d'origines approuvées. Si « Access-Control-Allow-Origin: * » doit être utilisé, vérifiez que la réponse ne contient aucune information sensible.	1
3.4.3	Vérifiez que les réponses HTTP incluent un champ d'en-tête de réponse Content-Security-Policy qui définit des directives garantissant que le navigateur charge et exécute uniquement du contenu ou des ressources fiables, afin de limiter l'exécution de JavaScript malveillant. Au minimum, une politique globale doit être utilisée, incluant les directives object-src 'none' et base-uri 'none', et définissant soit une liste blanche, soit des valeurs	2

#	Description	Niveau
	nonces ou des hachages. Pour une application L3, une politique par réponse avec des valeurs nonces ou des hachages doit être définie.	
3.4.4	Vérifiez que toutes les réponses HTTP contiennent un champ d'en-tête « X-Content-Type-Options: nosniff ». Cela indique aux navigateurs de ne pas utiliser l'analyse de contenu ni la détection du type MIME pour la réponse donnée, et d'exiger que la valeur du champ d'en-tête « Content-Type » de la réponse corresponde à la ressource de destination. Par exemple, la réponse à une demande de style n'est acceptée que si le type de contenu de la réponse est « text/css ». Cela permet également au navigateur d'utiliser la fonctionnalité de blocage de lecture inter-origines (CORB).	2
3.4.5	Vérifiez que l'application définit une politique de référencement afin d'empêcher la fuite de données techniquement sensibles vers des services tiers via le champ d'en-tête de requête HTTP « Referer ». Cela peut être effectué via le champ d'en-tête de réponse HTTP « Referrer-Policy » ou via les attributs d'élément HTML. Les données sensibles peuvent inclure le chemin d'accès et les données de requête dans l'URL, et pour les applications internes non publiques, le nom d'hôte.	2
3.4.6	Vérifiez que l'application Web utilise la directive frame-ancestors du champ d'en-tête Content-Security-Policy pour chaque réponse HTTP afin de garantir qu'elle ne peut pas être intégrée par défaut et que l'intégration de ressources spécifiques n'est autorisée qu'en cas de nécessité. Notez que le champ d'en-tête X-Frame-Options, bien que pris en charge par les navigateurs, est obsolète et n'est pas forcément fiable.	2
3.4.7	Vérifiez que le champ d’en-tête Content-Security-Policy spécifie un emplacement pour signaler les violations.	3
3.4.8	Vérifiez que toutes les réponses HTTP qui lancent le rendu d'un document (comme les réponses de type text/html) incluent le champ d'en-tête 'Cross-Origin-Opener-Policy' avec la directive 'same-origin' ou s'ame-origin-allow-popups', selon les besoins. Cela empêche les attaques abusant de l'accès partagé aux objets Window, telles que le tabnabbing et le comptage d'images.	3

V3.5 Séparation de l'origine du navigateur

Lorsqu'elle accepte une demande d'accès à une fonctionnalité sensible du côté du serveur, l'application doit s'assurer que la demande est initiée par l'application elle-même ou par un tiers de confiance et qu'elle n'a pas été falsifiée par un attaquant.

Les fonctionnalités sensibles dans ce contexte peuvent inclure l'acceptation de soumission de formulaires pour des utilisateurs authentifiés et non authentifiés (comme une demande d'authentification), des opérations de changement d'état ou des fonctionnalités exigeantes en ressources (comme l'exportation de données).

Les principales protections sont les politiques de sécurité du navigateur, comme la politique de même origine pour JavaScript et la logique SameSite pour les cookies. Le mécanisme de requête de pré-vérification cross-origin CORS est une autre protection courante. Ce mécanisme est essentiel pour les "endpoints" conçus pour être appelés depuis une origine différente, mais il peut également constituer un mécanisme utile de prévention contre la falsification de requêtes pour les "endpoints" qui ne sont pas conçus pour être appelés depuis une origine différente.

#	Description	Niveau
3.5.1	Vérifiez que, si l'application ne s'appuie pas sur le mécanisme de requête de pré-vérification cross-origin CORS pour empêcher les requêtes inter-origines non autorisées d'utiliser des	1

#	Description	Niveau
	fonctionnalités sensibles, ces requêtes sont validées afin de garantir leur origine. Cela peut se faire en utilisant et en validant des jetons anti-falsification ou en exigeant des champs d'en-tête HTTP supplémentaires qui ne sont pas des champs d'en-tête de requête CORS safelist. Ceci permet de se protéger contre les attaques de falsification de requêtes basées sur le navigateur, communément appelées falsification de requêtes intersites (CSRF).	
3.5.2	Vérifiez que, si l'application s'appuie sur le mécanisme de requête de pré-vérification cross-origin CORS pour empêcher l'utilisation inter-origines non autorisée de fonctionnalités sensibles, il est impossible d'appeler la fonctionnalité avec une requête qui ne déclenche pas de requête de pré-vérification cross-origin CORS. Cela peut nécessiter de vérifier les valeurs des champs d'en-tête de requête 'Origin' et 'Content-Type' ou d'utiliser un champ d'en-tête supplémentaire qui n'est pas un champ d'en-tête de la liste de sécurité CORS.	1
3.5.3	Vérifiez que les requêtes HTTP destinées aux fonctionnalités sensibles utilisent des méthodes HTTP appropriées, telles que POST, PUT, PATCH ou DELETE, et non des méthodes définies comme « sûres » par la spécification HTTP, telles que HEAD, OPTIONS ou GET. Une validation stricte des champs d'en-tête de requête Sec-Fetch-* peut également être utilisée pour garantir que la requête ne provient pas d'un appel inter-origines inapproprié, d'une requête de navigation ou d'un chargement de ressources (comme une source d'image) inattendu.	1
3.5.4	Vérifiez que des applications distinctes sont hébergées sur des noms d'hôte différents pour tirer parti des restrictions fournies par la politique de même origine, y compris la manière dont les documents ou scripts chargés par une origine peuvent interagir avec les ressources d'une autre origine et les restrictions basées sur le nom d'hôte sur les cookies.	2
3.5.5	Vérifiez que les messages reçus par l'interface postMessage sont rejetés si l'origine du message n'est pas digne de confiance ou si la syntaxe du message n'est pas valide.	2
3.5.6	Vérifiez que la fonctionnalité JSONP n'est activée nulle part dans l'application pour éviter les attaques Cross-Site Script Inclusion (XSSI).	3
3.5.7	Vérifiez que les données nécessitant une autorisation ne sont pas incluses dans les réponses des ressources de script, comme les fichiers JavaScript, pour empêcher les attaques par inclusion de script intersite (XSSI).	3
3.5.8	Vérifiez que les ressources authentifiées (telles que les images, les vidéos, les scripts et autres documents) peuvent être chargées ou intégrées pour le compte de l'utilisateur uniquement lorsque cela est prévu. Cela peut être réalisé en validant strictement les champs d'en-tête de requête HTTP Sec-Fetch-* afin de garantir que la requête ne provient pas d'un appel cross-origin inapproprié, ou en définissant un champ d'en-tête de réponse HTTP Cross-Origin-Resource-Policy restrictif pour indiquer au navigateur de bloquer le contenu renvoyé.	3

V3.6 Intégrité des ressources externes

Cette section fournit des conseils pour l'hébergement sécurisé de contenu sur des sites tiers.

#	Description	Niveau
3.6.1	Vérifiez que les ressources côté client, telles que les bibliothèques JavaScript, les feuilles de style CSS ou les polices web, sont hébergées en externe (par exemple, sur un réseau de diffusion de contenu) uniquement si la ressource est statique et versionnée, et si l'intégrité des sous-ressources (SRI) est utilisée pour valider l'intégrité de la ressource. Si cela n'est pas	3

#	Description	Niveau
	possible, une décision de sécurité documentée doit justifier cette décision pour chaque ressource.	

V3.7 Autres considérations sur la sécurité du navigateur

Cette section comprend divers autres contrôles de sécurité et fonctionnalités de sécurité de navigateur modernes nécessaires pour la sécurité du navigateur côté client.

#	Description	Niveau
3.7.1	Vérifiez que l'application utilise uniquement des technologies côté client toujours prises en charge et considérées comme sécurisées. Parmi les technologies qui ne répondent pas à cette exigence figurent les plugins NSAPI, Flash, Shockwave, ActiveX, Silverlight, NACL ou les applets Java côté client.	2
3.7.2	Vérifiez que l'application redirigera automatiquement l'utilisateur vers un nom d'hôte ou un domaine différent (qui n'est pas contrôlé par l'application) uniquement lorsque la destination apparaît sur une liste blanche.	2
3.7.3	Vérifiez que le domaine de premier niveau de l'application (par exemple, site.tld) est ajouté à la liste de préchargement publique pour HTTP Strict Transport Security (HSTS). Cela garantit que l'utilisation de TLS pour l'application est intégrée directement dans les principaux navigateurs, plutôt que de dépendre uniquement du champ d'en-tête de réponse Strict-Transport-Security.	3
3.7.4	Vérifiez que l'application affiche une notification lorsque l'utilisateur est redirigé vers une URL hors du contrôle de l'application, avec une option pour annuler la navigation.	3
3.7.5	Vérifiez que l'application se comporte comme documenté (par exemple, en avertissant l'utilisateur ou en bloquant l'accès) si le navigateur utilisé pour accéder à l'application ne prend pas en charge les fonctionnalités de sécurité attendues.	3

Références

Pour plus d'informations, voir également :

- Set-Cookie __Host- prefix details
- OWASP Content Security Policy Cheat Sheet
- OWASP Secure Headers Project
- OWASP Cross-Site Request Forgery Prevention Cheat Sheet
- HSTS Browser Preload List submission form
- OWASP DOM Clobbering Prevention Cheat Sheet

V4 API et service Web

Objectif de contrôle

Plusieurs considérations s'appliquent spécifiquement aux applications exposant des API destinées aux navigateurs web ou à d'autres consommateurs (généralement via JSON, XML ou GraphQL). Ce chapitre présente les configurations et mécanismes de sécurité pertinents à appliquer.

Notez que les problèmes d'authentification, de gestion de session et de validation des entrées des autres chapitres s'appliquent également aux API. Ce chapitre ne peut donc pas être sorti de son contexte ni testé de manière isolée.

V4.1 Sécurité du service Web générique

Cette section aborde les considérations générales sur la sécurité des services Web et, par conséquent, les pratiques d'hygiène de base des services Web.

#	Description	Niveau
4.1.1	Vérifiez que chaque réponse HTTP avec un corps de message contient un champ d'en-tête Content-Type qui correspond au contenu réel de la réponse, y compris le paramètre charset pour spécifier un codage de caractères sécurisé (par exemple, UTF-8, ISO-8859-1) conformément aux types de médias IANA, tels que « text/ », « /+xml » et « /xml ».	1
4.1.2	Vérifiez que seuls les terminaux utilisateurs (destinés à l'accès manuel via un navigateur web) redirigent automatiquement de HTTP vers HTTPS, tandis que les autres services ou terminaux n'implémentent pas de redirections transparentes. Cela permet d'éviter qu'un client envoie par erreur des requêtes HTTP non chiffrées, mais que, comme ces requêtes sont automatiquement redirigées vers HTTPS, la fuite de données sensibles passe inaperçue.	2
4.1.3	Vérifiez que tout champ d'en-tête HTTP utilisé par l'application et défini par une couche intermédiaire, telle qu'un équilibreur de charge, un proxy web ou un service backend-for-frontend, ne peut pas être remplacé par l'utilisateur final. Les exemples d'en-têtes peuvent inclure X-Real-IP, X-Forwarded-*, ou X-User-ID.	3
4.1.4	Vérifiez que seules les méthodes HTTP explicitement prises en charge par l'application ou son API (y compris les OPTIONS lors des demandes de contrôle en amont) peuvent être utilisées et que les méthodes inutilisées sont bloquées.	3
4.1.5	Vérifier que les signatures numériques par message sont utilisées pour fournir une assurance supplémentaire en plus des protections de transport pour les demandes ou les transactions qui sont très sensibles ou qui traversent un certain nombre de systèmes.	3

V4.2 Validation de la structure des messages HTTP

Cette section explique comment la structure et les champs d'en-tête d'un message HTTP doivent être validés afin de prévenir les attaques telles que la contrebande de requêtes, le fractionnement de réponses, l'injection d'en-têtes et le déni de service via des messages HTTP trop longs.

Ces exigences sont pertinentes pour le traitement et la génération de messages HTTP généraux, mais sont particulièrement importantes lors de la conversion de messages HTTP entre différentes versions HTTP.

#	Description	Niveau
4.2.1	Vérifiez que tous les composants de l'application (y compris les équilibreurs de charge, les pares-feux et les serveurs d'applications) déterminent les limites des messages HTTP entrants à l'aide du mécanisme approprié à la version HTTP afin d'empêcher la contrebande de requêtes HTTP. Dans HTTP/1.x, si un champ d'en-tête Transfer-Encoding est présent, l'en-tête Content-Length doit être ignoré conformément à la RFC 2616. Avec HTTP/2 ou HTTP/3, si un champ d'en-tête Content-Length est présent, le récepteur doit s'assurer qu'il est cohérent avec la longueur des trames DATA.	2
4.2.2	Vérifier que, lors de la génération de messages HTTP, le champ d'en-tête Content-Length n'entre pas en conflit avec la longueur du contenu déterminée par le cadrage du protocole HTTP, afin d'empêcher les attaques de type « request smuggling ».	2
4.2.3	Vérifiez que l'application n'envoie ni n'accepte de messages HTTP/2 ou HTTP/3 avec des champs d'en-tête spécifiques à la connexion, tels que Transfer-Encoding, afin d'éviter les attaques par fractionnement de la réponse et par injection d'en-tête.	3
4.2.4	Vérifiez que l'application n'accepte que les requêtes HTTP/2 et HTTP/3 dont les champs et valeurs d'en-tête ne contiennent pas de séquences CR (\r), LF (\n) ou CRLF (\r\n), afin d'éviter les attaques par injection d'en-tête.	3
4.2.5	Vérifiez que, si l'application (backend ou frontend) construit et envoie des requêtes, elle utilise des mécanismes de validation, d'assainissement ou autres pour éviter de créer des URI (comme pour les appels API) ou des champs d'en-tête de requête HTTP (comme Authorization ou Cookie) qui sont trop longs pour être acceptés par le composant récepteur. Cela pourrait entraîner un déni de service, par exemple lors de l'envoi d'une requête trop longue (par exemple, un long champ d'en-tête de cookie), ce qui fait que le serveur répond toujours avec un statut d'erreur.	3

V4.3 GraphQL

GraphQL est de plus en plus utilisé pour créer des clients riches en données, peu couplés à divers services backend. Cette section aborde les considérations de sécurité pour GraphQL.

#	Description	Niveau
4.3.1	Vérifiez qu'une liste d'autorisation de requête, une limitation de profondeur, une limitation de quantité ou une analyse des coûts de requête est utilisée pour empêcher le déni de service (DoS) de GraphQL ou d'expression de la couche données en raison de requêtes imbriquées coûteuses.	2
4.3.2	Vérifiez que les requêtes d'introspection GraphQL sont désactivées dans l'environnement de production, sauf si l'API GraphQL est destinée à être utilisée par d'autres parties.	2

V4.4 WebSocket

WebSocket est un protocole de communication qui fournit un canal de communication bidirectionnel simultané via une seule connexion TCP. Il a été normalisé par l'IETF sous la RFC 6455 en 2011 et se distingue de HTTP, bien qu'il soit conçu pour fonctionner sur les ports HTTP 443 et 80.

Cette section fournit les principales exigences de sécurité pour empêcher les attaques liées à la sécurité des communications et à la gestion des sessions qui exploitent spécifiquement ce canal de communication en temps réel.

#	Description	Niveau
4.4.1	Vérifiez que WebSocket sur TLS (WSS) est utilisé pour toutes les connexions WebSocket.	1
4.4.2	Vérifiez que, lors de la négociation HTTP WebSocket initiale, le champ d'en-tête 'Origin' est vérifié par rapport à une liste d'origines autorisées pour l'application.	2
4.4.3	Vérifiez que, si la gestion de session standard de l'application ne peut pas être utilisée, des jetons dédiés sont utilisés à cet effet, qui sont conformes aux exigences de sécurité de gestion de session pertinentes.	2
4.4.4	Vérifiez que les jetons de gestion de session WebSocket dédiés sont initialement obtenus ou validés via la session HTTPS précédemment authentifiée lors de la transition d'une session HTTPS existante vers un canal WebSocket.	2

Références

Pour plus d'informations, voir également :

- OWASP REST Security Cheat Sheet
- Resources on GraphQL Authorization from graphql.org and Apollo.
- OWASP Web Security Testing Guide: GraphQL Testing
- OWASP Web Security Testing Guide: Testing WebSockets

V5 Gestion des fichiers

Objectif de contrôle

L'utilisation de fichiers peut présenter divers risques pour l'application, notamment le déni de service, l'accès non autorisé et l'épuisement de l'espace de stockage. Ce chapitre présente les exigences pour gérer ces risques.

V5.1 Documentation sur la gestion des fichiers

Cette section comprend l'obligation de documenter les caractéristiques attendues des fichiers acceptés par l'application, comme condition préalable nécessaire au développement et à la vérification des contrôles de sécurité pertinents.

#	Description	Niveau
5.1.1	Vérifiez que la documentation définit les types de fichiers autorisés, les extensions de fichiers attendues et la taille maximale (y compris la taille décompressée) pour chaque fonctionnalité de Téléversement. De plus, assurez-vous que la documentation précise comment les fichiers sont sécurisés pour le téléchargement et le traitement des utilisateurs finaux, par exemple le comportement de l'application lorsqu'un fichier malveillant est détecté.	2

V5.2 Téléversement de fichiers et contenu

La fonctionnalité de téléversement de fichiers est une source majeure de fichiers non fiables. Cette section décrit les exigences permettant de garantir que la présence, le volume ou le contenu de ces fichiers ne nuisent pas à l'application.

#	Description	Niveau
5.2.1	Vérifiez que l'application n'acceptera que des fichiers d'une taille qu'elle peut traiter sans provoquer de perte de performances ou d'attaque par déni de service.	1
5.2.2	Vérifiez que lorsque l'application accepte un fichier, seul ou dans une archive telle qu'un fichier zip, elle vérifie si l'extension de fichier correspond à une extension attendue et que le contenu correspond au type représenté par l'extension. Cela inclut, sans s'y limiter, la vérification des « octets magiques » initiaux, la réécriture d'images et l'utilisation de bibliothèques spécialisées pour la validation du contenu des fichiers. Pour le niveau L1, cela peut se concentrer uniquement sur les fichiers utilisés pour prendre des décisions métier ou de sécurité spécifiques. À partir du niveau L2, cela doit s'appliquer à tous les fichiers acceptés.	1
5.2.3	Vérifiez que l'application vérifie les fichiers compressés (par exemple, zip, gz, docx, odt) par rapport à la taille maximale autorisée non compressée et au nombre maximal de fichiers avant de décompresser le fichier.	2
5.2.4	Vérifiez qu'un quota de taille de fichier et un nombre maximal de fichiers par utilisateur sont appliqués pour garantir qu'un seul utilisateur ne puisse pas remplir le stockage avec trop de fichiers ou des fichiers excessivement volumineux.	3
5.2.5	Vérifiez que l'application n'autorise pas le téléversement de fichiers compressés contenant des liens symboliques, sauf si cela est spécifiquement requis (auquel cas il sera nécessaire	3

#	Description	Niveau
	d'appliquer une liste d'autorisation des fichiers vers lesquels il est possible de créer des liens symboliques).	
5.2.6	Vérifiez que l'application rejette les images téléversées avec une taille de pixel supérieure au maximum autorisé, afin d'éviter les attaques par inondation de pixels.	3

V5.3 Stockage de fichiers

Cette section comprend des exigences visant à empêcher l'exécution inappropriée des fichiers après leur téléversement, à détecter le contenu dangereux et à éviter que des données non fiables ne soient utilisées pour contrôler l'emplacement de stockage des fichiers.

#	Description	Niveau
5.3.1	Vérifiez que les fichiers téléversés ou générés par une entrée non fiable et stockés dans un dossier public ne sont pas exécutés en tant que code de programme côté serveur lorsqu'ils sont accessibles directement avec une requête HTTP.	1
5.3.2	Vérifiez que lorsque l'application crée des chemins d'accès pour les opérations sur les fichiers, elle utilise des données générées en interne ou fiables plutôt que des noms de fichiers soumis par l'utilisateur. Si des noms de fichiers ou des métadonnées de fichiers soumis par l'utilisateur doivent être utilisés, une validation et un nettoyage stricts doivent être appliqués. Ceci permet de se protéger contre les attaques par traversée de chemin, l'inclusion de fichiers locaux ou distants (LFI, RFI) et la falsification de requêtes côté serveur (SSRF).	1
5.3.3	Vérifiez que le traitement des fichiers côté serveur, comme la décompression des fichiers, ignore les informations de chemin fournies par l'utilisateur pour éviter les vulnérabilités telles que le "zip slip".	3

V5.4 Téléchargement de fichier

Cette section décrit les exigences visant à atténuer les risques liés au téléchargement de fichiers, notamment les attaques par traversée de chemin et par injection. Elle s'assure également qu'ils ne contiennent pas de contenu dangereux.

#	Description	Niveau
5.4.1	Vérifiez que l'application valide ou ignore les noms de fichiers soumis par l'utilisateur, y compris dans un paramètre JSON, JSONP ou URL et spécifie un nom de fichier dans le champ d'en-tête Content-Disposition de la réponse.	2
5.4.2	Vérifiez que les noms de fichiers servis (par exemple, dans les champs d'en-tête de réponse HTTP ou les pièces jointes aux e-mails) sont encodés ou nettoyés (par exemple, conformément à la RFC 6266) pour préserver la structure du document et empêcher les attaques par injection.	2
5.4.3	Vérifiez que les fichiers obtenus à partir de sources non fiables sont analysés par des scanners antivirus pour empêcher la diffusion de contenu malveillant connu.	2

Références

Pour plus d'informations, voir également :

- OWASP File Upload Cheat Sheet
- Example of using symlinks for arbitrary file read
- Explanation of "Magic Bytes" from Wikipedia

V6 Authentification

Objectif de contrôle

L'authentification est le processus permettant d'établir ou de confirmer l'authenticité d'une personne ou d'un appareil. Elle consiste à vérifier les déclarations d'une personne ou d'un appareil, à garantir la résistance à l'usurpation d'identité et à empêcher la récupération ou l'interception des mots de passe.

Le NIST SP 800-63 est une norme moderne, fondée sur des preuves, qui est précieuse pour les organisations du monde entier, mais qui est particulièrement pertinente pour les agences américaines et ceux qui interagissent avec les agences américaines.

Bien que de nombreuses exigences de ce chapitre soient basées sur la deuxième section de la norme (appelée NIST SP 800-63B « Directives relatives à l'identité numérique - Authentification et gestion du cycle de vie »), ce chapitre se concentre sur les menaces courantes et les faiblesses d'authentification fréquemment exploitées. Il ne prétend pas couvrir tous les points de la norme de manière exhaustive. Pour les cas où une conformité totale à la norme NIST SP 800-63 est nécessaire, veuillez vous référer à cette dernière.

De plus, la terminologie du NIST SP 800-63 peut parfois différer, et ce chapitre utilise souvent une terminologie plus communément comprise pour améliorer la clarté.

Une fonctionnalité commune aux applications plus avancées est la possibilité d'adapter les étapes d'authentification requises en fonction de divers facteurs de risque. Cette fonctionnalité est abordée dans le chapitre « Autorisation », car ces mécanismes doivent également être pris en compte dans les décisions d'autorisation.

V6.1 Documentation d'authentification

Cette section détaille les exigences relatives à la documentation d'authentification à conserver pour une application. Elle est essentielle à la mise en œuvre et à l'évaluation de la configuration des contrôles d'authentification pertinents.

#	Description	Niveau
6.1.1	Vérifiez que la documentation de l'application définit comment les contrôles, tels que la limitation de débit, l'anti-automatisation et la réponse adaptative, sont utilisés pour se défendre contre les attaques telles que le « credential stuffing » et la force brute des mots de passe. La documentation doit expliquer clairement comment ces contrôles sont configurés et empêcher le blocage malveillant des comptes.	1
6.1.2	Vérifiez qu'une liste de mots spécifiques au contexte est documentée afin d'empêcher leur utilisation dans les mots de passe. Cette liste peut inclure des permutations de noms d'organisations, de produits, d'identifiants de systèmes, de noms de codes de projets, de noms de services ou de rôles, etc.	2
6.1.3	Vérifiez que, si l'application inclut plusieurs voies d'authentification, celles-ci sont toutes documentées avec les contrôles de sécurité et la force d'authentification qui doivent être appliqués de manière cohérente.	2

V6.2 Sécurité des mots de passe

Les mots de passe, appelés « secrets mémorisés » par la norme NIST SP 800-63, comprennent les mots de passe, les phrases de passe, les codes PIN, les schémas de déverrouillage et choisir « le bon chaton » ou d'un autre élément d'image. Ils sont généralement considérés comme « quelque chose que vous connaissez » et sont souvent utilisés comme mécanisme d'authentification à facteur unique.

Cette section contient donc des exigences visant à garantir la création et la gestion sécurisées des mots de passe. La plupart des exigences sont de niveau 1, car elles sont les plus importantes à ce niveau. À partir du niveau 2, des mécanismes d'authentification multifacteur sont requis, les mots de passe pouvant être l'un de ces facteurs.

Les exigences de cette section concernent principalement le chapitre § 5.1.1.2 du Guide du NIST.

#	Description	Niveau
6.2.1	Vérifiez que les mots de passe définis par l'utilisateur comportent au moins 8 caractères, bien qu'un minimum de 15 caractères soit fortement recommandé.	1
6.2.2	Vérifiez que les utilisateurs peuvent modifier leur mot de passe.	1
6.2.3	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.	1
6.2.4	Vérifiez que les mots de passe soumis lors de l'enregistrement du compte ou du changement de mot de passe sont vérifiés par rapport à un ensemble disponible d'au moins 3 000 mots de passe principaux qui correspondent à la politique de mot de passe de l'application, par exemple la longueur minimale.	1
6.2.5	Vérifiez que les mots de passe peuvent être utilisés, quelle que soit leur composition, sans restriction quant au type de caractères autorisés. Aucun nombre minimal de majuscules ou de minuscules, de chiffres ou de caractères spéciaux ne doit être exigé.	1
6.2.6	Vérifiez que les champs de saisie du mot de passe utilisent type=password pour masquer la saisie. Les applications peuvent permettre à l'utilisateur d'afficher temporairement l'intégralité du mot de passe masqué ou le dernier caractère saisi.	1
6.2.7	Vérifiez que la fonctionnalité « coller », les assistants de mot de passe du navigateur et les gestionnaires de mots de passe externes sont autorisés.	1
6.2.8	Vérifiez que l'application vérifie le mot de passe de l'utilisateur exactement tel qu'il a été reçu de l'utilisateur, sans aucune modification telle qu'une troncature ou une transformation de casse.	1
6.2.9	Vérifiez que les mots de passe d'au moins 64 caractères sont autorisés.	2
6.2.10	Vérifiez que le mot de passe d'un utilisateur reste valide jusqu'à ce qu'il soit découvert comme compromis ou qu'il soit renouvelé. L'application ne doit pas exiger de renouvellement périodique des identifiants.	2
6.2.11	Vérifiez que la liste documentée de mots spécifiques au contexte est utilisée pour éviter la création de mots de passe faciles à deviner.	2
6.2.12	Vérifiez que les mots de passe soumis lors de l'enregistrement du compte ou des modifications de mot de passe sont vérifiés par rapport à un ensemble de mots de passe fuités.	2

V6.3 Sécurité générale de l'authentification

Cette section contient les exigences générales relatives à la sécurité des mécanismes d'authentification et définit les différentes attentes en matière de niveaux. Les applications L2 doivent recourir à l'authentification multifacteur (MFA). Les applications L3 doivent utiliser une authentification matérielle, réalisée dans un environnement d'exécution certifié et approuvé (TEE). Cela peut inclure des clés d'accès liées à l'appareil, des authentificateurs eIDAS à niveau d'assurance élevé (LoA), des authentificateurs avec l'assurance NIST Authenticator Assurance Level 3 (AAL3) ou un mécanisme équivalent.

Bien qu'il s'agisse d'une position relativement agressive concernant l'authentification multifacteur, il est essentiel de relever le niveau à ce sujet pour protéger les utilisateurs, et toute tentative d'assouplissement de ces exigences doit être accompagnée d'un plan clair sur la manière dont les risques liés à l'authentification seront atténués, en tenant compte des orientations et des recherches du NIST sur le sujet.

Veuillez noter qu'au moment de la publication, la norme NIST SP 800-63 considère l'email comme non acceptable comme mécanisme d'authentification (copie archivée).

Les exigences de cette section concernent diverses sections du Guide du NIST, incluant : § 4.2.1, § 4.3.1, § 5.2.2, et § 6.1.2.

#	Description	Niveau
6.3.1	Vérifiez que les contrôles visant à empêcher les attaques telles que le bourrage d'informations d'identification et la force brute des mots de passe sont mis en œuvre conformément à la documentation de sécurité de l'application.	1
6.3.2	Vérifiez que les comptes d'utilisateur par défaut (par exemple, « root », « admin » ou « sa ») ne sont pas présents dans l'application ou sont désactivés.	1
6.3.3	Vérifiez que l'application exige que les utilisateurs utilisent un mécanisme d'authentification multifacteur ou une combinaison de mécanismes d'authentification à facteur unique.	2
6.3.4	Vérifiez que, si l'application inclut plusieurs voies d'authentification, il n'existe aucune voie non documentée et que les contrôles de sécurité et la force d'authentification sont appliqués de manière cohérente.	2
6.3.5	Vérifiez que les utilisateurs sont informés des tentatives d'authentification suspectes (réussies ou non). Il peut s'agir de tentatives d'authentification provenant d'un emplacement ou d'un client inhabituel, d'une authentification partiellement réussie (un seul facteur parmi plusieurs), d'une tentative d'authentification après une longue période d'inactivité ou d'une authentification réussie après plusieurs tentatives infructueuses.	3
6.3.6	Vérifiez que le courrier électronique n'est pas utilisé comme mécanisme d'authentification à facteur unique ou à facteurs multiples.	3
6.3.7	Vérifiez que les utilisateurs sont avertis après les mises à jour des détails d'authentification, telles que les réinitialisations d'informations d'identification ou la modification du nom d'utilisateur ou de l'adresse e-mail.	3
6.3.8	Vérifiez que les utilisateurs valides ne peuvent pas être déduits d'échecs d'authentification, par exemple en se basant sur des messages d'erreur, des codes de réponse HTTP ou des temps de réponse différents. Les fonctionnalités d'inscription et de mot de passe oublié doivent également bénéficier de cette protection.	3

V6.4 Cycle de vie et récupération du facteur d'authentification

Les facteurs d'authentification peuvent inclure des mots de passe, des jetons logiciels, des jetons matériels et des dispositifs biométriques. La gestion sécurisée du cycle de vie de ces mécanismes est essentielle à la sécurité d'une application, et cette section présente les exigences afférentes.

Les exigences de cette section concernent principalement les chapitres § 5.1.1.2 ou § 6.1.2.3 of Guide du NIST.

#	Description	Niveau
6.4.1	Vérifiez que les mots de passe initiaux ou les codes d'activation générés par le système sont générés de manière aléatoire et sécurisée, qu'ils respectent la politique de mots de passe en vigueur et qu'ils expirent après une courte période ou après leur première utilisation. Ces secrets initiaux ne doivent pas devenir des mots de passe permanents.	1
6.4.2	Vérifiez que les indices de mot de passe ou l'authentification basée sur les connaissances (appelées « questions secrètes ») ne sont pas présents.	1
6.4.3	Vérifiez qu'un processus sécurisé de réinitialisation d'un mot de passe oublié est mis en œuvre, qui ne contourne aucun mécanisme d'authentification multifacteur activé.	2
6.4.4	Vérifiez que si un facteur d'authentification multifacteur est perdu, la preuve de vérification d'identité est effectuée au même niveau que lors de l'inscription.	2
6.4.5	Vérifiez que les instructions de renouvellement des mécanismes d'authentification qui expirent sont envoyées avec suffisamment de temps pour être exécutées avant l'expiration de l'ancien mécanisme d'authentification, en configurant des rappels automatiques si nécessaire.	3
6.4.6	Vérifiez que les administrateurs peuvent initier la réinitialisation du mot de passe de l'utilisateur, mais que cela ne leur permet pas de modifier ou de choisir son mot de passe. Cela évite qu'ils ne connaissent le mot de passe de l'utilisateur.	3

V6.5 Exigences générales en matière d'authentification multifacteur

Cette section fournit des conseils généraux qui seront pertinents pour différentes méthodes d'authentification multifacteur.

Les mécanismes comprennent :

- Secret recherché dans une table
- Mots de passe à usage unique basés sur le temps (TOTP)
- Mécanismes out-of-band

Les secrets de recherche sont des listes pré-générées de codes secrets, similaires aux numéros d'autorisation de transaction (TAN), aux codes de récupération des réseaux sociaux ou à une grille contenant un ensemble de valeurs aléatoires. Ce type de mécanisme d'authentification est considéré comme « quelque chose que vous possédez », car les codes sont volontairement non mémorisables et doivent donc être stockés quelque part.

Les mots de passe à usage unique basés sur le temps (TOTP) sont des jetons physiques ou logiciels qui affichent un code pseudo-aléatoire à usage unique et changeant en permanence. Ce type de mécanisme d'authentification est considéré comme « quelque chose que vous possédez ». Les TOTP multifactoriels sont similaires aux TOTP à simple facteur", mais nécessitent la saisie d'un code PIN valide, un déverrouillage biométrique, une clé USB ou un appairage NFC, ou d'une valeur supplémentaire (comme des calculateurs de signature de transaction) pour créer le mot de passe à usage unique (OTP) final.

Des détails sur les mécanismes out-of-band seront fournis dans la section suivante.

Les exigences de ces sections concernent principalement les chapitres § 5.1.2, § 5.1.3, § 5.1.4.2, § 5.1.5.2, § 5.2.1, et § 5.2.3 du Guide du NIST.

#	Description	Niveau
6.5.1	Vérifiez que les secrets de recherche, les demandes ou codes d'authentification out-of-band et les mots de passe à usage unique basés sur le temps (TOTP) ne peuvent être utilisés avec succès qu'une seule fois.	2
6.5.2	Vérifiez que, lors de leur stockage dans le backend de l'application, les secrets de recherche comportant moins de 112 bits d'entropie (19 caractères alphanumériques aléatoires ou 34 chiffres aléatoires) sont hachés avec un algorithme de hachage de stockage de mots de passe approuvé, intégrant un sel aléatoire de 32 bits. Une fonction de hachage standard peut être utilisée si le secret comporte 112 bits d'entropie ou plus.	2
6.5.3	Vérifiez que les secrets de recherche, le code d'authentification out-of-band et les mots de passe à usage unique basés sur le temps sont générés à l'aide d'un générateur de nombres pseudo-aléatoires cryptographiquement sécurisé (CSPRNG) pour éviter les valeurs prévisibles.	2
6.5.4	Vérifiez que les secrets de recherche et les codes d'authentification out-of-band ont un minimum de 20 bits d'entropie (généralement 4 caractères alphanumériques aléatoires ou 6 chiffres aléatoires suffisent).	2
6.5.5	Vérifiez que les requêtes, codes ou jetons d'authentification out-of-band, ainsi que les mots de passe à usage unique (TOTP), ont une durée de vie définie. Les requêtes out-of-band doivent avoir une durée de vie maximale de 10 minutes et les TOTP, de 30 secondes.	2
6.5.6	Vérifiez que tout facteur d'authentification (y compris les appareils physiques) peut être révoqué en cas de vol ou autre perte.	3
6.5.7	Vérifiez que les mécanismes d'authentification biométrique ne sont utilisés que comme facteurs secondaires, avec quelque chose que vous possédez ou quelque chose que vous savez.	3
6.5.8	Vérifiez que les mots de passe à usage unique basés sur le temps (TOTP) sont vérifiés en fonction d'une source temporelle provenant d'un service de confiance et non d'une heure non fiable ou fournie par le client.	3

V6.6 Mécanismes d'authentification hors bande

Cela implique généralement que le serveur d'authentification communique avec un appareil physique via un canal secondaire sécurisé. Par exemple, l'envoi de notifications push aux appareils mobiles. Ce type de mécanisme d'authentification est considéré comme « quelque chose que vous possédez ».

Les mécanismes d'authentification out-of-band non sécurisés tels que le courrier électronique et la VoIP ne sont pas autorisés. L'authentification PSTN et SMS est actuellement considérée comme des mécanismes d'authentification « restreints » par le NIST et devrait être déconseillée au profit des mots de passe à usage unique basés sur le temps (TOTP), d'un mécanisme cryptographique ou similaire. La norme NIST SP 800-63B § 5.1.3.3 recommande de traiter les risques d'échange d'appareil, de changement de carte SIM, de portage de numéro ou d'autres comportements anormaux, si l'authentification out-of-band par téléphone ou SMS doit absolument être prise en charge. Bien que cette section ASVS n'impose pas cela comme une exigence, le fait de ne pas prendre ces précautions pour une application L2 sensible ou une application L3 doit être considéré comme un signal d'alarme important.

Notez que le NIST a également récemment publié des directives déconseillant l'utilisation des notifications push. Bien que cette section ASVS ne le fasse pas, il est important d'être conscient des risques de « push bombing ».

#	Description	Niveau
6.6.1	Vérifier que les mécanismes d'authentification utilisant le réseau téléphonique public commuté (RTPC) pour la transmission de mots de passe à usage unique (OTP) par téléphone ou SMS ne sont proposés que lorsque le numéro de téléphone a été préalablement validé. D'autres méthodes plus robustes (telles que les mots de passe à usage unique à durée déterminée) sont également proposées, et que le service informe les utilisateurs des risques de sécurité qu'elles présentent. Pour les applications de niveau 3, le téléphone et les SMS ne doivent pas être disponibles.	2
6.6.2	Vérifiez que les demandes d'authentification out-of-band, les codes ou les jetons sont liés à la demande d'authentification d'origine pour laquelle ils ont été générés et ne sont pas utilisables pour une demande précédente ou ultérieure.	2
6.6.3	Vérifiez qu'un mécanisme d'authentification out-of-band basé sur du code est protégé contre les attaques par force brute grâce au "rate limiting". Envisagez également d'utiliser un code avec au moins 64 bits d'entropie.	2
6.6.4	Vérifiez que, lorsque les notifications push sont utilisées pour l'authentification multifacteur, le "rate limiting" est appliqué afin d'empêcher les attaques de type « push bombing ». La concordance des numéros peut également atténuer ce risque.	3

V6.7 Mécanisme d'authentification cryptographique

Les mécanismes d'authentification cryptographique incluent les cartes à puce ou les clés FIDO. L'utilisateur doit connecter ou appairer le dispositif cryptographique à l'ordinateur pour finaliser l'authentification. Le serveur d'authentification envoie un nonce de défi au dispositif ou au logiciel cryptographique, qui calcule une réponse à partir d'une clé cryptographique stockée de manière sécurisée. Les exigences de cette section fournissent des conseils spécifiques à la mise en œuvre de ces mécanismes, les conseils sur les algorithmes cryptographiques étant traités dans le chapitre « Cryptographie ».

Lorsque des clés partagées ou secrètes sont utilisées pour l'authentification cryptographique, elles doivent être stockées à l'aide des mêmes mécanismes que les autres secrets système, comme documenté dans la section « Gestion des secrets » du chapitre « Configuration ».

Les exigences de cette section concernent principalement le chapitre § 5.1.7.2 du Guide du NIST.

#	Description	Niveau
6.7.1	Vérifiez que les certificats utilisés pour vérifier les assertions d'authentification cryptographique sont stockés de manière à les protéger contre toute modification.	3
6.7.2	Vérifiez que le nonce de défi a une longueur d'au moins 64 bits et qu'il est statistiquement unique ou unique sur toute la durée de vie du périphérique cryptographique.	3

V6.8 Authentification avec un fournisseur d'identité

Les fournisseurs d'identité (IdP) fournissent une identité fédérée aux utilisateurs. Ces derniers possèdent souvent plusieurs identités auprès de plusieurs IdP, comme une identité d'entreprise utilisant Azure AD, Okta, Ping Identity ou Google, ou une identité grand public utilisant Facebook, Twitter, Google ou WeChat, pour ne citer que quelques alternatives courantes. Cette liste ne constitue pas une recommandation pour ces entreprises ou services, mais simplement un encouragement aux développeurs à prendre en compte le fait que de nombreux utilisateurs possèdent de nombreuses identités établies. Les organisations devraient envisager l'intégration avec les identités utilisateur existantes, en fonction du profil de risque lié à la fiabilité de la vérification d'identité de l'IdP. Par exemple, il est peu probable qu'une organisation gouvernementale accepte

une identité de réseau social comme identifiant pour des systèmes sensibles, car il est facile de créer de fausses identités ou des identités jetables, tandis qu'une entreprise de jeux mobiles pourrait avoir besoin de s'intégrer aux principales plateformes de réseaux sociaux pour développer sa base de joueurs actifs.

L'utilisation sécurisée de fournisseurs d'identité externes nécessite une configuration et une vérification rigoureuses afin d'éviter l'usurpation d'identité ou la falsification d'assertions. Cette section décrit les exigences pour gérer ces risques.

#	Description	Niveau
6.8.1	Vérifiez que, si l'application prend en charge plusieurs fournisseurs d'identité (IdP), l'identité de l'utilisateur ne peut pas être usurpée via un autre fournisseur d'identité pris en charge (par exemple, en utilisant le même identifiant utilisateur). La mesure standard consisterait à ce que l'application enregistre et identifie l'utilisateur à l'aide d'une combinaison de l'ID du fournisseur d'identité (servant d'espace de noms) et de l'ID de l'utilisateur dans le fournisseur d'identité.	2
6.8.2	Vérifiez que la présence et l'intégrité des signatures numériques sur les assertions d'authentification (par exemple sur les assertions JWT ou SAML) sont toujours validées, en rejetant toutes les assertions non signées ou ayant des signatures non valides.	2
6.8.3	Vérifiez que les assertions SAML sont traitées de manière unique et utilisées une seule fois au cours de la période de validité pour éviter les attaques par rejeu.	2
6.8.4	Si une application utilise un fournisseur d'identité (IdP) distinct et attend une force, des méthodes ou une date d'authentification spécifiques pour des fonctions spécifiques, vérifiez que l'application les vérifie à l'aide des informations renvoyées par l'IdP. Par exemple, si OIDC est utilisé, cela peut être réalisé en validant les affirmations de jeton d'identification telles que « acr », « amr » et « auth_time » (le cas échéant). Si l'IdP ne fournit pas ces informations, l'application doit disposer d'une approche de secours documentée qui suppose que le mécanisme d'authentification de force minimale a été utilisé (par exemple, une authentification à facteur unique avec nom d'utilisateur et mot de passe).	2

Références

Pour plus d'informations, voir également :

- NIST SP 800-63 - Digital Identity Guidelines
- NIST SP 800-63B - Authentication and Lifecycle Management
- NIST SP 800-63 FAQ
- OWASP Web Security Testing Guide: Testing for Authentication
- OWASP Password Storage Cheat Sheet
- OWASP Forgot Password Cheat Sheet
- OWASP Choosing and Using Security Questions Cheat Sheet
- CISA Guidance on "Number Matching"
- Details on the FIDO Alliance

V7 Gestion des sessions

Objectif de contrôle

Les mécanismes de gestion de session permettent aux applications de corréler les interactions entre utilisateurs et appareils au fil du temps, même avec des protocoles de communication sans état (comme HTTP). Les applications modernes peuvent utiliser plusieurs jetons de session aux caractéristiques et objectifs distincts. Un système de gestion de session sécurisé empêche les attaquants d'obtenir, d'utiliser ou d'abuser de la session d'une victime. Les applications gérant des sessions doivent garantir le respect des exigences de gestion de session de haut niveau suivantes :

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

De nombreuses exigences de ce chapitre concernent des contrôles sélectionnés de NIST SP 800-63 Digital Identity Guidelines, en se concentrant sur les menaces courantes et les faiblesses d'authentification couramment exploitées.

Notez que les exigences relatives aux détails d'implémentation spécifiques de certains mécanismes de gestion de session peuvent être trouvées ailleurs :

- Les cookies HTTP sont un mécanisme courant de sécurisation des jetons de session. Les exigences de sécurité spécifiques aux cookies sont décrites dans le chapitre « Sécurité de l'interface web ».
- Les jetons autonomes sont fréquemment utilisés pour maintenir les sessions. Les exigences de sécurité spécifiques sont décrites dans le chapitre « Jetons autonomes ».

V7.1 Documentation sur la gestion des sessions

Il n'existe pas de modèle unique convenant à toutes les applications. Il est donc impossible de définir des limites universelles et adaptées à tous les cas. Une analyse des risques, accompagnée de décisions de sécurité documentées relatives à la gestion des sessions, doit être réalisée avant la mise en œuvre et les tests. Cela garantit que le système de gestion des sessions est adapté aux exigences spécifiques de l'application.

Que le mécanisme de session choisi soit avec ou sans état, l'analyse doit être complète et documentée afin de démontrer que la solution sélectionnée est capable de satisfaire à toutes les exigences de sécurité pertinentes. L'interaction avec les mécanismes d'authentification unique (SSO) utilisés doit également être prise en compte.

#	Description	Niveau
7.1.1	Vérifiez que le délai d'inactivité de la session de l'utilisateur et la durée de vie maximale absolue de la session sont documentés, sont appropriés en combinaison avec d'autres contrôles et que la documentation inclut une justification de tout écart par rapport aux exigences de réauthentification NIST SP 800-63B.	2
7.1.2	Vérifiez que la documentation définit le nombre de sessions simultanées (parallèles) autorisées pour un compte ainsi que les comportements et actions prévus à entreprendre lorsque le nombre maximal de sessions actives est atteint.	2
7.1.3	Vérifiez que tous les systèmes qui créent et gèrent des sessions utilisateur dans le cadre d'un écosystème de gestion des identités fédérées (tels que les systèmes SSO) sont documentés avec des contrôles pour coordonner la durée de vie des sessions, la résiliation et toute autre condition nécessitant une réauthentification.	2

V7.2 Sécurité fondamentale de la gestion des sessions

Cette section satisfait aux exigences essentielles des sessions sécurisées en vérifiant que les jetons de session sont générés et validés de manière sécurisée.

#	Description	Niveau
7.2.1	Vérifiez que l'application effectue toutes les vérifications des jetons de session à l'aide d'un service backend de confiance.	1
7.2.2	Vérifiez que l'application utilise des jetons autonomes ou de référence générés dynamiquement pour la gestion des sessions, c'est-à-dire sans utiliser de secrets et de clés d'API statiques.	1
7.2.3	Vérifiez que si des jetons par référence sont utilisés pour représenter les sessions utilisateur, ils sont uniques et générés à l'aide d'un générateur de nombres pseudo-aléatoires cryptographiquement sécurisé (CSPRNG) et possèdent au moins 128 bits d'entropie.	1
7.2.4	Vérifiez que l'application génère un nouveau jeton de session lors de l'authentification de l'utilisateur, y compris la réauthentification, et met fin au jeton de session actuel.	1

V7.3 Délai d'expiration de la session

Les mécanismes de temporisation de session servent à minimiser les risques de détournement de session et autres formes d'abus. Les temporisations doivent respecter des décisions de sécurité documentées.

#	Description	Niveau
7.3.1	Vérifiez qu'il existe un délai d'inactivité tel que la réauthentification soit appliquée conformément à l'analyse des risques et aux décisions de sécurité documentées.	2
7.3.2	Vérifiez qu'il existe une durée de vie maximale absolue de la session de sorte que la réauthentification soit appliquée conformément à l'analyse des risques et aux décisions de sécurité documentées.	2

V7.4 Fin de session

La fin de session peut être gérée soit par l'application elle-même, soit par le fournisseur d'authentification unique (SSO) si ce dernier gère la session à sa place. Il peut être nécessaire de déterminer si le fournisseur d'authentification unique est concerné par les exigences de cette section, car certaines peuvent être contrôlées par lui.

La fin de session doit entraîner une réauthentification et être effective dans l'ensemble de l'application, de la connexion fédérée (si présente) et de toutes les parties utilisatrices.

Pour les mécanismes de session avec état, la terminaison implique généralement l'invalidation de la session sur le serveur principal. Dans le cas de jetons autonomes, des mesures supplémentaires sont nécessaires pour révoquer ou bloquer ces jetons, car ils pourraient sinon rester valides jusqu'à leur expiration.

#	Description	Niveau
7.4.1	Vérifiez que lorsque la fin de session est déclenchée (par exemple, déconnexion ou expiration), l'application interdit toute utilisation ultérieure de la session. Pour les jetons	1

#	Description	Niveau
	par référence ou les sessions avec état, cela implique l'invalidation des données de session au niveau du backend de l'application. Les applications utilisant des jetons autonomes nécessiteront une solution, comme la gestion d'une liste des jetons terminés, l'interdiction des jetons produits avant une date et une heure spécifique à chaque utilisateur ou la rotation d'une clé de signature spécifique à chaque utilisateur.	
7.4.2	Vérifiez que l'application met fin à toutes les sessions actives lorsqu'un compte utilisateur est désactivé ou supprimé (par exemple, lorsqu'un employé quitte l'entreprise).	1
7.4.3	Vérifiez que l'application offre la possibilité de mettre fin à toutes les autres sessions actives après une modification ou une suppression réussie de tout facteur d'authentification (y compris la modification du mot de passe via une réinitialisation ou une récupération et, le cas échéant, une mise à jour des paramètres MFA).	2
7.4.4	Vérifiez que toutes les pages nécessitant une authentification disposent d'un accès facile et visible à la fonctionnalité de déconnexion.	2
7.4.5	Vérifiez que les administrateurs d'applications sont en mesure de mettre fin aux sessions actives pour un utilisateur individuel ou pour tous les utilisateurs.	2

V7.5 Défenses contre les abus de session

Cette section décrit les exigences visant à atténuer le risque posé par les sessions actives détournées ou utilisées abusivement par des vecteurs s'appuyant sur l'existence et les capacités des sessions utilisateur actives. Par exemple, l'exécution de contenu malveillant pour forcer un navigateur authentifié à effectuer une action en utilisant la session de la victime.

Veuillez noter que les instructions spécifiques au niveau dans le chapitre « Authentification » doivent être prises en compte lors de l'examen des exigences de cette section.

#	Description	Niveau
7.5.1	Vérifiez que l'application nécessite une réauthentification complète avant d'autoriser les modifications des attributs de compte sensibles qui peuvent affecter l'authentification, tels que l'adresse e-mail, le numéro de téléphone, la configuration MFA ou d'autres informations utilisées dans la récupération de compte.	2
7.5.2	Vérifiez que les utilisateurs peuvent afficher et (après s'être authentifiés à nouveau avec au moins un facteur) terminer une ou toutes les sessions actuellement actives.	2
7.5.3	Vérifiez que l'application nécessite une authentification supplémentaire avec au moins un facteur ou une vérification secondaire avant d'effectuer des transactions ou des opérations hautement sensibles.	3

V7.6 Réauthentification fédérée

Cette section s'adresse aux développeurs de code de partie utilisatrice (RP) ou de fournisseur d'identité (IdP). Ces exigences découlent de la norme NIST SP 800-63C relative à la fédération et aux assertions.

#	Description	Niveau
7.6.1	Vérifiez que la durée de vie et la fin de la session entre les parties de confiance (RP) et les fournisseurs d'identité (IdP) se comportent comme documenté, en exigeant une réauthentification si nécessaire, par exemple lorsque le temps maximal entre les événements d'authentification IdP est atteint.	2
7.6.2	Vérifiez que la création d'une session nécessite soit le consentement de l'utilisateur, soit une action explicite, empêchant ainsi la création de nouvelles sessions d'application sans interaction de l'utilisateur.	2

Références

Pour plus d'informations, voir également :

- OWASP Web Security Testing Guide: Session Management Testing
- OWASP Session Management Cheat Sheet

V8 Autorisation

Objectif de contrôle

L'autorisation garantit que l'accès est accordé uniquement aux utilisateurs autorisés (utilisateurs, serveurs et autres clients). Pour appliquer le principe du moindre privilège (POLP), les applications vérifiées doivent répondre aux exigences générales suivantes :

- Documenter les règles d'autorisation des documents, y compris les facteurs de prise de décision et les contextes environnementaux.
- Les consommateurs ne devraient avoir accès qu'aux ressources autorisées par leurs droits définis.

V8.1 Documentation d'autorisation

Une documentation exhaustive des autorisations est essentielle pour garantir que les décisions de sécurité sont appliquées de manière cohérente, vérifiables et conformes aux politiques de l'organisation. Cela réduit le risque d'accès non autorisé en clarifiant les exigences de sécurité et en les rendant applicables aux développeurs, administrateurs et testeurs.

#	Description	Niveau
8.1.1	Vérifiez que la documentation d'autorisation définit des règles de restriction de l'accès au niveau de la fonction et aux données en fonction des autorisations du consommateur et des attributs des ressources.	1
8.1.2	Vérifiez que la documentation d'autorisation définit les règles de restriction d'accès au niveau des champs (lecture et écriture) en fonction des autorisations des consommateurs et des attributs des ressources. Notez que ces règles peuvent dépendre d'autres valeurs d'attributs de l'objet de données concerné, telles que l'état ou le statut.	2
8.1.3	Vérifiez que la documentation de l'application définit les attributs environnementaux et contextuels (y compris, mais sans s'y limiter, l'heure de la journée, l'emplacement de l'utilisateur, l'adresse IP ou l'appareil) qui sont utilisés dans l'application pour prendre des décisions de sécurité, y compris celles relatives à l'authentification et à l'autorisation.	3
8.1.4	Vérifier que la documentation d'authentification et d'autorisation définit l'utilisation des facteurs environnementaux et contextuels dans la prise de décision, en plus des autorisations au niveau des fonctions, des données et des champs. Cela doit inclure les attributs évalués, les seuils de risque et les actions entreprises (par exemple, autorisation, contestation, refus, renforcement de l'authentification).	3

V8.2 Conception d'autorisation générale

La mise en œuvre de contrôles d'autorisation granulaires aux niveaux de la fonction, des données et des champs garantis que les consommateurs ne pourront accéder qu'à ce qui leur a été explicitement accordé.

#	Description	Niveau
8.2.1	Vérifiez que l'application garantit que l'accès basé sur la fonction est limité aux consommateurs disposant d'autorisations explicites.	1
8.2.2	Vérifiez que l'application garantit que l'accès spécifique aux données est limité aux consommateurs disposant d'autorisations explicites sur des éléments de données	1

#	Description	Niveau
	spécifiques afin d'atténuer les références d'objet directes non sécurisées (IDOR) et les autorisations de niveau d'objet rompues (BOLA).	
8.2.3	Vérifiez que l'application garantit que l'accès au niveau du champ est limité aux consommateurs disposant d'autorisations explicites sur des champs spécifiques afin d'atténuer les problèmes d'autorisation au niveau de la propriété d'objet (BOPLA).	2
8.2.4	Vérifiez que les contrôles de sécurité adaptatifs basés sur les attributs environnementaux et contextuels du consommateur (tels que l'heure, la localisation, l'adresse IP ou l'appareil) sont mis en œuvre pour les décisions d'authentification et d'autorisation, comme défini dans la documentation de l'application. Ces contrôles doivent être appliqués lorsque le consommateur tente de démarrer une nouvelle session et également pendant une session existante.	3

V8.3 Autorisation par opération

L'application immédiate des modifications d'autorisation dans le niveau approprié de l'architecture d'une application est essentielle pour empêcher les actions non autorisées, en particulier dans les environnements dynamiques.

#	Description	Niveau
8.3.1	Vérifier que l'application applique les règles d'autorisations au niveau de la couche service de confiance et ne s'appuie pas sur des contrôles qu'un consommateur non approuvé pourrait manipuler, comme JavaScript côté client.	1
8.3.2	Vérifiez que les modifications apportées aux valeurs sur lesquelles reposent les décisions d'autorisation sont appliquées immédiatement. Lorsque les modifications ne peuvent pas être appliquées immédiatement (par exemple, en s'appuyant sur des données de jetons autonomes), des contrôles d'atténuation doivent être mis en place pour alerter un consommateur lorsqu'il effectue une action alors qu'il n'est plus autorisé à le faire et annuler la modification. Notez que cette solution alternative ne limiterait pas les fuites d'informations.	3
8.3.3	Vérifiez que l'accès à un objet repose sur les autorisations du sujet d'origine (par exemple, le consommateur), et non sur celles d'un intermédiaire ou d'un service agissant en son nom. Par exemple, si un consommateur appelle un service web à l'aide d'un jeton d'authentification autonome, et que ce service demande ensuite des données à un autre service, ce dernier utilisera le jeton du consommateur, plutôt qu'un jeton inter-machine du premier service, pour prendre les décisions d'autorisation.	3

V8.4 Autres considérations relatives à l'autorisation

Des considérations supplémentaires concernant l'autorisation, en particulier pour les interfaces administratives et les environnements multi-tenants, aident à empêcher tout accès non autorisé.

#	Description	Niveau

8.4.1	Vérifiez que les applications multi-tenants utilisent des contrôles inter-tenants pour garantir que les opérations des consommateurs n'affecteront jamais les tenants avec lesquels ils ne sont pas autorisés à interagir.	2
8.4.2	Vérifiez que l'accès aux interfaces administratives intègre plusieurs couches de sécurité, notamment la vérification continue de l'identité du consommateur, l'évaluation de la posture de sécurité des appareils et l'analyse contextuelle des risques, garantissant que l'emplacement du réseau ou les endpoints approuvés ne sont pas les seuls facteurs d'autorisation, même s'ils peuvent réduire la probabilité d'accès non autorisé.	3

Références

Pour plus d'informations, voir également :

- OWASP Web Security Testing Guide: Authorization
- OWASP Authorization Cheat Sheet

V9 Jetons autonomes

Objectif de contrôle

Le concept de jeton autonome est mentionné dans la RFC 6749 OAuth 2.0 originale de 2012. Il désigne un jeton contenant des données ou des revendications sur lesquelles un service récepteur s'appuie pour prendre des décisions de sécurité. Il convient de le distinguer d'un simple jeton contenant uniquement un identifiant, qu'un service récepteur utilise pour rechercher des données localement. Les exemples les plus courants de jetons autonomes sont les jetons Web JSON (JWT) et les assertions SAML.

L'utilisation de jetons autonomes est devenue très répandue, même en dehors d'OAuth et d'OIDC. Parallèlement, la sécurité de ce mécanisme repose sur la capacité à valider l'intégrité du jeton et à garantir sa validité dans un contexte particulier. Ce processus présente de nombreux pièges, et ce chapitre détaille les mécanismes que les applications devraient mettre en place pour les éviter.

V9.1 Source et intégrité du jeton

Cette section comprend des exigences visant à garantir que le jeton a été produit par une partie de confiance et n'a pas été falsifié.

#	Description	Niveau
9.1.1	Vérifiez que les jetons autonomes sont validés à l'aide de leur signature numérique ou MAC pour les protéger contre toute falsification avant d'accepter le contenu du jeton.	1
9.1.2	Vérifiez que seuls les algorithmes d'une liste blanche peuvent être utilisés pour créer et vérifier des jetons autonomes, dans un contexte donné. La liste blanche doit inclure les algorithmes autorisés, idéalement symétriques ou asymétriques, et ne doit pas inclure l'algorithme « Aucun ». Si les algorithmes symétriques et asymétriques doivent être pris en charge, des contrôles supplémentaires seront nécessaires pour éviter toute confusion de clés.	1
9.1.3	Vérifiez que les clés utilisées pour valider les jetons autonomes proviennent de sources préconfigurées et fiables pour l'émetteur du jeton, empêchant ainsi les attaquants de spécifier des sources et des clés non fiables. Pour les JWT et autres structures JWS, les en-têtes tels que « jku », « x5u » et « jwk » doivent être validés par rapport à une liste blanche de sources fiables.	1

V9.2 Contenu du jeton

Avant de prendre des décisions de sécurité basées sur le contenu d'un jeton autonome, il est nécessaire de vérifier que le jeton a été présenté pendant sa période de validité et qu'il est destiné à être utilisé par le service destinataire et pour l'usage pour lequel il a été présenté. Cela permet d'éviter toute utilisation croisée non sécurisée entre différents services ou avec différents types de jetons provenant du même émetteur.

Les exigences spécifiques pour OAuth et OIDC sont couvertes dans le chapitre dédié.

#	Description	Niveau
9.2.1	Vérifiez que, si une période de validité est indiquée dans les données du jeton, celui-ci et son contenu ne sont acceptés que si la date de vérification est comprise dans cette période. Par exemple, pour les JWT, les revendications « nbf » et « exp » doivent être vérifiées.	1

#	Description	Niveau
9.2.2	Vérifiez que le service recevant un jeton valide le type de jeton et son utilisation avant d'en accepter le contenu. Par exemple, seuls les jetons d'accès peuvent être acceptés pour les décisions d'autorisation, et seuls les jetons d'identification peuvent être utilisés pour prouver l'authentification des utilisateurs.	2
9.2.3	Vérifiez que le service accepte uniquement les jetons destinés à être utilisés avec ce service (audience). Pour les JWT, cela peut être réalisé en validant la revendication « aud » par rapport à une liste blanche définie dans le service.	2
9.2.4	Si un émetteur de jetons utilise la même clé privée pour émettre des jetons destinés à différentes audiences, vérifiez que les jetons émis contiennent une restriction d'audience identifiant de manière unique les audiences visées. Cela empêchera la réutilisation d'un jeton avec une audience non prévue. Si l'identifiant d'audience est provisionné dynamiquement, l'émetteur de jetons doit valider ces audiences afin de garantir qu'elles n'entraînent pas d'usurpation d'identité.	2

Références

Pour plus d'informations, voir également :

- OWASP JSON Web Token Cheat Sheet for Java Cheat Sheet (mais il contient des orientations générales utiles)

V10 Oauth et OIDC

Objectif de contrôle

OAuth2 (appelé OAuth dans ce chapitre) est un Framework standard pour l'autorisation déléguée. Par exemple, grâce à OAuth, une application cliente peut accéder aux API (ressources serveur) au nom d'un utilisateur, à condition que ce dernier l'ait autorisée.

OAuth n'est pas conçu pour l'authentification dOauth et OIDCes utilisateurs. Le Framework OpenID Connect (OIDC) étend OAuth en ajoutant une couche d'identité utilisateur. OIDC prend en charge des fonctionnalités telles que la standardisation des informations utilisateur, l'authentification unique (SSO) et la gestion des sessions. OIDC étant une extension d'OAuth, les exigences OAuth décrites dans ce chapitre s'appliquent également à OIDC.

Les rôles suivants sont définis dans OAuth :

- Le client OAuth est l'application qui tente d'accéder aux ressources du serveur (par exemple, en appelant une API à l'aide du jeton d'accès émis). Il s'agit souvent d'une application côté serveur.
  - Un client confidentiel est un client capable de maintenir la confidentialité des informations d'identification qu'il utilise pour s'authentifier auprès du serveur d'autorisation.
  - Un client public n'est pas en mesure de préserver la confidentialité des informations d'identification nécessaires à l'authentification auprès du serveur d'autorisation. Par conséquent, au lieu de s'authentifier (par exemple, à l'aide des paramètres « client_id » et « client_secret »), il s'identifie uniquement (à l'aide du paramètre « client_id »).
- Le serveur de ressources OAuth (RS) est l'API du serveur qui expose les ressources aux clients OAuth.
- Le serveur d'autorisation OAuth (AS) est une application serveur qui émet des jetons d'accès aux clients OAuth. Ces jetons permettent aux clients OAuth d'accéder aux ressources RS, soit pour le compte d'un utilisateur final, soit pour leur propre compte. L'AS est souvent une application distincte, mais (le cas échéant) il peut être intégré à un RS approprié.
- Le propriétaire de la ressource (RO) est l'utilisateur final qui autorise les clients OAuth à obtenir en son nom un accès limité aux ressources hébergées sur le serveur de ressources. Le propriétaire de la ressource consent à cette autorisation déléguée en interagissant avec le serveur d'autorisation.

Les rôles suivants sont définis dans OIDC :

- La partie de confiance (RP) est l'application cliente qui demande l'authentification de l'utilisateur final via le fournisseur OpenID. Elle joue le rôle de client OAuth.
- Le fournisseur OpenID (OP) est un AS OAuth capable d'authentifier l'utilisateur final et de fournir des revendications OIDC à un RP. L'OP peut être le fournisseur d'identité (IdP), mais dans les scénarios fédérés, l'OP et le fournisseur d'identité (où l'utilisateur final s'authentifie) peuvent être des applications serveur différentes.

OAuth et OIDC ont été initialement conçus pour les applications tierces. Aujourd'hui, ils sont également souvent utilisés par les applications propriétaires. Cependant, lorsqu'ils sont utilisés dans des scénarios propriétaires, comme l'authentification et la gestion de session, le protocole ajoute une certaine complexité, ce qui peut engendrer de nouveaux défis de sécurité.

OAuth et OIDC peuvent être utilisés pour de nombreux types d'applications, mais l'accent d'ASVS et les exigences de ce chapitre portent sur les applications Web et les API.

Étant donné que OAuth et OIDC peuvent être considérés comme une logique au-dessus des technologies Web, les exigences générales des autres chapitres s'appliquent toujours et ce chapitre ne peut pas être sorti de son contexte.

Ce chapitre présente les meilleures pratiques actuelles pour OAuth2 et OIDC, conformément aux spécifications disponibles sur https://oauth.net/2/ et https://openid.net/developers/specs/. Même si les RFC sont considérées comme matures, elles sont fréquemment mises à jour. Il est donc important de se conformer aux dernières versions lors de l'application des exigences de ce chapitre. Consultez la section Références pour plus de détails.

Compte tenu de la complexité du domaine, il est essentiel qu'une solution OAuth ou OIDC sécurisée utilise des serveurs d'autorisation standard bien connus du secteur et applique la configuration de sécurité recommandée.

La terminologie utilisée dans ce chapitre est conforme aux RFC OAuth et aux spécifications OIDC ; mais notez que la terminologie OIDC n'est utilisée que pour les exigences spécifiques à OIDC, sinon, la terminologie OAuth est utilisée.

Dans le contexte d'OAuth et d'OIDC, le terme « jeton » dans ce chapitre fait référence à :

- Les jetons d'accès, qui ne peuvent être consommés que par le serveur de sécurité (RS) et qui peuvent être des jetons de référence validés par introspection ou des jetons autonomes validés à l'aide d'un matériau clé.
- Jetons de rafraîchissement, qui ne seront consommés que par le serveur d'autorisation qui a émis le jeton.
- Jetons d'identification OIDC, qui ne doivent être consommés que par le client qui a déclenché le flux d'autorisation.

Les niveaux de risque pour certaines exigences de ce chapitre varient selon que le client est confidentiel ou public. L'utilisation d'une authentification client forte limitant de nombreux vecteurs d'attaque, certaines exigences peuvent être assouplies lors de l'utilisation d'un client confidentiel pour les applications L1.

V10.1 Sécurité générique OAuth et OIDC

Cette section couvre les exigences architecturales génériques qui s'appliquent à toutes les applications utilisant OAuth ou OIDC.

#	Description	Niveau
10.1.1	Vérifiez que les jetons sont envoyés uniquement aux composants qui en ont absolument besoin. Par exemple, lors de l'utilisation d'un modèle backend-for-frontend pour les applications JavaScript basées sur un navigateur, les jetons d'accès et d'actualisation ne doivent être accessibles qu'au backend.	2
10.1.2	Vérifiez que le client n'accepte les valeurs du serveur d'autorisation (telles que le code d'autorisation ou le jeton d'identification) que si elles résultent d'un flux d'autorisation initié par la même session d'agent utilisateur et la même transaction. Cela nécessite que les secrets générés par le client, tels que la clé de preuve pour l'échange de codes (PKCE) « code_verifier », « state » ou le nonce OIDC, ne puissent être devinés, soient spécifiques à la transaction et soient liés de manière sécurisée au client et à la session d'agent utilisateur au cours de laquelle la transaction a été initiée.	2

V10.2 Client OAuth

Ces exigences détaillent les responsabilités des applications clientes OAuth. Le client peut être, par exemple, un serveur web backend (souvent utilisé comme backend pour frontend, BFF), une intégration de service backend ou une application monopage frontend (SPA, également appelée application basée sur un navigateur).

En général, les clients backend sont considérés comme confidentiels et les clients frontend comme publics. Cependant, les applications natives exécutées sur l'appareil de l'utilisateur final peuvent être considérées comme confidentielles lorsqu'elles s'enregistrent dynamiquement des clients OAuth.

#	Description	Niveau
10.2.1	Vérifiez que, si le flux de code est utilisé, le client OAuth dispose d'une protection contre les attaques de falsification de requêtes basées sur le navigateur, communément appelées falsification de requêtes intersites (CSRF), qui déclenchent des demandes de jetons, soit en utilisant la fonctionnalité de clé de preuve pour l'échange de code (PKCE), soit en vérifiant le paramètre « état » envoyé dans la demande d'autorisation.	2
10.2.2	Vérifiez que, si le client OAuth peut interagir avec plusieurs serveurs d'autorisation, il dispose d'une protection contre les attaques par confusion. Par exemple, il peut exiger que le serveur d'autorisation renvoie la valeur du paramètre « iss » et la valide dans la réponse d'autorisation et la réponse du jeton.	2
10.2.3	Vérifiez que le client OAuth demande uniquement les étendues requises (ou d'autres paramètres d'autorisation) dans les requêtes adressées au serveur d'autorisation.	3

V10.3 Serveur de ressources OAuth

Dans le contexte d'ASVS et de ce chapitre, le serveur de ressources est une API. Pour fournir un accès sécurisé, le serveur de ressources doit :

- Valider le jeton d'accès, conformément au format du jeton et aux spécifications du protocole, par exemple, validation JWT ou introspection du jeton OAuth.
- S'il est valide, appliquer les décisions d'autorisation en fonction des informations du jeton d'accès et des autorisations accordées. Par exemple, le serveur de ressources doit vérifier que le client (agissant au nom du RO) est autorisé à accéder à la ressource demandée.

Par conséquent, les exigences répertoriées ici sont spécifiques à OAuth ou OIDC et doivent être exécutées après la validation du jeton et avant d'effectuer l'autorisation basée sur les informations du jeton.

#	Description	Niveau
10.3.1	Vérifiez que le serveur de ressources accepte uniquement les jetons d'accès destinés à être utilisés avec ce service (audience). L'audience peut être incluse dans un jeton d'accès structuré (comme la revendication « aud » dans JWT) ou vérifiée à l'aide du point de terminaison d'introspection de jeton.	2
10.3.2	Vérifiez que le serveur de ressources applique les décisions d'autorisation en fonction des revendications du jeton d'accès définissant l'autorisation déléguée. Si des revendications telles que « sub », « scope » et « authorization_details » sont présentes, elles doivent être prises en compte dans la décision.	2

#	Description	Niveau
10.3.3	Vérifiez que si une décision de contrôle d'accès nécessite l'identification d'un utilisateur unique à partir d'un jeton d'accès (JWT ou réponse d'introspection de jeton associé), le serveur de ressources identifie l'utilisateur à partir de revendications non ré attribuables à d'autres utilisateurs. En général, cela implique l'utilisation d'une combinaison de revendications « iss » et « sub ».	2
10.3.4	Si le serveur de ressources requiert une force d'authentification, des méthodes ou une date d'expiration spécifiques, vérifiez que le jeton d'accès présenté respecte ces contraintes. Par exemple, s'il est présent, utilisez les revendications OIDC « acr », « amr » et « auth_time » respectivement.	2
10.3.5	Vérifiez que le serveur de ressources empêche l'utilisation de jetons d'accès volés ou la relecture de jetons d'accès (de parties non autorisées) en exigeant des jetons d'accès limités par l'expéditeur, soit Mutual TLS pour OAuth 2, soit OAuth 2 Demonstration of Proof of Possession (DPoP).	3

V10.4 Serveur d'autorisation OAuth

Ces exigences détaillent les responsabilités des serveurs d'autorisation OAuth, y compris les fournisseurs OpenID.

Pour l'authentification client, la méthode « self_signed_tls_client_auth » est autorisée avec les prérequis requis par la section 2.2 de la RFC 8705.

#	Description	Niveau
10.4.1	Vérifiez que le serveur d'autorisation valide les URI de redirection en fonction d'une liste d'autorisation spécifique au client d'URI préenregistrés à l'aide d'une comparaison de chaînes exacte.	1
10.4.2	Vérifiez que, si le serveur d'autorisation renvoie le code d'autorisation dans la réponse, celui-ci ne peut être utilisé qu'une seule fois pour une demande de jeton. Pour la deuxième demande valide avec un code d'autorisation déjà utilisé pour émettre un jeton d'accès, le serveur d'autorisation doit rejeter la demande de jeton et révoquer tous les jetons émis liés au code d'autorisation.	1
10.4.3	Vérifiez que le code d'autorisation est de courte durée. La durée de vie maximale peut atteindre 10 minutes pour les applications L1 et L2 et 1 minute pour les applications L3.	1
10.4.4	Vérifiez que, pour un client donné, le serveur d'autorisation autorise uniquement l'utilisation des autorisations nécessaires à ce client. Notez que les autorisations « token » (flux implicite) et « password » (flux d'informations d'identification du propriétaire de la ressource) ne doivent plus être utilisées.	1
10.4.5	Vérifiez que le serveur d'autorisation atténue les attaques par rejeu de jetons d'actualisation pour les clients publics, de préférence en utilisant des jetons d'actualisation limités par l'expéditeur, par exemple des jetons DPoP (Demonstrating Proof of Possession) ou des jetons d'accès liés à un certificat utilisant le protocole TLS mutuel (mTLS). Pour les applications L1 et L2, la rotation des jetons d'actualisation peut être utilisée. Si cette rotation est utilisée, le serveur d'autorisation doit invalider le jeton d'actualisation après utilisation et révoquer tous les jetons d'actualisation de cette autorisation si un jeton d'actualisation déjà utilisé et invalidé est fourni.	1

#	Description	Niveau
10.4.6	Vérifiez que, si l'octroi de code est utilisé, le serveur d'autorisation atténue les attaques par interception de code en exigeant une clé de preuve pour l'échange de code (PKCE). Pour les demandes d'autorisation, le serveur d'autorisation doit exiger une valeur « code_challenge » valide et ne doit pas accepter une valeur « plain » pour « code_challenge_method ». Pour une demande de jeton, il doit exiger la validation du paramètre « code_verifier ».	2
10.4.7	Vérifiez que si le serveur d'autorisation prend en charge l'enregistrement dynamique des clients non authentifiés, il réduit le risque d'applications clientes malveillantes. Il doit valider les métadonnées clientes, telles que les URI enregistrés, garantir le consentement de l'utilisateur et l'avertir avant de traiter une demande d'autorisation avec une application cliente non approuvée.	2
10.4.8	Vérifiez que les jetons d'actualisation ont une expiration absolue, y compris si l'expiration du jeton d'actualisation glissant est appliquée.	2
10.4.9	Vérifiez que les jetons d'actualisation et les jetons d'accès de référence peuvent être révoqués par un utilisateur autorisé à l'aide de l'interface utilisateur du serveur d'autorisation, afin d'atténuer le risque de clients malveillants ou de jetons volés.	2
10.4.10	Vérifiez que le client confidentiel est authentifié pour les demandes de canal arrière client-serveur autorisé telles que les demandes de jeton, les demandes d'autorisation poussée (PAR) et les demandes de révocation de jeton.	2
10.4.11	Vérifiez que le client confidentiel est authentifié pour les demandes de canal arrière client-serveur autorisé telles que les demandes de jeton, les demandes d'autorisation poussée (PAR) et les demandes de révocation de jeton.	2
10.4.12	Vérifiez que, pour un client donné, le serveur d'autorisation autorise uniquement la valeur « response_mode » dont ce client a besoin. Par exemple, en vérifiant cette valeur par rapport aux valeurs attendues ou en utilisant une requête d'autorisation poussée (PAR) ou une requête d'autorisation sécurisée par JWT (JAR).	3
10.4.13	Vérifiez que le type d'Autorisation « code » est toujours utilisé avec les demandes d'autorisation poussées (PAR).	3
10.4.14	Vérifiez que le serveur d'autorisation émet uniquement des jetons d'accès limités à l'expéditeur (preuve de possession), soit avec des jetons d'accès liés au certificat utilisant TLS mutuel (mTLS), soit avec des jetons d'accès liés au DPoP (démonstration de preuve de possession).	3
10.4.15	Vérifiez que, pour un client côté serveur (non exécuté sur l'appareil de l'utilisateur final), le serveur d'autorisation garantit que la valeur du paramètre « authorization_details » provient du backend client et que l'utilisateur ne l'a pas altérée. Par exemple, en exigeant l'utilisation d'une demande d'autorisation poussée (PAR) ou d'une demande d'autorisation sécurisée par JWT (JAR).	3
10.4.16	Vérifiez que le client est confidentiel et que le serveur d'autorisation requiert l'utilisation de méthodes d'authentification client fortes (basées sur la cryptographie à clé publique et résistantes aux attaques par relecture), telles que TLS mutuel ('tls_client_auth', 'self_signed_tls_client_auth') ou JWT à clé privée ('private_key_jwt').	3

V10.5 Client OIDC

Étant donné que la partie utilisatrice d'OIDC agit en tant que client OAuth, les exigences de la section « Client OAuth » s'appliquent également.

Notez que la section « Authentification avec un fournisseur d'identité » du chapitre « Authentification » contient également des exigences générales pertinentes.

#	Description	Niveau
10.5.1	Vérifiez que le client (en tant que partie utilisatrice) atténue les attaques par rejeu de jeton d'identification. Par exemple, en vous assurant que la valeur « nonce » du jeton d'identification correspond à la valeur « nonce » envoyée dans la demande d'authentification au fournisseur OpenID (appelée demande d'autorisation envoyée au serveur d'autorisation dans OAuth2).	2
10.5.2	Vérifiez que le client identifie de manière unique l'utilisateur à partir des revendications de jeton d'identification, généralement la revendication « sub », qui ne peut pas être réaffectée à d'autres utilisateurs (pour la portée d'un fournisseur d'identité).	2
10.5.3	Vérifiez que le client rejette les tentatives d'un serveur d'autorisation malveillant d'usurper l'identité d'un autre serveur d'autorisation via ses métadonnées. Le client doit rejeter les métadonnées du serveur d'autorisation si l'URL de l'émetteur figurant dans ces métadonnées ne correspond pas exactement à l'URL d'émetteur préconfigurée attendue par le client.	2
10.5.4	Vérifiez que le client valide que le jeton d'identification est destiné à être utilisé pour ce client (audience) en vérifiant que la revendication « aud » du jeton est égale à la valeur « client_id » pour le client.	2
10.5.5	Vérifiez que, lors de l'utilisation de la déconnexion du canal arrière OIDC, la partie utilisatrice atténue les risques de déni de service liés à la déconnexion forcée et à la confusion entre JWT dans le flux de déconnexion. Le client doit vérifier que le jeton de déconnexion est correctement typé avec la valeur « logout+jwt », qu'il contient la revendication « event » avec le nom de membre correct et qu'il ne contient pas de revendication « nonce ». Il est également recommandé d'utiliser une expiration courte (par exemple, 2 minutes).	2

V10.6 Fournisseur OpenID

Étant donné que les fournisseurs OpenID agissent comme des serveurs d'autorisation OAuth, les exigences de la section « Serveur d'autorisation OAuth » s'appliquent également.

Notez que si vous utilisez le flux de jeton d'identification (et non le flux de code), aucun jeton d'accès n'est émis et de nombreuses exigences pour OAuth AS ne sont pas applicables.

#	Description	Niveau
10.6.1	Vérifiez que le fournisseur OpenID n'autorise que les valeurs « code », « ciba », « id_token » ou « id_token code » pour le mode de réponse. Notez que « code » est préférable à « id_token code » (flux hybride OIDC) et que « token » (tout flux implicite) ne doit pas être utilisé.	2
10.6.2	Vérifiez que le fournisseur OpenID atténue les risques de déni de service liés à la déconnexion forcée. Pour ce faire, obtenez une confirmation explicite de l'utilisateur final	2

#	Description	Niveau
	ou, le cas échéant, validez les paramètres de la requête de déconnexion (initiée par la partie utilisatrice), comme « id_token_hint ».	

V10.7 Gestion du consentement

Ces exigences couvrent la vérification du consentement de l'utilisateur par le serveur d'autorisation. Sans vérification appropriée du consentement de l'utilisateur, un acteur malveillant peut obtenir des autorisations au nom de l'utilisateur par usurpation d'identité ou ingénierie sociale.

#	Description	Niveau
10.7.1	Vérifiez que le serveur d'autorisation garantit le consentement de l'utilisateur à chaque demande d'autorisation. Si l'identité du client ne peut être garantie, le serveur d'autorisation doit toujours demander explicitement le consentement de l'utilisateur.	2
10.7.2	Vérifiez que lorsque le serveur d'autorisation demande le consentement de l'utilisateur, il fournit des informations claires et suffisantes sur ce qui est consenti. Le cas échéant, ces informations doivent inclure la nature des autorisations demandées (généralement en fonction de la portée, du serveur de ressources et des détails d'autorisation RAR), l'identité de l'application autorisée et la durée de vie de ces autorisations.	2
10.7.3	Vérifiez que l'utilisateur peut consulter, modifier et révoquer les consentements qu'il a accordés via le serveur d'autorisation.	2

Références

Pour plus d'informations, voir également :

- oauth.net
- OWASP OAuth 2.0 Protocol Cheat Sheet

Pour les exigences liées à OAuth dans ASVS, les RFC publiées et à l'état de projet suivantes sont utilisées :

- RFC6749 The OAuth 2.0 Authorization Framework
- RFC6750 The OAuth 2.0 Authorization Framework: Bearer Token Usage
- RFC6819 OAuth 2.0 Threat Model and Security Considerations
- RFC7636 Proof Key for Code Exchange by OAuth Public Clients
- RFC7591 OAuth 2.0 Dynamic Client Registration Protocol
- RFC8628 OAuth 2.0 Device Authorization Grant
- RFC8707 Resource Indicators for OAuth 2.0
- RFC9068 JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens
- RFC9126 OAuth 2.0 Pushed Authorization Requests
- RFC9207 OAuth 2.0 Authorization Server Issuer Identification
- RFC9396 OAuth 2.0 Rich Authorization Requests
- RFC9449 OAuth 2.0 Demonstrating Proof of Possession (DPoP)

- RFC9700 Best Current Practice for OAuth 2.0 Security
- draft OAuth 2.0 for Browser-Based Applications
- draft The OAuth 2.1 Authorization Framework

Pour plus d'informations sur OpenID Connect, veuillez consulter :

- OpenID Connect Core 1.0
- FAPI 2.0 Security Profile

V11 Cryptographie

Objectif de contrôle

L'objectif de ce chapitre est de définir les meilleures pratiques pour l'utilisation générale de la cryptographie, ainsi que d'inculquer une compréhension fondamentale des principes cryptographiques et d'encourager une évolution vers des approches plus résilientes et modernes. Il encourage les actions suivantes :

- Mise en œuvre de systèmes cryptographiques robustes, sécurisés, adaptables à l'évolution des menaces et pérennes.
- Utilisation de mécanismes cryptographiques sécurisés et conformes aux meilleures pratiques du secteur.
- Maintien d'un système de gestion des clés cryptographiques sécurisé, avec des contrôles d'accès et des audits appropriés.
- Évaluation régulière du paysage cryptographique pour identifier les nouveaux risques et adapter les algorithmes en conséquence.
- Identification et gestion des cas d'utilisation cryptographiques tout au long du cycle de vie de l'application afin de garantir la prise en compte et la sécurité de tous les actifs cryptographiques.

Outre les principes généraux et les bonnes pratiques, ce document fournit également des informations techniques plus détaillées sur les exigences de l'annexe V – Normes de cryptographie. Cela inclut les algorithmes et les modes considérés comme « approuvés » aux fins des exigences de ce chapitre.

Les exigences qui utilisent la cryptographie pour résoudre un problème distinct, comme la gestion des secrets ou la sécurité des communications, figureront dans différentes parties de la norme.

V11.1 Inventaire et documentation cryptographiques

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

Il est important de veiller à ce que tous les actifs cryptographiques soient régulièrement découverts, inventoriés et évalués. Veuillez consulter l'annexe pour plus d'informations sur la procédure à suivre.

Il est également crucial de pérenniser les systèmes cryptographiques face à l'essor futur de l'informatique quantique. La cryptographie post-quantique (PQC) désigne les algorithmes cryptographiques conçus pour résister aux attaques des ordinateurs quantiques, qui sont susceptibles de casser des algorithmes largement utilisés tels que RSA et la cryptographie à courbe elliptique (ECC).

Veuillez consulter l'annexe pour obtenir des conseils actualisés sur les primitives et les normes PQC approuvées.

#	Description	Niveau
11.1.1	Vérifiez qu'il existe une politique documentée pour la gestion des clés cryptographiques et un cycle de vie des clés cryptographiques qui suit une norme de gestion des clés telle que NIST SP 800-57. Cela devrait inclure la garantie que les clés ne sont pas sur-partagées (par exemple, avec plus de deux entités pour les secrets partagés et plus d'une entité pour les clés privées).	2
11.1.2	Vérifiez qu'un inventaire cryptographique est réalisé, maintenu et régulièrement mis à jour, et qu'il inclut toutes les clés, algorithmes et certificats cryptographiques utilisés par	2

	l'application. Il doit également documenter les emplacements du système où les clés peuvent ou non être utilisées, ainsi que les types de données qui peuvent ou non être protégées par ces clés.	
11.1.3	Vérifiez que les mécanismes de découverte cryptographique sont utilisés pour identifier toutes les instances de cryptographie dans le système, y compris les opérations de chiffrement, de hachage et de signature.	3
11.1.4	Vérifier la tenue d'un inventaire cryptographique. Celui-ci doit inclure un plan documenté décrivant la migration vers de nouvelles normes cryptographiques, telles que la cryptographie post-quantique, afin de réagir aux menaces futures.	3

V11.2 Mise en œuvre de la cryptographie sécurisée

Cette section définit les exigences relatives à la sélection, à la mise en œuvre et à la gestion continue des algorithmes cryptographiques de base d'une application. L'objectif est de garantir que seules des primitives cryptographiques robustes et reconnues par l'industrie soient déployées, conformément aux normes en vigueur (par exemple, NIST, ISO/IEC) et aux meilleures pratiques. Les organisations doivent s'assurer que chaque composant cryptographique est sélectionné sur la base de preuves validées par des pairs et de tests de sécurité pratiques.

#	Description	Niveau
11.2.1	Vérifiez que les implémentations validées par l'industrie (y compris les bibliothèques et les implémentations accélérées par le matériel) sont utilisées pour les opérations cryptographiques.	2
11.2.2	Vérifiez que l'application est conçue avec une agilité cryptographique, de sorte que les nombres aléatoires, le chiffrement authentifié, le MAC ou les algorithmes de hachage, les longueurs de clés, les tours, les chiffrements et les modes puissent être reconfigurés, mis à niveau ou échangés à tout moment, afin de se protéger contre les failles cryptographiques. De même, il doit être possible de remplacer les clés et les mots de passe, ainsi que de rechiffrer les données. Cela permettra des mises à niveau transparentes vers la cryptographie post-quantique (PQC), une fois que des implémentations hautement sécurisées de schémas ou de normes PQC approuvés seront largement disponibles.	2
11.2.3	Vérifiez que toutes les primitives cryptographiques utilisent un minimum de 128 bits de sécurité en fonction de l'algorithme, de la taille de la clé et de la configuration. Par exemple, une clé ECC de 256 bits offre environ 128 bits de sécurité, tandis que RSA nécessite une clé de 3072 bits pour atteindre 128 bits de sécurité.	2
11.2.4	Vérifiez que toutes les opérations cryptographiques sont à temps constant, sans opérations de « court-circuit » dans les comparaisons, les calculs ou les retours, pour éviter toute fuite d'informations.	3
11.2.5	Vérifiez que tous les modules cryptographiques échouent en toute sécurité et que les erreurs sont gérées de manière à ne pas permettre de vulnérabilités, telles que les attaques Padding Oracle.	3

V11.3 Algorithmes de chiffrement

Les algorithmes de chiffrement authentifiés basés sur AES et CHACHA20 constituent l'épine dorsale de la pratique cryptographique moderne.

#	Description	Niveau
11.3.1	Vérifiez que les modes de bloc non sécurisés (par exemple, ECB) et les schémas de remplissage faibles (par exemple, PKCS#1 v1.5) ne sont pas utilisés.	1
11.3.2	Vérifiez que seuls les chiffrements et modes approuvés tels que AES avec GCM sont utilisés.	1
11.3.3	Vérifiez que les données chiffrées sont protégées contre toute modification non autorisée, de préférence en utilisant une méthode de chiffrement authentifiée approuvée ou en combinant une méthode de chiffrement approuvée avec un algorithme MAC approuvé.	2
11.3.4	Vérifiez que les nonces, les vecteurs d'initialisation et autres nombres à usage unique ne sont pas utilisés pour plus d'une paire clé de chiffrement/élément de données. La méthode de génération doit être adaptée à l'algorithme utilisé.	3
11.3.5	Vérifiez que toute combinaison d'un algorithme de chiffrement et d'un algorithme MAC fonctionne en mode chiffrement puis MAC.	3

V11.4 Hachage et fonctions basées sur le hachage

Les hachages cryptographiques sont utilisés dans une grande variété de protocoles cryptographiques, tels que les signatures numériques, HMAC, les fonctions de dérivation de clés (KDF), la génération de bits aléatoires et le stockage de mots de passe. La sécurité d'un système cryptographique dépend des fonctions de hachage sous-jacentes utilisées. Cette section décrit les exigences relatives à l'utilisation de fonctions de hachage sécurisées dans les opérations cryptographiques.

Pour le stockage des mots de passe, ainsi que pour l'annexe sur la cryptographie, la OWASP Password Storage Cheatsheet fournira également un contexte et des conseils utiles.

#	Description	Niveau
11.4.1	Vérifiez que seules les fonctions de hachage approuvées sont utilisées pour les cas d'utilisation cryptographiques généraux, notamment les signatures numériques, HMAC, KDF et la génération de bits aléatoires. Les fonctions de hachage non autorisées, telles que MD5, ne doivent pas être utilisées à des fins cryptographiques.	1
11.4.2	Vérifiez que les mots de passe sont stockés à l'aide d'une fonction de dérivation de clés approuvée et gourmande en ressources de calcul (également appelée « fonction de hachage de mots de passe »), dont les paramètres sont configurés conformément aux directives en vigueur. Ces paramètres doivent équilibrer sécurité et performances afin de rendre les attaques par force brute suffisamment difficiles pour le niveau de sécurité requis.	2
11.4.3	Vérifiez que les fonctions de hachage utilisées dans les signatures numériques, dans le cadre de l'authentification ou de l'intégrité des données, sont résistantes aux collisions et possèdent des longueurs de bits appropriées. Si la résistance aux collisions est requise, la longueur de sortie doit être d'au moins 256 bits. Si seule la résistance aux attaques de seconde pré-image est requise, la longueur de sortie doit être d'au moins 128 bits.	2
11.4.4	Vérifiez que l'application utilise des fonctions de dérivation de clés approuvées avec des paramètres d'extension de clés lors de la dérivation de clés secrètes à partir de mots de passe. Les paramètres utilisés doivent concilier sécurité et performances afin d'empêcher les attaques par force brute de compromettre la clé cryptographique obtenue.	2

V11.5 Valeurs aléatoires

La génération de nombres pseudo-aléatoires cryptographiquement sécurisée (CSPRNG) est extrêmement difficile à mettre en œuvre. En général, les bonnes sources d'entropie d'un système s'épuisent rapidement si elles sont surexploitées, tandis que des sources moins aléatoires peuvent conduire à des clés et des secrets prévisibles.

#	Description	Niveau
11.5.1	Vérifiez que tous les nombres et chaînes aléatoires destinés à être non devinables doivent être générés à l'aide d'un générateur de nombres pseudo-aléatoires cryptographiquement sécurisé (CSPRNG) et posséder au moins 128 bits d'entropie. Notez que les UUID ne respectent pas cette condition.	2
11.5.2	Vérifiez que le mécanisme de génération de nombres aléatoires utilisé est conçu pour fonctionner en toute sécurité, même en cas de forte demande.	3

V11.6 Cryptographie à clé publique

La cryptographie à clé publique sera utilisée lorsqu'il est impossible ou non souhaitable de partager une clé secrète entre plusieurs parties.

Dans ce contexte, des mécanismes d'échange de clés approuvés, tels que Diffie-Hellman et Elliptic Curve Diffie-Hellman (ECDH), sont nécessaires pour garantir la sécurité du cryptosystème face aux menaces modernes. Le chapitre « Communication sécurisée » décrit les exigences pour TLS. Les exigences de cette section sont donc destinées aux situations où la cryptographie à clé publique est utilisée dans des cas d'utilisation autres que TLS.

#	Description	Niveau
11.6.1	Vérifier que seuls des algorithmes cryptographiques et des modes opératoires approuvés sont utilisés pour la génération et l'amorçage des clés, ainsi que pour la génération et la vérification des signatures numériques. Les algorithmes de génération de clés ne doivent pas générer de clés non sécurisées et vulnérables aux attaques connues, par exemple les clés RSA vulnérables à la factorisation de Fermat.	2
11.6.2	Vérifiez que des algorithmes cryptographiques approuvés sont utilisés pour l'échange de clés (tels que Diffie-Hellman), en veillant à ce que les mécanismes d'échange de clés utilisent des paramètres sécurisés. Cela permettra d'éviter les attaques contre le processus d'établissement des clés, susceptibles de conduire à des attaques de type « adversaire du milieu » ou à des failles cryptographiques.	3

V11.7 Cryptographie des données en cours d'utilisation

La protection des données pendant leur traitement est "le nec plus ultra". Des techniques telles que le chiffrement intégral de la mémoire, le chiffrement des données en transit et s'assurer que le les données sont chiffrés le plus tôt possible après utilisation.

#	Description	Niveau
11.7.1	Vérifiez que le chiffrement complet de la mémoire est utilisé pour protéger les données sensibles pendant leur utilisation, empêchant ainsi l'accès par des utilisateurs ou des processus non autorisés.	3
11.7.2	Vérifiez que la minimisation des données garantit que la quantité minimale de données est exposée pendant le traitement et assurez-vous que les données sont chiffrées immédiatement après utilisation ou dès que possible.	3

Références

Pour plus d'informations, voir également :

- OWASP Web Security Testing Guide: Testing for Weak Cryptography
- OWASP Cryptographic Storage Cheat Sheet
- FIPS 140-3
- NIST SP 800-57

V12 Communication sécurisée

Objectif de contrôle

Ce chapitre comprend des exigences relatives aux mécanismes spécifiques qui doivent être mis en place pour protéger les données en transit, à la fois entre un client utilisateur final et un service backend, ainsi qu'entre les services internes et backend.

Les concepts généraux promus par ce chapitre sont les suivants

- S'assurer que les communications sont chiffrées à l'extérieur et, idéalement, à l'intérieur.
- Configurer les mécanismes de chiffrement en utilisant les dernières recommandations, y compris les algorithmes et les chiffrements préférés.
- S'assurer que les communications ne sont pas interceptées par des parties non autorisées à l'aide de certificats signés.

Outre les principes généraux et les meilleures pratiques, l'ASVS fournit également des informations techniques plus approfondies sur la puissance cryptographique dans l'annexe C - Normes de cryptographie.

V12.1 Conseils généraux de sécurité TLS

Cette section fournit des conseils initiaux sur la sécurisation des communications TLS. Des outils à jour doivent être utilisés pour vérifier régulièrement la configuration TLS.

Bien que l'utilisation de certificats TLS génériques ne soit pas intrinsèquement dangereuse, la compromission d'un certificat déployé dans tous les environnements (par exemple, production, préproduction, développement et test) peut compromettre la sécurité des applications qui l'utilisent. Une protection et une gestion appropriées, ainsi que l'utilisation de certificats TLS distincts dans différents environnements, sont recommandées si possible.

#	Description	Niveau
12.1.1	Vérifiez que seules les dernières versions recommandées du protocole TLS sont activées, telles que TLS 1.2 et TLS 1.3. La dernière version du protocole TLS doit être privilégiée.	1
12.1.2	Vérifiez que seules les suites de chiffrement recommandées sont activées, les suites de chiffrement les plus puissantes étant définies comme préférées. Les applications L3 doivent uniquement prendre en charge les suites de chiffrement assurant le secret d'acheminement.	2
12.1.3	Vérifiez que l'application valide que les certificats clients mTLS sont approuvés avant d'utiliser l'identité du certificat pour l'authentification ou l'autorisation.	2
12.1.4	Vérifiez que la révocation de certification appropriée, telle que l'agrafage du protocole OCSP (Online Certificate Status Protocol), est activée et configurée.	3
12.1.5	Vérifiez que Encrypted Client Hello (ECH) est activé dans les paramètres TLS de l'application pour empêcher l'exposition de métadonnées sensibles, telles que l'indication du nom du serveur (SNI), pendant les processus de négociation TLS.	3

V12.2 Communication HTTPS avec des services externes

Assurez-vous que tout le trafic HTTP vers les services externes exposés par l'application est envoyé chiffré, avec des certificats de confiance publique.

#	Description	Niveau
12.2.1	Vérifiez que TLS est utilisé pour toute connectivité entre un client et des services externes basés sur HTTP, et ne recourt pas à des communications non sécurisées ou non chiffrées.	1
12.2.2	Vérifiez que les services externes utilisent des certificats TLS publiquement approuvés.	1

V12.3 Sécurité des communications entre services généraux

Les communications serveur (internes et externes) ne se limitent pas au protocole HTTP. Les connexions vers et depuis d'autres systèmes doivent également être sécurisées, idéalement via TLS.

#	Description	Niveau
12.3.1	Vérifiez qu'un protocole chiffré tel que TLS est utilisé pour toutes les connexions entrantes et sortantes vers et depuis l'application, y compris les systèmes de surveillance, les outils de gestion, l'accès à distance et SSH, les intergiciels, les bases de données, les mainframes, les systèmes partenaires ou les API externes. Le serveur ne doit pas recourir à des protocoles non sécurisés ou non chiffrés.	2
12.3.2	Vérifiez que les clients TLS valident les certificats reçus avant de communiquer avec un serveur TLS.	2
12.3.3	Vérifiez que TLS ou un autre mécanisme de chiffrement de transport approprié est utilisé pour toute connectivité entre les services internes basés sur HTTP au sein de l'application et ne se replie pas vers des communications non sécurisées ou non chiffrées.	2
12.3.4	Vérifiez que les connexions TLS entre les services internes utilisent des certificats de confiance. Lorsque des certificats générés en interne ou auto-signés sont utilisés, le service consommateur doit être configuré pour n'approuver que des autorités de certification internes spécifiques et des certificats auto-signés spécifiques.	2
12.3.5	Vérifiez que les services communiquant en interne au sein d'un système (communications intra-services) utilisent une authentification forte pour garantir la vérification de chaque point de terminaison. Des méthodes d'authentification forte, telles que l'authentification client TLS, doivent être utilisées pour garantir l'identité, en utilisant une infrastructure à clé publique et des mécanismes résistants aux attaques par rejeu. Pour les architectures de microservices, envisagez d'utiliser un maillage de services pour simplifier la gestion des certificats et renforcer la sécurité.	3

Références

Pour plus d'informations, voir également :

- OWASP - Transport Layer Security Cheat Sheet
- Mozilla's Server Side TLS configuration guide
- Mozilla's tool to generate known good TLS configurations.
- O-Saft - OWASP Project to validate TLS configuration

V13 Configuration

Objectif de contrôle

La configuration par défaut de l'application doit être sécurisée pour une utilisation sur Internet.

Ce chapitre fournit des conseils sur les différentes configurations nécessaires pour y parvenir, y compris celles appliquées pendant le développement, la construction et le déploiement.

Les sujets abordés incluent la prévention des fuites de données, la gestion sécurisée des communications entre les composants et la protection des secrets.

V13.1 Documentation de configuration

Cette section décrit les exigences de documentation relatives à la communication de l'application avec les services internes et externes, ainsi que les techniques permettant d'éviter toute perte de disponibilité due à l'inaccessibilité des services. Elle aborde également la documentation relative aux secrets.

#	Description	Niveau
13.1.1	Vérifiez que tous les besoins de communication de l'application sont documentés. Cela doit inclure les services externes dont l'application dépend et les cas où un utilisateur final pourrait fournir un emplacement externe auquel l'application se connectera.	2
13.1.2	Vérifiez que pour chaque service utilisé par l'application, la documentation définit le nombre maximal de connexions simultanées (par exemple, les limites du pool de connexions) et le comportement de l'application lorsque cette limite est atteinte, y compris les mécanismes de secours ou de récupération, pour éviter les conditions de déni de service.	3
13.1.3	Vérifiez que la documentation de l'application définit des stratégies de gestion des ressources pour chaque système ou service externe utilisé (par exemple, bases de données, descripteurs de fichiers, threads, connexions HTTP). Cela doit inclure les procédures de libération des ressources, les paramètres de délai d'expiration, la gestion des échecs et l'implémentation de la logique de nouvelle tentative, en spécifiant les limites de nouvelle tentative, les délais et les algorithmes de retour arrière. Pour les opérations de requête-réponse HTTP synchrones, la documentation doit imposer des délais d'expiration courts et soit désactiver les nouvelles tentatives, soit les limiter strictement afin d'éviter les retards en cascade et l'épuisement des ressources.	3
13.1.4	Vérifiez que la documentation de l'application définit les secrets essentiels à la sécurité de l'application et un calendrier de rotation de ceux-ci, en fonction du modèle de menace et des exigences commerciales de l'organisation.	3

V13.2 Configuration de la communication backend

Les applications interagissent avec plusieurs services, notamment des API, des bases de données ou d'autres composants. Ceux-ci peuvent être considérés comme internes à l'application, mais non inclus dans les mécanismes de contrôle d'accès standard de l'application, ou être entièrement externes. Dans les deux cas, il est nécessaire de configurer l'application pour interagir en toute sécurité avec ces composants et, si nécessaire, de protéger cette configuration.

Remarque : le chapitre « Communication sécurisée » fournit des conseils pour le chiffrement en transit.

#	Description	Niveau
13.2.1	Vérifiez que les communications entre les composants de l'application backend qui ne prennent pas en charge le mécanisme de session utilisateur standard de l'application, notamment les API, les intergiciels et les couches de données, sont authentifiées. L'authentification doit utiliser des comptes de service individuels, des jetons à court terme ou une authentification par certificat, et non des identifiants immuables tels que des mots de passe, des clés API ou des comptes partagés avec accès privilégié.	2
13.2.2	Vérifiez que les communications entre les composants de l'application backend, y compris les services locaux ou du système d'exploitation, les API, les intergiciels et les couches de données, sont effectuées avec des comptes dotés des privilèges les moins nécessaires.	2
13.2.3	Vérifiez que si des informations d'identification doivent être utilisées pour l'authentification du service, les informations d'identification utilisées par le consommateur ne sont pas des informations d'identification par défaut (par exemple, root/root ou admin/admin).	2
13.2.4	Vérifiez qu'une liste blanche est utilisée pour définir les ressources ou systèmes externes avec lesquels l'application est autorisée à communiquer (par exemple, pour les requêtes sortantes, les chargements de données ou l'accès aux fichiers). Cette liste blanche peut être implémentée au niveau de la couche applicative, du serveur web, du pare-feu ou d'une combinaison de différentes couches.	2
13.2.5	Vérifiez que le serveur Web ou d'application est configuré avec une liste autorisée de ressources ou de systèmes vers lesquels le serveur peut envoyer des requêtes ou charger des données ou des fichiers.	2
13.2.6	Vérifiez que lorsque l'application se connecte à des services distincts, elle suit la configuration documentée pour chaque connexion, comme le nombre maximal de connexions parallèles, le comportement lorsque le nombre maximal de connexions autorisées est atteint, les délais d'expiration de connexion et les stratégies de nouvelle tentative.	3

V13.3 Gestion du secret

La gestion des secrets est une tâche de configuration essentielle pour garantir la protection des données utilisées dans l'application. Les exigences spécifiques en matière de cryptographie sont décrites dans le chapitre « Cryptographie », mais cette section se concentre sur les aspects de gestion et de traitement des secrets.

#	Description	Niveau
13.3.1	Vérifiez qu'une solution de gestion des secrets, telle qu'un coffre-fort de clés, est utilisée pour créer, stocker, contrôler l'accès et détruire en toute sécurité les secrets backend. Ceux-ci peuvent inclure des mots de passe, du matériel de clé, des intégrations avec des bases de données et des systèmes tiers, des clés et des graines pour des jetons temporels, d'autres secrets internes et des clés API. Les secrets ne doivent pas être inclus dans le code source de l'application ni dans les artefacts de build. Pour une application L3, cela doit impliquer une solution matérielle telle qu'un HSM.	2
13.3.2	Vérifiez que l'accès aux ressources secrètes respecte le principe du moindre privilège.	2
13.3.3	Vérifiez que toutes les opérations cryptographiques sont effectuées à l'aide d'un module de sécurité isolé (tel qu'un coffre-fort ou un module de sécurité matériel) pour gérer et	3

#	Description	Niveau
	protéger en toute sécurité le matériel clé contre toute exposition en dehors du module de sécurité.	
13.3.4	Vérifiez que les secrets sont configurés pour expirer et être renouvelés en fonction de la documentation de l'application.	3

V13.4 Fuite d'informations involontaire

Les configurations de production doivent être renforcées afin d'éviter la divulgation de données inutiles. Nombre de ces problèmes sont rarement considérés comme des risques importants, mais sont souvent associés à d'autres vulnérabilités. Si ces problèmes ne sont pas présents par défaut, la barre est plus haute pour les attaques contre une application.

Par exemple, masquer la version des composants côté serveur n'élimine pas la nécessité de corriger tous les composants, et désactiver la liste des dossiers ne supprime pas la nécessité d'utiliser des contrôles d'autorisation ou de garder les fichiers loin du dossier public, mais cela place la barre plus haut.

#	Description	Niveau
13.4.1	Vérifiez que l'application est déployée soit sans aucune métadonnée de contrôle de source, y compris les dossiers .git ou .svn, soit de manière à ce que ces dossiers soient inaccessibles à la fois en externe et à l'application elle-même.	1
13.4.2	Vérifiez que les modes de débogage sont désactivés pour tous les composants dans les environnements de production afin d'éviter l'exposition des fonctionnalités de débogage et la fuite d'informations.	2
13.4.3	Vérifiez que les serveurs Web n'exposent pas les listes de répertoires aux clients, sauf si cela est explicitement prévu.	2
13.4.4	Vérifiez que l'utilisation de la méthode HTTP TRACE n'est pas prise en charge dans les environnements de production, afin d'éviter toute fuite d'informations potentielle.	2
13.4.5	Vérifiez que la documentation (comme pour les API internes) et les endpoints de surveillance ne sont pas exposés, sauf si cela est explicitement prévu.	2
13.4.6	Vérifiez que l'application n'expose pas d'informations de version détaillées des composants backend.	3
13.4.7	Vérifiez que la couche web est configuré pour servir uniquement les fichiers avec des extensions de fichier spécifiques afin d'éviter toute fuite involontaire d'informations, de configuration et de code source.	3

Références

Pour plus d'informations, voir également :

- OWASP Web Security Testing Guide: Configuration and Deployment Management Testing

V14 Protection des données

Objectif de contrôle

Les applications ne peuvent pas prendre en compte tous les modèles d'utilisation et comportements des utilisateurs et doivent donc mettre en œuvre des contrôles pour limiter l'accès non autorisé aux données sensibles sur les appareils clients.

Ce chapitre comprend les exigences relatives à la définition des données à protéger, de la manière dont elles doivent être protégées et des mécanismes spécifiques à mettre en œuvre ou des pièges à éviter.

Un autre élément à prendre en compte pour la protection des données est l'extraction massive, la modification ou l'utilisation excessive. Les exigences de chaque système étant probablement très différentes, déterminer ce qui est « anormal » doit tenir compte du modèle de menace et du risque métier. Du point de vue d'ASVS, la détection de ces problèmes est traitée dans le chapitre « Journalisation de sécurité et gestion des erreurs », et la définition de limites dans le chapitre « Validation et logique métier ».

V14.1 Documentation sur la protection des données

Une condition préalable essentielle à la protection des données est de catégoriser les données considérées comme sensibles. Il existe probablement plusieurs niveaux de sensibilité, et pour chaque niveau, les contrôles requis pour protéger les données seront différents.

Il existe diverses réglementations et lois sur la confidentialité qui régissent la manière dont les applications doivent gérer le stockage, l'utilisation et la transmission des informations personnelles sensibles. Cette section ne vise plus à reproduire ces types de législation sur la protection des données ou la confidentialité, mais se concentre plutôt sur les considérations techniques clés pour la protection des données sensibles. Veuillez consulter les lois et réglementations locales et, si nécessaire, faire appel à un spécialiste de la confidentialité ou à un avocat qualifié.

#	Description	Niveau
14.1.1	Vérifiez que toutes les données sensibles créées et traitées par l'application ont été identifiées et classées selon leur niveau de protection. Cela inclut les données uniquement codées et donc facilement déchiffrables, telles que les chaînes Base64 ou la charge utile en texte clair d'un JWT. Les niveaux de protection doivent tenir compte des réglementations et normes de protection des données et de confidentialité auxquelles l'application est tenue de se conformer.	2
14.1.2	Vérifier que tous les niveaux de protection des données sensibles sont assortis d'un ensemble documenté d'exigences de protection. Cela doit inclure (sans s'y limiter) les exigences relatives au chiffrement général, à la vérification de l'intégrité, à la conservation, à la journalisation des données, aux contrôles d'accès aux données sensibles dans les journaux, au chiffrement au niveau de la base de données, aux technologies de confidentialité et de renforcement de la confidentialité à utiliser, ainsi qu'aux autres exigences de confidentialité.	2

V14.2 Protection générale des données

Cette section présente diverses exigences pratiques relatives à la protection des données. La plupart sont spécifiques à des problématiques particulières, telles que les fuites involontaires de données, mais il existe également une exigence générale de mise en œuvre de contrôles de protection en fonction du niveau de protection requis pour chaque donnée.

#	Description	Niveau
14.2.1	Vérifiez que les données sensibles sont uniquement envoyées au serveur dans les champs de corps ou d'en-tête du message HTTP, et que l'URL et la chaîne de requête ne contiennent pas d'informations sensibles, telles qu'une clé API ou un jeton de session.	1
14.2.2	Vérifiez que l'application empêche la mise en cache des données sensibles dans les composants du serveur, tels que les équilibreurs de charge et les caches d'application, ou garantit que les données sont purgées en toute sécurité après utilisation.	2
14.2.3	Vérifiez que les données sensibles définies ne sont pas envoyées à des parties non fiables (par exemple, des trackers d'utilisateurs) pour empêcher la collecte indésirable de données en dehors du contrôle de l'application.	2
14.2.4	Vérifiez que les contrôles autour des données sensibles liés au chiffrement, à la vérification de l'intégrité, à la conservation, à la manière dont les données doivent être enregistrées, aux contrôles d'accès autour des données sensibles dans les journaux, à la confidentialité et aux technologies d'amélioration de la confidentialité, sont mis en œuvre comme défini dans la documentation pour le niveau de protection des données spécifiques.	2
14.2.5	Vérifiez que les mécanismes de mise en cache sont configurés pour ne mettre en cache que les réponses dont le type de contenu est conforme à la ressource et qui ne contiennent pas de contenu sensible et dynamique. Le serveur web doit renvoyer une réponse 404 ou 302 lors de l'accès à un fichier inexistant plutôt qu'un fichier valide différent. Cela devrait empêcher les attaques par empoisonnement de cache web.	3
14.2.6	Vérifiez que l'application ne renvoie que les données sensibles minimales requises pour son fonctionnement. Par exemple, ne renvoyez que certains chiffres d'un numéro de carte de crédit, et non le numéro complet. Si les données complètes sont requises, elles doivent être masquées dans l'interface utilisateur, sauf si l'utilisateur les consulte spécifiquement.	3
14.2.7	Vérifiez que les informations sensibles sont soumises à une classification de conservation des données, en veillant à ce que les données obsolètes ou inutiles soient supprimées automatiquement, selon un calendrier défini ou selon les besoins de la situation.	3
14.2.8	Vérifiez que les informations sensibles sont supprimées des métadonnées des fichiers soumis par l'utilisateur, sauf si le stockage est autorisé par l'utilisateur.	3

V14.3 Protection des données côté client

Cette section contient des exigences empêchant la fuite de données de manière spécifique du côté client ou de l'agent utilisateur d'une application.

#	Description	Niveau
14.3.1	Vérifiez que les données authentifiées sont effacées du stockage client, comme le DOM du navigateur, après la fin du client ou de la session. Le champ d'en-tête de réponse HTTP « Clear-Site-Data » peut être utile, mais le côté client devrait également pouvoir effectuer la correction si la connexion au serveur n'est pas disponible à la fin de la session.	1
14.3.2	Vérifiez que l'application définit suffisamment de champs d'en-tête de réponse HTTP anti-caching (c'est-à-dire Cache-Control : no-store) afin que les données sensibles ne soient pas mises en cache dans les navigateurs.	2

#	Description	Niveau
14.3.3	Vérifiez que les données stockées dans le stockage du navigateur (telles que localStorage, sessionStorage, IndexedDB ou les cookies) ne contiennent pas de données sensibles, à l'exception des jetons de session.	2

Références

Pour plus d'informations, voir également :

- Consider using the Security Headers website to check security and anti-caching header fields
- Documentation about anti-caching headers by Mozilla
- OWASP Secure Headers project
- OWASP Privacy Risks Project
- OWASP User Privacy Protection Cheat Sheet
- Australian Privacy Principle 11 - Security of personal information
- European Union General Data Protection Regulation (GDPR) overview
- European Union Data Protection Supervisor - Internet Privacy Engineering Network
- Information on the "Clear-Site-Data" header
- White paper on Web Cache Deception

V15 Développement et architecture sécurisés

Objectif de contrôle

De nombreuses exigences de l'ASVS se rapportent soit à un domaine particulier de la sécurité, comme l'authentification ou l'autorisation, soit à un type particulier de fonctionnalité de l'application, comme la journalisation ou la gestion des fichiers.

Ce chapitre présente les exigences générales de sécurité à prendre en compte lors de la conception et du développement d'applications. Ces exigences portent non seulement sur la propreté de l'architecture et la qualité du code, mais aussi sur les pratiques spécifiques d'architecture et de codage nécessaires à la sécurité des applications.

V15.1 Documentation sur le développement et l'architecture sécurisés

De nombreuses exigences pour établir une architecture sécurisée et défendable dépendent d'une documentation claire des décisions prises concernant la mise en œuvre de contrôles de sécurité spécifiques et des composants utilisés dans l'application.

Cette section décrit les exigences en matière de documentation, notamment l'identification des composants considérés comme contenant des « fonctionnalités dangereuses » ou comme des « composants à risque ».

Un composant présentant une « fonctionnalité dangereuse » peut être un composant développé en interne ou par un tiers, effectuant des opérations telles que la désérialisation de données non fiables, l'analyse de fichiers bruts ou binaires, l'exécution de code dynamique ou la manipulation directe de la mémoire. Les vulnérabilités dans ces types d'opérations présentent un risque élevé de compromettre l'application et d'exposer potentiellement son infrastructure sous-jacente.

Un « composant à risque » est une bibliothèque tierce (c'est-à-dire non développée en interne) dont les contrôles de sécurité concernant ses processus de développement ou ses fonctionnalités sont absents ou mal implémentés. Il peut s'agir, par exemple, de composants mal entretenus, non pris en charge, en fin de vie ou présentant des antécédents de vulnérabilités importantes.

Cette section souligne également l'importance de définir des délais appropriés pour traiter les vulnérabilités des composants tiers.

#	Description	Niveau
15.1.1	Vérifiez que la documentation de l'application définit les délais de correction basés sur les risques pour les versions de composants tiers présentant des vulnérabilités et pour la mise à jour des bibliothèques en général, afin de minimiser les risques liés à ces composants.	1
15.1.2	Vérifiez qu'un catalogue d'inventaire, tel qu'une nomenclature logicielle (SBOM), est conservé pour toutes les bibliothèques tierces utilisées, y compris en vérifiant que les composants proviennent de référentiels prédéfinis, fiables et continuellement maintenus.	2
15.1.3	Vérifiez que la documentation de l'application identifie les fonctionnalités consommatrices en temps ou gourmandes en ressources. Elle doit notamment indiquer comment prévenir une perte de disponibilité due à une utilisation excessive de ces fonctionnalités et éviter que la génération d'une réponse ne prenne plus de temps que le délai d'expiration du consommateur. Les mesures de protection potentielles peuvent inclure le traitement asynchrone, l'utilisation de files d'attente et la limitation des processus parallèles par utilisateur et par application.	2
15.1.4	Vérifiez que la documentation de l'application met en évidence les bibliothèques tierces qui sont considérées comme des « composants à risque ».	3

#	Description	Niveau
15.1.5	Vérifiez que la documentation de l'application met en évidence les parties de l'application où des « fonctionnalités dangereuses » sont utilisées.	3

V15.2 Architecture de sécurité et dépendances

Cette section inclut les exigences relatives à la gestion des dépendances et des composants risqués, obsolètes ou non sécurisés.

Elle inclut également l'utilisation de techniques au niveau architectural telles que le sandboxing, l'encapsulation, la conteneurisation et l'isolation du réseau pour réduire l'impact de l'utilisation d'« opérations dangereuses » ou de « composants risqués » (tels que définis dans la section précédente) et éviter la perte de disponibilité due à une utilisation excessive de fonctionnalités exigeantes en ressources.

#	Description	Niveau
15.2.1	Vérifiez que l'application contient uniquement des composants qui n'ont pas dépassé les délais de mise à jour et de correction documentés.	1
15.2.2	Vérifiez que l'application a mis en œuvre des défenses contre la perte de disponibilité due à des fonctionnalités qui prennent du temps ou qui nécessitent des ressources, sur la base des décisions et stratégies de sécurité documentées à cet effet.	2
15.2.3	Vérifiez que l'environnement de production inclut uniquement les fonctionnalités nécessaires au fonctionnement de l'application et n'expose pas de fonctionnalités superflues telles que du code de test, des exemples d'extraits et des fonctionnalités de développement.	2
15.2.4	Vérifiez que les composants tiers et toutes leurs dépendances transitives sont inclus à partir du référentiel attendu, qu'il soit détenu en interne ou qu'il s'agisse d'une source externe, et qu'il n'y a aucun risque d'attaque par confusion de dépendances.	3
15.2.5	Vérifiez que l'application implémente des protections supplémentaires autour des parties documentées comme contenant des « fonctionnalités dangereuses » ou utilisant des bibliothèques tierces considérées comme des « composants à risque ». Cela peut inclure des techniques telles que le sandboxing, l'encapsulation, la conteneurisation ou l'isolation au niveau du réseau pour retarder et dissuader les attaquants qui compromettent une partie de l'application de se propager ailleurs.	3

V15.3 Développement défensif

Cette section couvre les types de vulnérabilités, y compris le jonglage de types, la pollution de prototypes et autres, qui résultent de l'utilisation de modèles de codage non sécurisés dans un langage particulier. Certaines peuvent ne pas être pertinentes pour tous les langages, tandis que d'autres bénéficieront de correctifs spécifiques à chaque langage ou pourront être liées à la façon dont un langage ou un framework particulier gère une fonctionnalité telle que les paramètres HTTP. Elle aborde également le risque lié à l'absence de validation cryptographique des mises à jour d'applications.

Elle prend également en compte les risques liés à l'utilisation d'objets pour représenter des éléments de données, ainsi qu'à leur acceptation et leur renvoi via des API externes. Dans ce cas, l'application doit s'assurer que les champs de données non accessibles en écriture ne sont pas modifiés par l'utilisateur (affectation de masse) et que l'API sélectionne les champs de données renvoyés. Lorsque l'accès aux champs dépend des autorisations de l'utilisateur, il convient d'en tenir compte dans le contexte des exigences de contrôle d'accès au niveau des champs décrites dans le chapitre « Autorisation ».

#	Description	Niveau
15.3.1	Vérifiez que l'application ne renvoie que le sous-ensemble requis de champs d'un objet de données. Par exemple, elle ne doit pas renvoyer un objet de données entier, car certains champs individuels ne doivent pas être accessibles aux utilisateurs.	1
15.3.2	Vérifiez que lorsque le backend de l'application effectue des appels vers des URL externes, il est configuré pour ne pas suivre les redirections, sauf s'il s'agit d'une fonctionnalité prévue.	2
15.3.3	Vérifiez que l'application dispose de contre-mesures pour se protéger contre les attaques d'affectation de masse en limitant les champs autorisés par contrôleur et action, par exemple, il n'est pas possible d'insérer ou de mettre à jour une valeur de champ lorsqu'elle n'était pas destinée à faire partie de cette action.	2
15.3.4	Vérifiez que tous les composants proxy et middleware transfèrent correctement l'adresse IP d'origine de l'utilisateur à l'aide de champs de données fiables qui ne peuvent pas être manipulés par l'utilisateur final, et que l'application et le serveur Web utilisent cette valeur correcte pour la journalisation et les décisions de sécurité telles que la limitation du débit, en tenant compte du fait que même l'adresse IP d'origine peut ne pas être fiable en raison d'adresses IP dynamiques, de VPN ou de pare-feu d'entreprise.	2
15.3.5	Vérifiez que l'application garantit explicitement que les variables sont du type correct et effectue des opérations d'égalité et de comparaison strictes. Cela permet d'éviter les vulnérabilités liées à la confusion de types, causées par des hypothèses du code de l'application sur le type d'une variable.	2
15.3.6	Vérifiez que le code JavaScript est écrit de manière à éviter la pollution du prototype, par exemple en utilisant Set() ou Map() au lieu de littéraux d'objet.	2
15.3.7	Vérifiez que l'application dispose de défenses contre les attaques de pollution des paramètres HTTP, en particulier si le framework d'application ne fait aucune distinction quant à la source des paramètres de requête (chaîne de requête, paramètres de corps, cookies ou champs d'en-tête).	2

V15.4 Concurrence sécurisée

Les problèmes de concurrence tels que les situations de concurrence, les vulnérabilités TOCTOU (Time-of-Check-to-Time-of-Use), les interblocages, les livelock, la privation de threads et les synchronisations incorrectes peuvent entraîner des comportements imprévisibles et des risques de sécurité. Cette section présente diverses techniques et stratégies pour atténuer ces risques.

#	Description	Niveau
15.4.1	Vérifiez que les objets partagés dans le code multithread (tels que les caches, les fichiers ou les objets en mémoire accessibles par plusieurs threads) sont accessibles en toute sécurité à l'aide de types thread-safe et de mécanismes de synchronisation tels que des verrous ou des sémaphores pour éviter les conditions de concurrence et la corruption des données.	3
15.4.2	Vérifiez que les vérifications de l'état d'une ressource, comme son existence ou ses autorisations, et des actions qui en dépendent, sont effectuées en une seule opération atomique afin d'éviter les situations de concurrence entre le moment de la vérification et	3

	le moment de l'utilisation (TOCTOU). Par exemple, vérifier l'existence d'un fichier avant de l'ouvrir ou l'accès d'un utilisateur avant de l'accorder.	
15.4.3	Vérifiez que les verrous sont utilisés de manière cohérente pour éviter que les threads ne se bloquent, que ce soit en s'attendant les uns les autres ou en réessayant sans fin, et que la logique de verrouillage reste dans le code responsable de la gestion de la ressource pour garantir que les verrous ne peuvent pas être modifiés par inadvertance ou de manière malveillante par des classes ou du code externes.	3
15.4.4	Vérifiez que les politiques d'allocation des ressources empêchent la saturation de threads en garantissant un accès équitable aux ressources, par exemple en exploitant les pools de threads, permettant aux threads de priorité inférieure de continuer dans un délai raisonnable.	3

Références

Pour plus d'informations, voir également :

- OWASP Prototype Pollution Prevention Cheat Sheet
- OWASP Mass Assignment Prevention Cheat Sheet
- OWASP CycloneDX Bill of Materials Specification
- OWASP Web Security Testing Guide: Testing for HTTP Parameter Pollution

V16 Journalisation de sécurité et gestion des erreurs

Objectif de contrôle

Les journaux de sécurité se distinguent des journaux d'erreurs ou de performances et servent à enregistrer les événements liés à la sécurité, tels que les décisions d'authentification, les décisions de contrôle d'accès et les tentatives de contournement des contrôles de sécurité, comme la validation des entrées ou la validation de la logique métier. Leur objectif est de faciliter la détection, la réponse et l'investigation en fournissant des données structurées et à haut niveau de signal aux outils d'analyse tels que les SIEM.

Les journaux ne doivent pas contenir de données personnelles sensibles, sauf obligation légale, et toutes les données enregistrées doivent être protégées comme des ressources de grande valeur. La journalisation ne doit pas compromettre la confidentialité ni la sécurité du système. Les applications doivent également être sécurisées, évitant toute divulgation ou interruption inutile.

Pour des conseils de mise en œuvre détaillés, consultez les aide-mémoire de l'OWASP indiquées dans la section Références.

V16.1 Documentation sur la journalisation de sécurité

Cette section garantit un inventaire clair et complet de la journalisation de l'ensemble de la pile applicative. Ceci est essentiel pour une surveillance efficace de la sécurité, une réponse aux incidents et une conformité optimale.

#	Description	Niveau
16.1.1	Vérifiez qu'il existe un inventaire documentant la journalisation effectuée à chaque couche de la pile technologique de l'application, les événements enregistrés, les formats de journal, l'endroit où cette journalisation est stockée, la manière dont elle est utilisée, la manière dont l'accès à celle-ci est contrôlé et la durée de conservation des journaux.	2

V16.2 Journalisation générale

La présente section énonce des exigences visant à garantir que les journaux de sécurité sont structurés de manière cohérente et contiennent les métadonnées attendues. L'objectif est de rendre les journaux lisibles par une machine et analysables par des systèmes et des outils distribués.

Les événements de sécurité impliquent souvent des données sensibles. Si ces données sont enregistrées sans surveillance, les journaux eux-mêmes deviennent classifiés et donc soumis à des exigences de chiffrement, à des politiques de conservation plus strictes et à une divulgation potentielle lors des audits.

Il est donc essentiel de ne loguer que ce qui est nécessaire et de traiter les données des journaux avec le même soin que les autres ressources sensibles.

Les exigences ci-dessous établissent les exigences fondamentales pour la journalisation des métadonnées, la synchronisation, le format et le contrôle.

#	Description	Niveau
16.2.1	Vérifiez que chaque entrée de journal inclut les métadonnées nécessaires (telles que quand, où, qui, quoi) qui permettraient une enquête détaillée sur la chronologie lorsqu'un événement se produit.	2
16.2.2	Vérifiez que les sources de temps de tous les composants de journalisation sont synchronisées et que les horodatages des métadonnées des événements de sécurité utilisent le temps UTC ou incluent un décalage horaire explicite. L'UTC est recommandé	2

	pour garantir la cohérence entre les systèmes distribués et éviter toute confusion lors du passage à l'heure d'été.	
16.2.3	Vérifiez que l'application stocke ou diffuse uniquement les journaux vers les fichiers et services documentés dans l'inventaire des journaux.	2
16.2.4	Vérifiez que les journaux peuvent être lus et corrélés par le processeur de journaux utilisé, de préférence en utilisant un format de journalisation commun.	2
16.2.5	Vérifiez que l'application applique la journalisation en fonction du niveau de protection des données lors de la journalisation de données sensibles. Par exemple, certaines données, comme les identifiants ou les informations de paiement, peuvent être interdites. D'autres données, comme les jetons de session, ne peuvent être enregistrées que par hachage ou masquage, en totalité ou en partie.	2

V16.3 Événements de sécurité

Cette section définit les exigences relatives à la journalisation des événements de sécurité au sein de Cette section définit les exigences relatives à la journalisation des événements de sécurité au sein de l'application. La capture de ces événements est essentielle pour détecter les comportements suspects, faciliter les enquêtes et respecter les obligations de conformité.

Cette section décrit les types d'événements à journaliser, sans toutefois prétendre à l'exhaustivité. Chaque application présente des facteurs de risque et un contexte opérationnel spécifiques.

Notez que si ASVS inclut la journalisation des événements de sécurité, les alertes et la corrélation (par exemple, les règles SIEM ou l'infrastructure de surveillance) sont considérées comme hors de portée et sont gérées par les systèmes opérationnels et de surveillance.

#	Description	Niveau
16.3.1	Vérifiez que toutes les opérations d'authentification sont enregistrées, y compris les tentatives réussies et infructueuses. Des métadonnées supplémentaires, telles que le type d'authentification ou les facteurs utilisés, doivent également être collectées.	2
16.3.2	Vérifier que les tentatives d'autorisation infructueuses sont consignées. Pour le niveau 3, cela doit inclure la journalisation de toutes les décisions d'autorisation, y compris lors de l'accès à des données sensibles (sans journalisation des données sensibles elles-mêmes).	2
16.3.3	Vérifiez que l'application enregistre les événements de sécurité définis dans la documentation et enregistre également les tentatives de contournement des contrôles de sécurité, tels que la validation des entrées, la logique métier et l'anti-automatisation.	2
16.3.4	Vérifiez que l'application enregistre les erreurs inattendues et les échecs de contrôle de sécurité tels que les échecs TLS du backend.	2

V16.4 Protection des journaux

Les journaux sont des artefacts d'investigation précieux et doivent être protégés. S'ils peuvent être facilement modifiés ou supprimés, ils perdent leur intégrité et deviennent peu fiables pour les enquêtes sur les incidents ou les procédures judiciaires. Ils peuvent exposer le comportement interne des applications ou des métadonnées sensibles, ce qui en fait une cible de choix pour les attaquants.

Cette section définit les exigences visant à garantir la protection des journaux contre les accès non autorisés, les falsifications et les divulgations, ainsi que leur transmission et leur stockage sécurisés dans des systèmes sécurisés et isolés.

#	Description	Niveau
16.4.1	Vérifiez que tous les composants de journalisation codent correctement les données pour empêcher l'injection de journaux.	2
16.4.2	Vérifiez que les journaux sont protégés contre tout accès non autorisé et ne peuvent pas être modifiés.	2
16.4.3	Vérifiez que les journaux sont transmis de manière sécurisée à un système logique distinct pour analyse, détection, alerte et remontée des informations. L'objectif est de garantir qu'en cas de violation de l'application, les journaux ne seront pas compromis.	2

V16.5 Gestion des erreurs

Cette section définit les exigences visant à garantir que les applications échouent proprement et sécurisée sans divulguer de détails internes sensibles.

#	Description	Niveau
16.5.1	Vérifiez qu'un message générique est renvoyé au consommateur lorsqu'une erreur inattendue ou sensible à la sécurité se produit, garantissant ainsi l'absence d'exposition de données système internes sensibles telles que les traces de pile, les requêtes, les clés secrètes et les jetons.	2
16.5.2	Vérifiez que l'application continue de fonctionner en toute sécurité lorsque l'accès aux ressources externes échoue, par exemple, en utilisant des modèles tels que des disjoncteurs ou une dégradation progressive.	2
16.5.3	Vérifiez que l'application échoue proprement et sécurisée, y compris lorsqu'une exception se produit, en évitant les accès en cas d'erreur telles que le traitement d'une transaction malgré les erreurs résultant de la logique de validation.	2
16.5.4	Vérifiez qu'un gestionnaire d'erreurs de « dernier recours » est défini pour intercepter toutes les exceptions non gérées. Cela permet d'éviter la perte des informations d'erreur qui doivent être loguées dans les fichiers journaux et de garantir qu'une erreur ne paralyse pas l'ensemble du processus applicatif, entraînant une perte de disponibilité.	3

Remarque : Certains langages (dont Swift, Go et, par des pratiques de conception courantes, de nombreux langages fonctionnels) ne prennent pas en charge les exceptions ni les gestionnaires d'événements de dernier recours. Dans ce cas, les architectes et les développeurs doivent utiliser une méthode compatible avec les modèles, les langages ou les Frameworks pour garantir que les applications peuvent gérer en toute sécurité les événements exceptionnels, inattendus ou liés à la sécurité.

Références

Pour plus d'informations, voir également :

- OWASP Web Security Testing Guide: Testing for Error Handling
- OWASP Authentication Cheat Sheet section about error messages
- OWASP Logging Cheat Sheet

- OWASP Application Logging Vocabulary Cheat Sheet

V17 WebRTC

Objectif de contrôle

La communication Web en temps réel (WebRTC) permet l'échange de voix, de vidéo et de données en temps réel dans les applications modernes. Avec l'adoption croissante de cette technologie, la sécurisation de l'infrastructure WebRTC devient cruciale. Cette section présente les exigences de sécurité pour les acteurs qui développent, hébergent ou intègrent des systèmes WebRTC.

Le marché du WebRTC peut être classé en trois segments :

1. Développeurs de produits : Fournisseurs propriétaires et open source qui créent et fournissent des produits et solutions WebRTC. Leur objectif est de développer des technologies WebRTC robustes et sécurisées, utilisables par d'autres.
2. Plateformes de communication en tant que service (CPaaS) : fournisseurs proposant des API, des SDK et l'infrastructure ou les plateformes nécessaires pour exploiter les fonctionnalités WebRTC. Les fournisseurs CPaaS peuvent utiliser des produits de la première catégorie ou développer leur propre logiciel WebRTC pour offrir ces services.
3. Fournisseurs de services : organisations qui exploitent les produits de développeurs ou de fournisseurs CPaaS, ou qui développent leurs propres solutions WebRTC. Elles créent et implémentent des applications pour les conférences en ligne, la santé, l'apprentissage en ligne et d'autres domaines où la communication en temps réel est essentielle.

Les exigences de sécurité décrites ici s'adressent principalement aux développeurs de produits, aux CPaaS et aux fournisseurs de services qui :

- utilisent des solutions open source pour développer leurs applications WebRTC ;
- utilisent des produits WebRTC commerciaux au sein de leur infrastructure ;
- utilisent des solutions WebRTC développées en interne ou intègrent divers composants dans une offre de services cohérente.

Il est important de noter que ces exigences de sécurité ne s'appliquent pas aux développeurs qui utilisent exclusivement les SDK et les API fournis par les fournisseurs CPaaS. Pour ces développeurs, les fournisseurs CPaaS sont généralement responsables de la plupart des problèmes de sécurité sous-jacents à leurs plateformes, et une norme de sécurité générique comme ASVS peut ne pas répondre pleinement à leurs besoins.

V17.1 Serveur TURN

Cette section définit les exigences de sécurité pour les systèmes utilisant leurs propres serveurs TURN (Traversal Using Relays around NAT). Les serveurs TURN contribuent au relais des médias dans les environnements réseau restrictifs, mais peuvent présenter des risques en cas de mauvaise configuration. Ces contrôles se concentrent sur le filtrage sécurisé des adresses et la protection contre l'épuisement des ressources.

#	Description	Niveau
17.1.1	Vérifiez que le service TURN (Traversal Using Relays around NAT) autorise uniquement l'accès aux adresses IP non réservées à des fins spécifiques (par exemple, réseaux internes, diffusion, bouclage). Notez que cela s'applique aux adresses IPv4 et IPv6.	2

#	Description	Niveau
17.1.2	Vérifiez que le service Traversal Using Relays around NAT (TURN) n'est pas susceptible d'épuiser ses ressources lorsque des utilisateurs légitimes tentent d'ouvrir un grand nombre de ports sur le serveur TURN.	3

V17.2 Média

Ces exigences s'appliquent uniquement aux systèmes hébergeant leurs propres serveurs multimédias WebRTC, tels que les unités de transfert sélectif (SFU), les unités de contrôle multipoint (MCU), les serveurs d'enregistrement ou les serveurs de passerelle. Les serveurs multimédias gèrent et distribuent les flux multimédias, ce qui rend leur sécurité essentielle pour protéger les communications entre homologues. La protection des flux multimédias est primordiale dans les applications WebRTC pour prévenir les écoutes clandestines, les falsifications et les attaques par déni de service susceptibles de compromettre la confidentialité des utilisateurs et la qualité des communications.

En particulier, il est nécessaire de mettre en œuvre des protections contre les attaques par saturation, telles que la limitation du débit, la validation des horodatages, l'utilisation d'horloges synchronisées pour correspondre aux intervalles en temps réel et la gestion des tampons afin d'éviter les débordements et de maintenir une synchronisation correcte. Si les paquets d'une session multimédia spécifique arrivent trop rapidement, les paquets excédentaires doivent être rejetés. Il est également important de protéger le système contre les paquets malformés en implémentant la validation des entrées, en gérant en toute sécurité les dépassements d'entiers, en prévenant les dépassements de tampon et en utilisant d'autres techniques robustes de gestion des erreurs.

Les systèmes qui s'appuient uniquement sur la communication multimédia peer-to-peer entre navigateurs Web, sans l'intervention de serveurs multimédias intermédiaires, sont exclus de ces exigences de sécurité spécifiques liées aux médias.

Cette section traite de l'utilisation de la sécurité de la couche de transport des datagrammes (DTLS) dans le contexte de WebRTC. L'exigence relative à la mise en place d'une politique documentée de gestion des clés cryptographiques est décrite dans le chapitre « Cryptographie ». Des informations sur les méthodes cryptographiques approuvées sont disponibles dans l'annexe Cryptographie de l'ASVS ou dans des documents tels que NIST SP 800-52 Rev. 2 ou BSI TR-02102-2 (version 2025-01).

#	Description	Niveau
17.2.1	Vérifiez que la clé du certificat Datagram Transport Layer Security (DTLS) est gérée et protégée conformément à la politique documentée de gestion des clés cryptographiques.	2
17.2.2	Vérifiez que le serveur multimédia est configuré pour utiliser et prendre en charge les suites de chiffrement DTLS (Datagram Transport Layer Security) approuvées et un profil de protection sécurisé pour l'extension DTLS pour l'établissement de clés pour le protocole de transport en temps réel sécurisé (DTLS-SRTP).	2
17.2.3	Vérifiez que l'authentification SRTP (Secure Real-time Transport Protocol) est vérifiée sur le serveur multimédia pour empêcher les attaques par injection RTP (Real-time Transport Protocol) d'entraîner une condition de déni de service ou l'insertion de médias audio ou vidéo dans les flux multimédias.	2
17.2.4	Vérifiez que le serveur multimédia est en mesure de continuer à traiter le trafic multimédia entrant lorsqu'il rencontre des paquets SRTP (Secure Real-time Transport Protocol) mal formés.	2

#	Description	Niveau
17.2.5	Vérifiez que le serveur multimédia est en mesure de continuer à traiter le trafic multimédia entrant pendant un flot de paquets SRTP (Secure Real-time Transport Protocol) provenant d'utilisateurs légitimes.	3
17.2.6	Vérifiez que le serveur multimédia n'est pas sensible à la vulnérabilité de condition de concurrence « ClientHello » dans Datagram Transport Layer Security (DTLS) en vérifiant si le serveur multimédia est publiquement connu comme étant vulnérable ou en effectuant le test de condition de concurrence.	3
17.2.7	Vérifiez que tous les mécanismes d'enregistrement audio ou vidéo associés au serveur multimédia sont en mesure de continuer à traiter le trafic multimédia entrant pendant un flot de paquets SRTP (Secure Real-time Transport Protocol) provenant d'utilisateurs légitimes.	3
17.2.8	Vérifiez que le certificat Datagram Transport Layer Security (DTLS) est vérifié par rapport à l'attribut d'empreinte cryptographique du protocole de description de session (SDP), en mettant fin au flux multimédia si la vérification échoue, pour garantir l'authenticité du flux multimédia.	3

V17.3 Signalisation

Cette section définit les exigences pour les systèmes qui exploitent leurs propres serveurs de signalisation WebRTC. La signalisation coordonne les communications « pair à pair » et doit être résiliente face aux attaques susceptibles de perturber l'établissement ou le contrôle des sessions.

Pour garantir une signalisation sécurisée, les systèmes doivent gérer correctement les entrées malformées et rester disponibles sous charge.

#	Description	Niveau
17.3.1	Vérifiez que le serveur de signalisation est capable de continuer à traiter les messages de signalisation entrants légitimes lors d'une attaque par saturation. Cela peut être réalisé en implémentant une limitation de débit au niveau de la signalisation.	2
17.3.2	Vérifiez que le serveur de signalisation est capable de continuer à traiter les messages de signalisation légitimes en cas de message mal formé susceptible de provoquer un déni de service. Cela peut inclure la validation des entrées, la gestion sécurisée des dépassements d'entiers, la prévention des dépassements de tampon et l'utilisation d'autres techniques robustes de gestion des erreurs.	2

Références

Pour plus d'informations, voir également :

- Le WebRTC DTLS ClientHello DoS est mieux documenté ici : Enable Security's blog post aimed at security professionals et ici white paper aimed at WebRTC developers
- RFC 3550 - RTP: A Transport Protocol for Real-Time Applications
- RFC 3711 - The Secure Real-time Transport Protocol (SRTP)
- RFC 5764 - Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP))
- RFC 8825 - Overview: Real-Time Protocols for Browser-Based Applications

- RFC 8826 - Security Considerations for WebRTC
- RFC 8827 - WebRTC Security Architecture
- DTLS-SRTP Protection Profiles

Annexe A : Glossaire

Annexe A : Glossaire

- **Architecture de sécurité** – 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 utilisateur et applicatives.
- **Address Space Layout Randomization (ASLR)** - Une technique pour rendre plus difficile l'exploitation des bugs de corruption de la mémoire.
- **Analyse de la composition logicielle** (SCA) – Ensemble de technologies conçues pour analyser la composition, les dépendances, les bibliothèques et les packages des applications afin de détecter les vulnérabilités de sécurité des versions de composants spécifiques utilisées. À ne pas confondre avec l'analyse du code source, communément appelée SAST.
- **Authentification** – Vérification de l'identité revendiquée par un utilisateur d'application.
- **Authentification client TLS**, également appelée **TLS mutuel** (mTLS) - Dans une connexion TLS standard, un client peut utiliser le certificat fourni par le serveur pour valider son identité. Lorsque l'authentification client TLS est utilisée, le client utilise également sa propre clé privée et son propre certificat pour permettre au serveur de valider également son identité.
- **Authentificateur à facteur unique** – Mécanisme permettant de vérifier l'authentification d'un utilisateur. Il peut s'agir d'un élément que vous connaissez (secrets mémorisés, mots de passe, phrases secrètes, codes PIN), d'une information que vous êtes (biométrie, empreinte digitale, reconnaissance faciale) ou d'un élément que vous possédez (jetons OTP, dispositif cryptographique tel qu'une carte à puce).
- **Authentification multifacteur** (MFA) - Authentification qui inclut au moins deux facteurs uniques.
- **Authentification par authentification unique** (SSO) – Elle se produit lorsqu'un utilisateur se connecte à une application et est ensuite automatiquement connecté à d'autres applications sans avoir à se réauthentifier. Par exemple, lorsqu'il se connecte à Google, il sera automatiquement connecté à d'autres services Google tels que YouTube, Google Docs et Gmail.
- **Bill of Materials logiciels** (SBOM) – Liste structurée et complète de tous les composants, modules, bibliothèques, Framework et autres ressources nécessaires à la création ou à l'assemblage d'une application logicielle.
- **Certificat X.509** – Un certificat X.509 est un certificat numérique qui utilise la norme internationale d'infrastructure à clés publiques (PKI) X.509, largement acceptée, pour vérifier qu'une clé publique appartient à l'identité de l'utilisateur, de l'ordinateur ou du service contenue dans le certificat.
- **Code d'authentification de message** (MAC) - Somme de contrôle cryptographique des données, calculée par un algorithme de génération de MAC, utilisée pour garantir leur intégrité et leur authenticité.
- **Communication Web en temps réel** (WebRTC) – Pile de protocoles et API web associée, utilisées pour le transport de flux multimédias dans les applications web, généralement dans le cadre de téléconférences. Basée sur SRTP, SRTCP, DTLS, SDP et STUN/TURN.
- **Configuration de sécurité** – Configuration d'exécution d'une application qui affecte l'utilisation des contrôles de sécurité.
- **Contrôle de sécurité** – Fonction ou composant qui effectue un contrôle de sécurité (par exemple, un contrôle d'autorisation) ou, lorsqu'il est appelé, produit un effet de sécurité (par exemple, la génération d'un enregistrement d'audit).
- **Composant** – Unité de code autonome, avec des interfaces disque et réseau associées, qui communique avec d'autres composants.
- **Couche de service sécurisé** – Tout point d'application de contrôle sécurisé, tel qu'un microservice, une API sans serveur, côté serveur, une API sécurisée sur un appareil client doté d'un démarrage sécurisé,

des API partenaires ou externes, etc. Le terme « sécurisé » signifie qu'il n'y a aucun risque qu'un utilisateur non sécurisé puisse contourner ou ignorer la couche ou les contrôles implémentés à cette couche.

- **Cross-Site Scripting** (XSS) – Vulnérabilité de sécurité généralement présente dans les applications web, permettant l'injection de scripts côté client dans le contenu.
- **Cycle de vie du développement logiciel** (SDLC) – Processus étape par étape de développement d'un logiciel, depuis les exigences initiales jusqu'au déploiement et à la maintenance.
- **Datagram Transport Layer Security** (DTLS) – Protocole cryptographique assurant la sécurité des communications sur une connexion réseau. Basé sur le protocole TLS, il est adapté à la protection des protocoles orientés datagrammes (généralement via UDP). Défini dans la RFC 9147 pour DTLS 1.3.
- **Délai d'inactivité** – Durée pendant laquelle une session peut rester active en l'absence d'interaction de l'utilisateur avec l'application. Il s'agit d'un élément de l'expiration de la session.
- **Durée de vie maximale absolue d'une session** – Également appelée « délai d'expiration global » par le NIST, il s'agit de la durée maximale pendant laquelle une session peut rester active après authentification, quelle que soit l'interaction de l'utilisateur. Il s'agit d'un élément de l'expiration de la session.
- **Entité externe XML** (XXE) – Type d'entité XML permettant d'accéder à du contenu local ou distant via un identifiant système déclaré. Cela peut conduire à diverses attaques par injection.
- **Énumération des faiblesses courantes** (CWE) – Liste communautaire des faiblesses courantes en matière de sécurité logicielle. Elle sert de langage commun, de critère de mesure pour les outils de sécurité logicielle et de référence pour l'identification, l'atténuation et la prévention des faiblesses.
- **Environnement d'exécution sécurisé** (TEE) - Un environnement de traitement isolé dans lequel les applications peuvent être exécutées en toute sécurité, indépendamment du reste du système.
- **Extension de sécurité de la couche de transport de datagrammes pour l'établissement de clés pour le protocole de transport sécurisé en temps réel** (DTLS-SRTP) – Mécanisme d'utilisation d'une négociation DTLS pour l'établissement des clés d'une session SRTP. Définie dans la RFC 5764.
- **Falsification de requête côté serveur** (SSRF) – Attaque qui exploite les fonctionnalités du serveur pour lire ou mettre à jour des ressources internes. L'attaquant fournit ou modifie une URL, que le code exécuté sur le serveur lit ou à laquelle il soumet des données.
- **Fast IDentity Online** (FIDO) – Ensemble de normes d'authentification permettant l'utilisation de différentes méthodes d'authentification, notamment la biométrie, les modules de plateforme sécurisée (TPM), les jetons de sécurité USB, etc.
- **Fonction de dérivation de clé basée sur un mot de passe 2** (PBKDF2) – Algorithme unidirectionnel spécial utilisé pour créer une clé cryptographique forte à partir d'un texte d'entrée (tel qu'un mot de passe) et d'une valeur de sel aléatoire supplémentaire. Il peut donc être utilisé pour rendre plus difficile le forçage d'un mot de passe hors ligne si la valeur résultante est stockée à la place du mot de passe d'origine.
- **Fournisseur d'identité** (IdP) – Également appelé fournisseur de services d'identification (CSP) dans les références NIST. Entité qui fournit une source d'authentification pour d'autres applications.
- **Fournisseur de services d'identification** (CSP) – Également appelé fournisseur d'identité (IdP). Source de données utilisateur pouvant être utilisée comme source d'authentification par d'autres applications.
- **Générateur de nombres pseudo-aléatoires cryptographiquement sécurisé** (CSPRNG) - Un générateur de nombres pseudo-aléatoires avec des propriétés qui le rendent adapté à une utilisation en cryptographie, également appelé générateur de nombres aléatoires cryptographiques (CRNG).
- **Gestion des informations et des événements de sécurité** (SIEM) - Système de détection des menaces, de conformité et de gestion des incidents de sécurité par la collecte et l'analyse de données de sécurité provenant de diverses sources au sein de l'infrastructure informatique d'une organisation.

- **Graphiques vectoriels évolutifs** (SVG) – Langage de balisage XML pour la description de graphiques vectoriels bidimensionnels.
- **Hardware Security Module** (HSM) – Composant matériel qui stocke les clés cryptographiques et autres secrets de manière protégée.
- **Hibernate Query Language** (HQL) – Langage de requête similaire au SQL utilisé par la bibliothèque Hibernate ORM.
- **HTTP Strict Transport Security** (HSTS) – Politique qui indique au navigateur de se connecter uniquement au domaine renvoyant l'en-tête via TLS et lorsqu'un certificat valide est présenté. Elle est activée via le champ d'en-tête de réponse Strict-Transport-Security.
- **HyperText Transfer Protocol** (HTTP) – Protocole d'application pour les systèmes d'information hypermédia distribués et collaboratifs. Il constitue le fondement de la communication de données sur le World Wide Web.
- **Identifiant de ressource uniforme** (URI) – Chaîne de caractères unique identifiant une ressource, telle qu'une page web, une adresse e-mail, des lieux.
- **Identifiant de session** ou **ID de session** – Clé identifiant une session avec état stockée dans le backend. Transférée vers et depuis le client, soit sous forme de jeton de référence, soit à l'intérieur d'un jeton de référence.
- **Identifiant unique universel** (UUID) – Un numéro de référence unique utilisé comme identifiant dans un logiciel.
- **Inclusion de fichiers distants** (RFI) – Attaque exploitant des procédures d'inclusion vulnérables dans l'application, entraînant l'inclusion de fichiers distants.
- **Inclusion de fichiers locaux** (LFI) – Attaque exploitant les procédures d'inclusion de fichiers vulnérables d'une application, entraînant l'inclusion de fichiers locaux déjà présents sur le serveur.
- **Inclusion de script intersite** (XSSI) – Variante d'une attaque de script intersite (XSS) dans laquelle une application web récupère du code malveillant depuis une ressource externe et l'intègre à son propre contenu.
- **Infrastructure à clés publiques** (PKI) – Dispositif qui lie les clés publiques aux identités respectives des entités. Cette liaison est établie par un processus d'enregistrement et de délivrance de certificats auprès et par une autorité de certification (AC).
- **Injection SQL** (SQLi) - Technique d'injection de code utilisée pour attaquer les applications pilotées par les données, dans laquelle des instructions SQL malveillantes sont insérées dans un point d'entrée.
- **Jeton anti-falsification** – Mécanisme par lequel un ou plusieurs jetons sont transmis dans une requête et validés par le serveur d'app.
- **Code d'authentification de message** (MAC) - Somme de contrôle cryptographique des données, calculée par un algorithme de génération de MAC, utilisée pour garantir leur intégrité et leur authenticité.
- **Jeton autonome** – Jeton qui encapsule un ou plusieurs attributs indépendants de l'état côté serveur ou d'un autre stockage externe. Ces jetons garantissent l'authenticité et l'intégrité des attributs qu'ils contiennent, permettant un échange d'informations sécurisé et « sans état » entre les systèmes. Les jetons autonomes sont généralement sécurisés à l'aide de techniques cryptographiques, telles que les signatures numériques ou les codes d'authentification de messages (MAC), afin de garantir l'authenticité, l'intégrité et, dans certains cas, la confidentialité de leurs données. Les assertions SAML et les JWT en sont des exemples courants.
- **Jeton de référence** – Type de jeton servant de pointeur ou d'identifiant vers un état ou des métadonnées stockées sur un serveur, parfois appelés jetons aléatoires ou jetons opaques. Contrairement aux jetons autonomes, qui intègrent certaines de leurs données pertinentes au sein même du jeton, les jetons de référence ne contiennent aucune information intrinsèque et dépendent du serveur pour le contexte. Le jeton de référence sera ou contiendra un identifiant de session.

- **Jeton de session** – Expression générique utilisée dans la présente norme pour désigner le jeton ou la valeur utilisé(e) dans les mécanismes de session sans état (qui utilisent un jeton autonome) ou dans les mécanismes de session avec état (qui utilisent un jeton de référence).
- **Jeton Web JSON** (JWT) – La RFC 7519 définit une norme pour un objet de données JSON composé d'une section d'en-tête expliquant comment valider l'objet, d'une section de corps contenant un ensemble de revendications et d'une section de signature contenant une signature numérique permettant de valider le contenu de la section de corps. Il s'agit d'un type de jeton autonome.
- **Liste d'autorisation** – Liste de données ou d'opérations autorisées, par exemple une liste de caractères autorisés à valider les entrées.
- **Localisateur de ressources uniforme** (URL) – Chaîne spécifiant l'emplacement d'une ressource sur Internet.
- **Malware** - Code introduit dans une application pendant son développement, à l'insu de son propriétaire, qui contourne la politique de sécurité prévue de l'application. Différent des logiciels malveillants tels qu'un virus ou un ver !
- **Mappage objet-relationnel** (ORM) - Système permettant de référencer et d'interroger une base de données relationnelle/table au sein d'un programme d'application à l'aide d'un modèle objet compatible avec l'application.
- **Mécanisme de session avec état** - Dans un mécanisme de session avec état, l'application conserve l'état de la session dans le backend, ce qui correspond généralement à un jeton de session, généré à l'aide d'un générateur de nombres pseudo-aléatoires cryptographiquement sécurisé (CSPRNG), délivré à l'utilisateur final.
- **Mécanisme de session sans état** - Un mécanisme de session sans état utilise un jeton autonome transmis aux clients et contenant des informations de session qui ne sont pas nécessairement stockées au sein du service, qui reçoit et valide ensuite le jeton. En réalité, un service doit avoir accès à certaines informations de session (comme une liste de révocation JWT) pour pouvoir appliquer les contrôles de sécurité requis.
- **Module cryptographique** – Matériel, logiciel et/ou micrologiciel implémentant des algorithmes cryptographiques et/ou générant des clés cryptographiques.
- **Module de plateforme sécurisé** (TPM) – Type de HSM généralement rattaché à un composant matériel plus important, comme une carte mère, et servant de « racine de confiance » pour ce système.
- **Mot de passe à usage unique** (OTP) - Mot de passe généré de manière unique pour une utilisation unique.
- **Open Worldwide Application Security Project** (OWASP) – L'OWASP (Open Worldwide Application Security Project) est une communauté mondiale, libre et ouverte, qui se consacre à l'amélioration de la sécurité des logiciels applicatifs. Notre mission est de rendre la sécurité des applications visible, afin que les personnes et les organisations puissent prendre des décisions éclairées face aux risques de sécurité applicative. Voir : https://www.owasp.org/.
- **OTP basé sur l'heure** - Méthode de génération d'un OTP où l'heure actuelle fait partie de l'algorithme de génération du mot de passe.
- **Partie de confiance** (RP) – Généralement une application qui s'appuie sur l'authentification d'un utilisateur auprès d'un fournisseur d'authentification distinct. L'application s'appuie sur un jeton ou un ensemble d'assertions signées fournies par ce fournisseur d'authentification pour garantir que l'utilisateur est bien celui qu'il prétend être.
- **Protocole de description de session** (SDP) - Format de message permettant de configurer une session multimédia (utilisé par exemple dans WebRTC). Défini dans la RFC 4566.
- **Protocole de transport en temps réel** (RTP) et **Protocole de contrôle de transport en temps réel** (RTCP) - Deux protocoles utilisés conjointement pour le transport de flux multimédia. Utilisé par la pile WebRTC. Défini dans la RFC 3550.

- **Protocole de transfert hypertexte sur SSL/TLS** (HTTPS) – Méthode de sécurisation des communications HTTP par chiffrement à l'aide du protocole TLS (Transport Layer Security).
- **Rapport de vérification de la sécurité des applications** – Rapport documentant les résultats globaux et l'analyse justificative produite par le vérificateur pour une application donnée.
- **Réseau téléphonique public commuté** (RTPC) – Réseau téléphonique traditionnel qui comprend à la fois les téléphones fixes et les téléphones mobiles.
- **Secure Real-time Transport Protocol** (SRTP) et **Secure Real-time Transport Control Protocol** (SRTCP) – Profil des protocoles RTP et RTCP prenant en charge le chiffrement des messages, l'authentification et la protection de leur intégrité. Défini dans la RFC 3711.
- **Security Assertion Markup Language** (SAML) – Une norme ouverte pour l'authentification unique basée sur la transmission d'assertions signées (généralement des objets XML) entre le fournisseur d'identité et la partie utilisatrice.
- **Sécurité des applications** – La sécurité au niveau des applications se concentre sur l'analyse des composants de la couche applicative du modèle de référence d'interconnexion des systèmes ouverts (OSI), plutôt que sur le système d'exploitation sous-jacent ou les réseaux connectés, par exemple.
- **Test boîte noire** – Méthode de test logiciel qui examine les fonctionnalités d'une application sans scruter ses structures ou son fonctionnement internes.
- **Tests dynamiques de sécurité des applications** (DAST) – Technologies conçues pour détecter les conditions indiquant une vulnérabilité de sécurité dans une application en cours d'exécution.
- **Time-of-check to time-of-use** (TOCTOU) - Situation dans laquelle une application vérifie l'état d'une ressource avant de l'utiliser, mais cet état peut être modifié entre la vérification et l'utilisation. Cela peut invalider les résultats de la vérification et entraîner des actions non valides de l'application en raison de cette incohérence d'état.
- **Time based One-time Passwords** (TOTPs) - Méthode de génération d'un OTP dans laquelle l'heure actuelle fait partie de l'algorithme de génération du mot de passe.
- **Transport Layer Security** (TLS) - Protocoles cryptographiques qui assurent la sécurité des communications sur une connexion réseau.
- **Traversée utilisant des relais autour de NAT** (TURN) – Extension du protocole STUN utilisant un serveur TURN comme relais lorsque les connexions « pair à pair » directes ne peuvent être établies. Défini dans la RFC 8656.
- **TLS mutuel** (mTLS) - Voir authentification client TLS.
- **Utilitaires de traversée de session pour NAT** (STUN) - Protocole utilisé pour faciliter la traversée NAT afin d'établir des communications « pair à pair ». Défini dans la RFC 3489.
- **Validation des entrées** - Canonisation et validation des entrées utilisateur non fiables.
- **Vérificateur** – Personne ou équipe qui examine une demande par rapport aux exigences ASVS de l'OWASP.
- **Vérification automatisée** – Utilisation d'outils automatisés (analyse dynamique, analyse statique, ou les deux) qui utilisent les signatures de vulnérabilité pour détecter les problèmes.
- **Vérification de la conception** – Évaluation technique de l'architecture de sécurité d'une application.
- **Vérification de la sécurité des applications** – Évaluation technique d'une application par rapport à l'ASVS de l'OWASP.
- **Vérification dynamique** – Utilisation d'outils automatisés utilisant les signatures de vulnérabilité pour détecter les problèmes lors de l'exécution d'une application.
- **WebSocket sur TLS** (WSS) – Pratique de sécurisation des communications WebSocket en superposant WebSocket sur le protocole TLS.

- **What You See Is What You Get** (WYSIWYG) – Éditeur de contenu riche qui montre l'apparence réelle du contenu une fois rendu, plutôt que le codage utilisé pour gérer le rendu.

Annexe B : Références

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

Projets principaux de l'OWASP

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

Projet de la série de fiches de triche de l'OWASP

Ce projet contient plusieurs aide-mémoires qui seront pertinents pour différents sujets de l'ASVS.

Il existe un mappage vers l'ASVS qui peut être trouvé ici : https://cheatsheetseries.owasp.org/IndexASVS.html

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 and Mobile Application Security Verification Standard : https://owasp.org/www-project-mobile-security-testing-guide/

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

1. OWASP Internet of Things Project : https://owasp.org/www-project-internet-of-things/

Projets sans serveur OWASP

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

Autres

De même, les sites Web suivants sont susceptibles d'être utiles aux utilisateurs/adoptants de cette norme

1. SecLists Github : https://github.com/danielmiessler/SecLists
2. MITRE Common Weakness Enumeration : https://cwe.mitre.org/
3. PCI Security Standards Council : https://www.pcisecuritystandards.org/
4. PCI Data Security Standard (DSS) v3.2.1 Requirements and Security Assessment Procedures : https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf
5. PCI Software Security Framework - Secure Software Requirements and Assessment Procedures : https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf
6. PCI Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures : https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf

7. OWASP ASVS 4.0 Testing Guide : https://github.com/BlazingWind/OWASP-ASVS-4.0-testing-guide

Annexe C : Standards cryptographiques

Le chapitre « Cryptographie » va au-delà de la simple définition des bonnes pratiques. Il vise à améliorer la compréhension des principes de cryptographie et à encourager l'adoption de méthodes de sécurité plus résilientes et modernes. Cette annexe fournit des informations techniques détaillées sur chaque exigence, en complément des normes générales décrites dans le chapitre « Cryptographie ».

Cette annexe définit le niveau d'approbation des différents mécanismes cryptographiques :

- Les mécanismes *Approuvés* (A) peuvent être utilisés dans les applications.
- Les mécanismes *Hérités* (L) ne devraient pas être utilisés dans les applications, mais peuvent néanmoins être utilisés uniquement pour assurer la compatibilité avec les applications ou le code existants. Bien que l'utilisation de ces mécanismes ne soit pas actuellement considérée comme une vulnérabilité en soi, ils devraient être remplacés par des mécanismes plus sûrs et évolutifs dès que possible.
- Les mécanismes *Interdits* (D) ne devraient pas être utilisés car ils sont actuellement considérés comme cassés ou n'offrent pas une sécurité suffisante.

Cette liste peut être modifiée dans le contexte d'une application donnée pour diverses raisons, notamment :

- nouvelles évolutions dans le domaine de la cryptographie ;
- conformité à une réglementation.

Inventaire et documentation cryptographiques

Cette section fournit des informations complémentaires sur l'inventaire et la documentation cryptographiques V11.1.

Il est important de s'assurer que tous les actifs cryptographiques, tels que les algorithmes, les clés et les certificats, sont régulièrement découverts, inventoriés et évalués. Pour le niveau 3, cela devrait inclure l'utilisation d'analyses statiques et dynamiques pour découvrir l'utilisation de la cryptographie dans une application. Des outils tels que les SAST et DAST peuvent être utiles, mais des outils dédiés pourraient être nécessaires pour une couverture plus complète. Voici quelques exemples d'outils gratuits :

- CryptoMon - Moniteur de cryptographie réseau - utilisant eBPF, écrit en python
- Cryptobom Forge Tool: Generating Comprehensive CBOMs from CodeQL Outputs

Forces équivalentes des paramètres cryptographiques

Les niveaux de sécurité relatifs des différents systèmes cryptographiques sont présentés dans ce tableau (extrait de NIST SP 800-57 Partie 1, p.71) :

Force de sécurité	Algorithmes à clés symétriques	Corps finis	Factorisation d'entiers	Courbe elliptique
<= 80	2TDEA	L = 1024 N = 160	k = 1024	f = 160-223
112	3TDEA	L = 2048 N = 224	k = 2048	f = 224-255
128	AES-128	L = 3072 N = 256	k = 3072	f = 256-383

Force de sécurité	Algorithmes à clés symétriques	Corps finis	Factorisation d'entiers	Courbe elliptique
192	AES-192	L = 7680 N = 384	k = 7680	f = 384-511
256	AES-256	L = 15360 N = 512	k = 15360	f = 512+

Exemples d'applications :

- Cryptographie à corps finis : DSA, FFDH, MQV
- Cryptographie à factorisation entière : RSA
- Cryptographie à courbes elliptiques : ECDSA, EdDSA, ECDH, MQV

Remarque : cette section suppose qu'aucun ordinateur quantique n'existe ; si un tel ordinateur existait, les estimations des trois dernières colonnes ne seraient plus valides.

Valeurs aléatoires

Cette section fournit des informations supplémentaires sur les valeurs aléatoires de la version 11.5.

Nom	Version/Référence	Notes	Statut
/dev/random	Linux 4.8+ (oct. 2016), également présent dans iOS, Android et d'autres systèmes d'exploitation POSIX basés sur Linux. Basé sur RFC7539	Utilise le flux ChaCha20. Trouvé dans iOS SecRandomCopyBytes et Android Secure Random avec les paramètres corrects fournis pour chacun.	A
/dev/urandom	Fichier spécial du noyau Linux pour fournir des données aléatoires	Fournit des sources d'entropie de haute qualité à partir du caractère aléatoire matériel	A
AES-CTR-DRBG	NIST SP800-90A	Tel qu'utilisé dans les implémentations courantes, telles que l'API Windows CNG BCryptGenRandom définie par BCRYPT_RNG_ALGORITHM.	A
HMAC-DRBG	NIST SP800-90A		A
Hash-DRBG	NIST SP800-90A		A
getentropy()	OpenBSD, disponible sous Linux glibc 2.25+ et macOS 10.12+	Fournit des octets aléatoires sécurisés directement depuis la source d'entropie du noyau, grâce à une API simple et minimale. Plus moderne, elle évite les pièges des anciennes API.	A

La fonction de hachage sous-jacente utilisée avec HMAC-DRBG ou Hash-DRBG doit être approuvée pour cette utilisation.

Algorithmes de chiffrement

Cette section fournit des informations supplémentaires sur les algorithmes de chiffrement V11.3.

Les algorithmes de chiffrement approuvés sont classés par ordre de préférence.

Algorithmes à clé symétrique	Référence	Statut
AES-256	FIPS 197	A
Salsa20	Salsa 20 specification	A
XChaCha20	XChaCha20 Draft	A
XSalsa20	Extending the Salsa20 nonce	A
ChaCha20	RFC 8439	A
AES-192	FIPS 197	A
AES-128	FIPS 197	L
2TDEA		D
TDEA (3DES/3DEA)		D
IDEA		D
RC4		D
Blowfish		D
ARC4		D
DES		D

Modes de chiffrement AES

Les chiffrements par blocs, comme AES, peuvent être utilisés avec différents modes opératoires. De nombreux modes, comme le livre de codes électronique (ECB), ne sont pas sécurisés et ne doivent pas être utilisés. Les modes Galois/Compteur (GCM) et Compteur avec chaînage de blocs et code d'authentification de message (CCM) assurent un chiffrement authentifié et doivent être utilisés dans les applications modernes.

Les modes approuvés sont classés par ordre de préférence.

Mode	Authentifié	Référence	Statut	Restriction
GCM	Oui	NIST SP 800-38D	A	
CCM	Oui	NIST SP 800-38C	A	
CBC	Non	NIST SP 800-38A	L	
CCM-8	Oui		D	
ECB	Non		D	
CFB	Non		D	

Mode	Authentifié	Référence	Statut	Restriction
OFB	Non		D	
CTR	Non		D	

Remarques :

- Tous les messages chiffrés doivent être authentifiés. Toute utilisation du mode CBC nécessite l'association d'un algorithme MAC de hachage pour valider le message. En général, cet algorithme doit être appliqué à la méthode « Chiffrer puis hacher » (mais TLS 1.2 utilise plutôt la méthode « Hash puis chiffrer »). Si cela ne peut être garanti, le mode CBC NE DOIT PAS être utilisé. La seule application où le chiffrement sans algorithme MAC est autorisé est le chiffrement de disque.
- Si le mode CBC est utilisé, la vérification du remplissage doit être garantie en temps constant.
- Lors de l'utilisation de CCM-8, la balise MAC ne dispose que de 64 bits de sécurité. Ceci n'est pas conforme à l'exigence 6.2.9 qui exige au moins 128 bits de sécurité.
- Le chiffrement de disque est considéré comme hors du champ d'application de l'ASVS. Par conséquent, cette annexe ne répertorie aucune méthode approuvée pour le chiffrement de disque. Pour cette utilisation, le chiffrement sans authentification est généralement accepté et les modes XTS, XEX et LRW sont généralement utilisés.

key wrapping

L'encapsulation (et le déchiffrement) d'une clé cryptographique est une méthode de protection d'une clé existante par encapsulation (c'est-à-dire par encapsulation) grâce à un mécanisme de chiffrement supplémentaire, afin que la clé d'origine ne soit pas exposée de manière visible, par exemple lors d'un transfert. Cette clé supplémentaire, utilisée pour protéger la clé d'origine, est appelée clé d'encapsulation.

Cette opération peut être effectuée lorsqu'il est souhaitable de protéger des clés dans des emplacements jugés non fiables, ou d'envoyer des clés sensibles sur des réseaux non fiables ou au sein d'applications.

Cependant, il est important de bien comprendre la nature (par exemple, l'identité et la finalité) de la clé d'origine avant de s'engager dans une procédure d'encapsulation/déchiffrement, car cela peut avoir des répercussions sur les systèmes/applications sources et cibles en termes de sécurité, et notamment de conformité, ce qui peut inclure des pistes d'audit de la fonction d'une clé (par exemple, la signature) ainsi qu'un stockage approprié des clés.

Plus précisément, AES-256 doit être utilisé pour l'encapsulation des clés, conformément à la norme NIST SP 800-38F (https://csrc.nist.gov/pubs/sp/800/38/f/final) et en tenant compte des dispositions prospectives contre la menace quantique. Les modes de chiffrement utilisant AES sont les suivants, par ordre de préférence :

Enveloppement de clé	Référence	Statut
KW	NIST SP 800-38F	A
KWP	NIST SP 800-38F	A

AES-192 et AES-128 PEUVENT être utilisés si le cas d'utilisation l'exige, mais leur motivation DOIT être documentée dans l'inventaire cryptographique de l'entité.

Chiffrement authentifié

À l'exception du chiffrement du disque, les données chiffrées doivent être protégées contre toute modification non autorisée à l'aide d'un schéma de chiffrement authentifié (AE), généralement un schéma de chiffrement authentifié avec données associées (AEAD).

L'application doit de préférence utiliser un schéma AEAD approuvé. Elle peut également combiner un schéma de chiffrement approuvé et un algorithme MAC approuvé avec une construction « Chiffrer puis MAC ».

La méthode « MAC puis chiffrer » est toujours autorisée pour des raisons de compatibilité avec les applications existantes. Elle est utilisée dans TLS v1.2 avec les anciennes suites de chiffrement.

Mécanisme AEAD	Référence	Statut
AES-GCM	SP 800-38D	A
AES-CCM	SP 800-38C	A
ChaCha-Poly1305	RFC 7539	A
AEGIS-256	AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)	A
AEGIS-128	AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)	A
AEGIS-128L	AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)	A
Encrypt-then-MAC		A
MAC-then-encrypt		L

Fonctions de hachage

Cette section fournit des informations supplémentaires sur le hachage V11.4 et les fonctions basées sur le hachage.

Fonctions de hachage pour les cas d'utilisation généraux

Le tableau suivant répertorie les fonctions de hachage approuvées pour les cas d'utilisation cryptographiques généraux, tels que les signatures numériques :

- Les fonctions de hachage approuvées offrent une forte résistance aux collisions et conviennent aux applications de haute sécurité.
- Certains de ces algorithmes offrent une forte résistance aux attaques lorsqu'ils sont utilisés avec une gestion appropriée des clés cryptographiques. Ils sont donc également approuvés pour les fonctions HMAC, KDF et RBG.
- Les fonctions de hachage dont le résultat est inférieur à 254 bits présentent une résistance aux collisions insuffisante et ne doivent pas être utilisées pour la signature numérique ou d'autres applications nécessitant une résistance aux collisions. Pour d'autres utilisations, elles peuvent être utilisées à des fins de compatibilité et de vérification UNIQUEMENT avec les systèmes existants, mais ne doivent pas être utilisées dans les nouvelles conceptions.

Fonction de hachage	Référence	Statut	Restrictions
SHA3-512	FIPS 202	A	
SHA-512	FIPS 180-4	A	
SHA3-384	FIPS 202	A	
SHA-384	FIPS 180-4	A	
SHA3-256	FIPS 202	A	
SHA-512/256	FIPS 180-4	A	
SHA-256	FIPS 180-4	A	
SHAKE256	FIPS 202	A	
BLAKE2s	BLAKE2: simpler, smaller, fast as MD5	A	
BLAKE2b	BLAKE2: simpler, smaller, fast as MD5	A	
BLAKE3	BLAKE3 one function, fast everywhere	A	
SHA-224	FIPS 180-4	L	Ne convient pas aux signatures numériques HMAC, KDF, RBG
SHA-512/224	FIPS 180-4	L	Ne convient pas aux signatures numériques HMAC, KDF, RBG
SHA3-224	FIPS 202	L	Ne convient pas aux signatures numériques HMAC, KDF, RBG
SHA-1	RFC 3174 et RFC 6194	L	Ne convient pas aux signatures numériques HMAC, KDF, RBG
CRC (toute longueur)		D	
MD4	RFC 1320	D	
MD5	RFC 1321	D	

Fonctions de hachage pour le stockage des mots de passe

Pour un hachage sécurisé des mots de passe, des fonctions de hachage dédiées doivent être utilisées. Ces algorithmes de hachage lent atténuent les attaques par force brute et par dictionnaire en augmentant la difficulté de calcul du craquage des mots de passe.

KDF	Référence	Paramètres requis	Statut
argon2id	RFC 9106	t = 1: m ≥ 47104 (46 MiB), p = 1t = 2: m ≥ 19456 (19 MiB), p = 1t ≥ 3: m ≥ 12288 (12 MiB), p = 1	A

KDF	Référence	Paramètres requis	Statut
scrypt	RFC 7914	p = 1: N ≥ 2^17 (128 MiB), r = 8p = 2: N ≥ 2^16 (64 MiB), r = 8p ≥ 3: N ≥ 2^15 (32 MiB), r = 8	A
bcrypt	A Future-Adaptable Password Scheme	coût ≥ 10	A
PBKDF2-HMAC-SHA-512	NIST SP 800-132, FIPS 180-4	itérations ≥ 210,000	A
PBKDF2-HMAC-SHA-256	NIST SP 800-132, FIPS 180-4	itérations ≥ 600,000	A
PBKDF2-HMAC-SHA-1	NIST SP 800-132, FIPS 180-4	itérations ≥ 1,300,000	L

Les fonctions de dérivation de clés basées sur des mots de passe approuvées peuvent être utilisées pour le stockage des mots de passe.

Mécanismes d'échange de clés

Cette section fournit des informations complémentaires sur la cryptographie à clé publique V11.6.

Systèmes KEX

Une force de sécurité de 112 bits ou plus DOIT être garantie pour tous les méthodes d'échange de clés, et leur mise en œuvre DOIT suivre les choix de paramètres dans le tableau suivant.

Schéma	Paramètres du domaine	Confidentialité persistante	Statut
Diffie-Hellman à corps finis (FFDH)	L >= 3072 et N >= 256	Oui	A
Courbe elliptique Diffie-Hellman (ECDH)	f >= 256-383	Oui	A
Transport de clés chiffrées avec RSA-PKCS#1 v1.5		Non	D

Où les paramètres suivants sont :

- k est la taille de la clé RSA.
- L est la taille de la clé publique et N est la taille de la clé privée pour la cryptographie à corps finis.
- f est la plage de tailles de clés pour ECC.

Toute nouvelle implémentation NE DOIT PAS utiliser de schéma non conforme aux normes NIST SP 800-56A et B et NIST SP 800-77. Plus précisément, IKEv1 NE DOIT PAS être utilisé en production.

Groupes Diffie-Hellman

Les groupes suivants sont approuvés pour la mise en œuvre de l'échange de clés Diffie-Hellman. Les niveaux de sécurité sont décrits dans la norme NIST SP 800-56A, l'annexe D et la norme NIST SP 800-57, partie 1, révision 5.

Groupe	Statut
P-224, secp224r1	A
P-256, secp256r1	A
P-384, secp384r1	A
P-521, secp521r1	A
K-233, sect233k1	A
K-283, sect283k1	A
K-409, sect409k1	A
K-571, sect571k1	A
B-233, sect233r1	A
B-283, sect283r1	A
B-409, sect409r1	A
B-571, sect571r1	A
Curve448	A
Curve25519	A
MODP-2048	A
MODP-3072	A
MODP-4096	A
MODP-6144	A
MODP-8192	A
ffdhe2048	A
ffdhe3072	A
ffdhe4096	A
ffdhe6144	A
ffdhe8192	A

Codes d'authentification des messages (MAC)

Les codes d'authentification de message (MAC) sont des structures cryptographiques permettant de vérifier l'intégrité et l'authenticité d'un message. Un MAC prend en entrée un message et une clé secrète et génère une étiquette de taille fixe (la valeur MAC). Les MAC sont largement utilisés dans les protocoles de communication sécurisés (par exemple, TLS/SSL) pour garantir l'authenticité et l'intégrité des messages échangés entre les parties.

Algorithme MAC	Référence	Statut	Restrictions
HMAC-SHA-256	RFC 2104 et FIPS 198-1	A	
HMAC-SHA-384	RFC 2104 et FIPS 198-1	A	
HMAC-SHA-512	RFC 2104 et FIPS 198-1	A	
KMAC128	NIST SP 800-185	A	
KMAC256	NIST SP 800-185	A	
BLAKE3 (mode keyed_hash)	BLAKE3 one function, fast everywhere	A	
AES-CMAC	RFC 4493 et NIST SP 800-38B	A	
AES-GMAC	NIST SP 800-38D	A	
Poly1305-AES	The Poly1305-AES message-authentication code	A	
HMAC-SHA-1	RFC 2104 et FIPS 198-1	L	
HMAC-MD5	RFC 1321	D	

Signatures numériques

Les schémas de signature DOIVENT utiliser des tailles de clés et des paramètres approuvés conformément à la norme NIST SP 800-57 Partie 1.

Algorithme de signature	Référence	Statut
EdDSA (Ed25519, Ed448)	RFC 8032	A
XEdDSA (Curve25519, Curve448)	XEdDSA	A
ECDSA (P-256, P-384, P-521)	FIPS 186-4	A
RSA-RSSA-PSS	RFC 8017	A
RSA-SSA-PKCS#1 v1.5	RFC 8017	D
DSA (n'importe quelle taille de clé)	FIPS 186-4	D

Normes de chiffrement post-quantique

Les implémentations PQC doivent être conformes à la norme FIPS-203/204/205, car il existe encore un code renforcé minimal et une référence d'implémentation minimale. https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

La méthode d'accord de clé TLS hybride post-quantique proposée mlkem768x25519 est prise en charge par les principaux navigateurs tels que Firefox version 132 et Chrome version 131. Elle peut être utilisée dans des environnements de test cryptographique ou lorsqu'elle est disponible dans des bibliothèques approuvées par l'industrie ou le gouvernement.

Annexe D : Recommandations

Introduction

Lors de la préparation de la version 5.0 de la norme de vérification de la sécurité des applications (ASVS), il est apparu clairement que plusieurs éléments, existants ou nouvellement proposés, ne devaient pas être inclus comme exigences dans la version 5.0. Cela peut s'expliquer par le fait qu'ils n'entraient pas dans le champ d'application de l'ASVS selon la définition de la version 5.0, ou bien par le fait que, malgré leur pertinence, ils ne pouvaient être rendus obligatoires.

Ne voulant pas perdre tous ces éléments entièrement, certains ont été capturés dans cette annexe.

Mécanismes de portée recommandés

Les éléments suivants sont concernés par ASVS. Ils ne devraient pas être rendus obligatoires, mais il est fortement recommandé de les prendre en compte dans le cadre d'une application sécurisée.

- Un indicateur de sécurité des mots de passe devrait être fourni pour aider les utilisateurs à définir un mot de passe plus fort.
- Créez un fichier security.txt accessible au public à la racine ou dans le répertoire .well-known de l'application, définissant clairement un lien ou une adresse e-mail permettant de contacter les propriétaires en cas de problème de sécurité.
- La validation des entrées côté client devrait être appliquée en plus de la validation au niveau d'une couche de service de confiance, car cela permet de détecter si un utilisateur a contourné les contrôles côté client pour tenter d'attaquer l'application.
- Empêchez l'affichage de pages sensibles et accessibles accidentellement dans les moteurs de recherche à l'aide d'un fichier robots.txt, de l'en-tête de réponse X-Robots-Tag ou d'une balise méta HTML robots.
- Lorsque vous utilisez GraphQL, implémentez la logique d'autorisation au niveau de la couche logique métier plutôt que dans la couche GraphQL ou du résolveur afin d'éviter de gérer l'autorisation sur chaque interface distincte.

Références :

- Plus d'informations sur security.txt, y compris un lien vers la RFC

Principes de sécurité des logiciels

Les éléments suivants figuraient déjà dans ASVS, mais ne constituent pas de véritables exigences. Il s'agit plutôt de principes à prendre en compte lors de la mise en œuvre de contrôles de sécurité, dont le respect permettra de renforcer la robustesse. Parmi ceux-ci :

- Les contrôles de sécurité doivent être centralisés, simples (économie de conception), vérifiables et réutilisables. Cela permet d'éviter les contrôles dupliqués, manquants ou inefficaces.
- Dans la mesure du possible, utilisez des implémentations de contrôle de sécurité préalablement écrites et bien contrôlées plutôt que de vous fier à la mise en œuvre de contrôles à partir de zéro.
- Idéalement, un mécanisme de contrôle d'accès unique devrait être utilisé pour accéder aux données et ressources protégées. Toutes les requêtes devraient transiter par ce mécanisme unique afin d'éviter les copier-coller ou les chemins alternatifs non sécurisés.
- Le contrôle d'accès basé sur les attributs ou les fonctionnalités est un modèle recommandé : le code vérifie l'autorisation de l'utilisateur pour une fonctionnalité ou un élément de données, plutôt que simplement son rôle. Les autorisations doivent toujours être attribuées à l'aide de rôles.

Processus de sécurité des logiciels

Plusieurs processus de sécurité ont été supprimés d'ASVS 5.0, mais restent pertinents. Le projet OWASP SAMM pourrait être une bonne source d'inspiration pour une mise en œuvre efficace de ces processus. Parmi les éléments précédemment présents dans ASVS, on peut citer :

- Vérifier l'utilisation d'un cycle de vie sécurisé pour le développement de logiciels qui prend en compte la sécurité à tous les stades du développement.
- Vérifier l'utilisation de la modélisation des menaces pour chaque changement 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.
- Vérifier que toutes les histoires d'utilisateurs et les caractéristiques contiennent des contraintes de sécurité fonctionnelles, telles que "En tant qu'utilisateur, je devrais être en mesure de voir et de modifier mon profil. Je ne dois pas pouvoir consulter ou modifier le profil de quelqu'un d'autre"
- Vérifier la disponibilité d'une liste de contrôle de codage sécurisé, d'exigences de sécurité, de lignes directrices ou d'une politique pour tous les développeurs et les testeurs.
- Vérifier l'existence d'un processus permanent visant à garantir que le code source de l'application est exempt de portes dérobées, de codes malveillants (attaques par salami, bombes logiques, bombes à retardement) et de caractéristiques non documentées ou cachées (œufs de Pâques, outils de débogage non sécurisés). Il n'est pas possible de se conformer à cette section sans un accès complet au code source, y compris aux bibliothèques tierces, et cela ne convient donc probablement qu'aux applications exigeant les niveaux de sécurité les plus élevés.
- Vérifier que des mécanismes sont en place pour détecter les dérives de configuration dans les environnements déployés et y répondre. Il peut s'agir de l'utilisation d'une infrastructure immuable, d'un redéploiement automatisé à partir d'une base sécurisée ou d'outils de détection de dérive qui comparent l'état actuel aux configurations approuvées.
- Vérifier que le durcissement de la configuration est effectué sur tous les produits tiers, les bibliothèques,

Références:

- OWASP Threat Modeling Cheat Sheet
- OWASP Threat modeling
- OWASP Software Assurance Maturity Model Project
- Microsoft SDL

Appendix E - Contributeurs

Nous remercions les personnes suivantes pour leurs contributions, leurs commentaires et leurs demandes d'extension depuis la publication de ASVS 4.0.0.

Si vous avez connaissance d'erreurs ou si vous souhaitez que votre nom apparaisse différemment, merci de nous le faire savoir.

Johan Sydseter (sydseter)	luis servin (lfservin)	Oleksii Dovydkov (oleksiidov)	IZUKA Masahiro (maizuka)
James Sulinski (jsulinski)	Eli Saad (ThunderSon)	kkshitish9	Andrew van der Stock (vanderaj)
Rick M (kingthorin)	Bankde Eakasit (Bankde)	Michael Gargiullo (mgargiullo)	Raphael Dunant (Racater)
Cesar Kohl (cesarkohl)	inaz0	Joerg Bruenner (JoergBruenner)	David Deatherage (securitydave)
John Carroll (yosignals)	Jim Fenton (jimfenton)	Matteo Pace (M4tteoP)	Sebastien gioria (SPoint42)
Steven van der Baan (vdbaan)	Jeremy Bonghwan Choi (jeremychoi)	craig-shony	Riccardo Sirigu (ricsirigu)
Tomasz Wrobel (tw2as)	Alena Dubeshko (belalena)	Rafael Green (RafaelGreen1)	mjang-cobalt
clallier94	Kevin W. Wall (kwwall)	Jordan Sherman (jsherm-fwdsec / deleterepo)	Ingo Rauner (ingo-rauner)
Dirk Wetter (drwetter)	Moshe Zioni (moshe-apiiro)	Patrick Dwyer (coderpatros)	David Clarke (davidclarke-au)
Takaharu Ogasa (takaharuogasa)	Arkadii Yakovets (arkid15r)	Motoyasu Saburi (motoyasu-saburi)	leirn
wet-certitude	timhemel	RL Thornton (thornshadow99)	Thomas Bandt (aspnetde)
Roel Storms (roelstorms)	Jeroen Willemsen (commjoen)	anonymous-31	Kamran Saifullah (deFr0ggy)
Steve Springett (stevespringett)	Spyros (northdpole)	Hans Herrera (hansphp)	Marx314
CarlosAllendes	Yonah Russ (yruss972)	Sander Maijers (sanmai-NL)	Luboš Bretschneider (bretik)
Eva Sarafianou (esarafianou)	Ata Seren ataseren	Steve Thomas (Sc00bz)	Dominique RIGHETTO (righettod)
Steven van der Baan (svdb-ncc)	Michael Vacarella (Aif4thah)	Tonimir Kisasondi (tkisason)	Stefan Streichsbier (streichsbaer)

hi-unc1e	sb3k (starbuck3000)	mario-platt	Devdatta Akhawe (devd)
Michael Gissing (scolytus)	Jet Anderson (thatsjet)	Dave Wichers (davewichers)	Jonny Schnittger (JonnySchnittger)
Silvia Väli (silviavali)	jackgates73	1songb1rd	Timur - (timurozkul)
Gareth Heyes (hackvertor)	appills	suvikaartinen	chaals (chaals)
DanielPharos (AtlasHackert)	will Farrell (willfarrell)	Alina Vasiljeva (avasiljeva)	Paul McCann (ismisepaul)
Sage (SajjadPourali)	rbsec	Benedikt Bauer (mastacheata)	James Jardine (jamesjardine)
Mark Burnett (m8urnett)	dschwarz91	Cyber-AppSec (Cyber-AppSec)	Tib3rius
BitnessWise (bitnesswise)	damienbod (damienbod)	Jared Meit (jmeit-fwdsec)	Stefan Seelmann (sseelmann)
Brendan O'Connor (ussjoin)	Andrei Titov (andrettv)	Hans-Petter Fjeld (atluxity)	markehack
Neil Madden (NeilMadden)	Michael Geramb (mgeramb)	Osama Elnaggar (ossie-git)	mackowski
Ravi Balla (raviballa)	Hazana (hazanasec)	David Means (dmeans82)	Alexander Stein (tohch4)
BaeSenseii (baesenseii)	Vincent De Schutter (VincentDS)	S Bani (sbani)	Mitsuaki Akiyama (mak1yama)
Christopher Loessl (hashier)	victorxm	Michal Rada (michalradacz)	Veeresh Devireddy (drveresh)
MaknaSEO	darkzero2022	Liam (LiamDobbelaere)	Frank Denis (jedisct1)
Otto Sulin (ottosulin)	carllaw6885	Anders Johan Holmefjord (aholmis)	Richard Fritsch (rfricz)
mesutgungor	Scott Helme (ScottHelme)	Carlo Reggiani (carloreggiani)	Suyash Srivastava (suyash5053)
Mark Potter (markonweb)	Arjan Lamers (alamers)	Gøran Breivik (gobrtg)	flo-blg
Guillaume Déflache (guillaume-d)	Toufik Airane (toufik-airane)	Keith Hoodlet (securingdev)	Sinner (SoftwareSinner)
iloving	Jeroen Beckers (TheDauntless)	Joubin Jabbari (joubin)	yu fujioka (fujiokayu)
execjosh (execjosh)	Alicja Kario (tomato42)	Sidney Ribeiro (srjsoftware)	Gabriel Marquet (Gby56)
Drew Schulz (drschulz)	bedirhan	muralito	Ronnie Flathers (ropnop)

Philippe De Ryck (philippederyck)	Malte (mal33)	MazeOfThoughts	Andreas Falk (andifalk)
Javi (javixeneize)	Daniel Hahn (averell23)	borislav-c	Robin Wood (digininja)
miro2ns	Jan Dockx (jandockx)	vipinsaini434	priyanshukumar397
Nat Sakimura (sakimura)	Benjamin Häublein (BenjaminHae)	unknown-user-from	Ali Ramazan TAŞDELEN (alitasdln)
Pedro Escaleira (oEscal)	Josh (josh-hemphill)	Tim Würtele (SECtim)	AviD (avidouglen)
SheHacksPurple (shehackspurple)	fcerullo-cycubix	Hector Eryx Paredes Camacho (heryxpc)	Irene Michlin (irene221b)
Jonah Y-M (TG-Techie)	Dhiraj Bahroos (bahroos)	Jef Meijvis (jefmeijvis)	IzmaDoesItbeta
Abdessamad TEMMAR (TmmmmmR)	sectroyer	Soh Satoh (sohsatoh)	regoravalaz
james-t (james-bitherder)	Aram Hovsepyan (aramhovsepyan)	JaimeGomezGarciaSan	ValdiGit01
iwatachan (ishowta)	Vinod Anandan (VinodAnandan)	Kevin Kien (KevinKien)	paul-williamson-swoop
endergzr	Radhwan Alshamamri (Rado0z)	Grant Ongers (rewtd)	Cure53 (cure53)
AliR2Linux	Ads Dawson (GangGreenTemperTatum)	William Reyor (BillReyor)	gabe (gcrow)
mascotter	luissaiz	Suren Manukyan (vx-sec)	Piotr Gliźniewicz (pgliźniewicz)
Tadeusz Wachowski (tadeuszwachowski)	Nasir aka Nate (andesec)	settantasette	Lars Haulin (LarsH)
Terence Eden (edent)	JasmineScholz	Arun Sivadasan (teavanist)	Yusuf GÜR (yusuffgur)
Troy Marshall (troymarshall)	Tanner Prynn (tprynn)	Nick K. (nickific)	raoul361
Azeem Ilyas (TheAxZim)	Evo Stamatov (avioli)	Tim Potter (timpotter87)	Gavin Ray (GavinRay97)
monis (demideus)	Marcin Hoppe (MarcinHoppe)	Grambulf (ramshazar)	Jordan Pike (computersarebad)
Jason Rogers (jason-invision)	Ben Hall (benbhall)	JamesPoppyCock (jamesly123)	WhiteHackLabs (whitehacklabs)
Alex Gaynor (alex)	Filip van Laenen (filipvanlaenen)	jeurgen	GraoMelo
Andreas Kurtz (ay-kay)	Tom Tervoort (TomTervoort)	old man (deveras)	Marco Schnüriger (marcortw)

stiin	infoseclearn (teaminfoseclearn)	hljupkij	Noe (nmarher)
Lyz (lyz-code)	Martin Riedel (mrtnrdl)	KIM Jaesuck (tcaesvk)	Barbara Schachner (bschach)
René Reuter (AresSec)	carhackpils	Tyler (tyler2cr)	Hugo (hasousa)
Wouter Bloeyaert (Someniak)	Mark de Rijk (markderijkinfosec)	Ramin (picohub)	Philip D. Turner (philipdturner)
Will Chatham (willc)			