

Groupe 6

Professeur: Pr. Gervais Mendy

Membres du groupe:

Arame Diaw (tr) | Mouhamadou Lamine Gueye (inf) |
Hassastou Diallo (inf) | Ibrahima Miniane Diouf (tr) |
Tibiang yeshua Doumgou (iinf)

Ecole Supérieure Polytechnique de Dakar | Département Génie Informatique

DIC3

Ingénierie Cryptographique

Introduction

Le top 10 est OWASP est un document informatif . Elle est utilisée aussi par certaines entreprises en tant que standard de développement ou de réalisation de tests . Cependant il n'est qu'un point de départ et ne couvre que le strict minimum.

L'OWASP est un nouveau type d'organisation. Leur indépendance vis-à-vis des pressions commerciales nous permet de fournir des informations impartiales, pratiques et rentables sur la sécurité des applications.

Explications des facteurs d'OWASP top 10:

- **CWEs associées** : le nombre de CWEs associées à une catégorie par l'équipe du Top 10.
- **Taux d'incidence** : le taux d'incidence est le pourcentage d'applications vulnérables à cette CWE parmi la population testée par cette organisation pour cette année.
- **Exploitation pondérée** : le sous-score Exploitation des scores CVSSv2 et CVSSv3 attribués aux CVEs associées aux CWEs, normalisés et placés sur une échelle de 10 points.
- **Impact pondéré** : le sous-score d'impact des scores CVSSv2 et CVSSv3 attribués aux CVEs associées aux CWEs, normalisés et placés sur une échelle de 10 points.
- **Couverture (Test)** : Le pourcentage d'applications testées par toutes les organisations pour une CWE donnée.
- **Nombre total d'occurrences** : nombre total d'applications trouvées pour lesquelles les CWEs sont associées à une catégorie.

- Nombre total de CVEs : nombre total de CVEs dans la base de données NVD qui ont été associées aux CWEs associées à une catégorie.

Le CWE (common weakness enumeration) est un projet qui a pour but de cataloguer les vulnérabilités des applications.


CVSS (*common Vulnerability Scoring System*) est un système permettant de noter les vulnérabilités.

Nom/ Critères		CW Es ass oci ées	Taux d'inci denc e max	Taux d'incid ence moye n	Expl oitati on pond érée moy enne	Impa ct pond éré moye n	Couvert ure max	Couvert ure moyenn e	Nombre total d'occurr ences	Nombre total de CVEs
A01:2021 Contrôles d'accès défaillants	–	34	55,97 %	3,81 %	6,92	5,93	94,55 %	47,72 %	318 487	19 013
A02:2021 Défaillances cryptographiques	–	29	46,44 %	4,49 %	7,29	6,81	79,33 %	34,85 %	233 788	3 075
A03:2021 Injection	–	33	19,09 %	3,37 %	7,25	7,15	94,04 %	47,90 %	274 228	32 078
A04:2021 Conception sécurisée	– non	40	24,19 %	3,00 %	6,46	6,78	77,25 %	42,51 %	262 407	2 691
A05:2021 Mauvaise configuration de sécurité	– de	20	19,84 %	4,51 %	8,12	6,56	89,58 %	44,84 %	208 387	789

A06:2021 Composants vulnérables et obsolètes	– 3	27,96 %	8,77 %	5,00	5,00	51,78 %	22,47 %	30 457	0
A07:2021 Identification et authentification de mauvaise qualité	– 22	14,84 %	2,55 %	7,40	6,50	79,51 %	45,72 %	132,19 5	3 897
A08:2021 Manque d'intégrité des données et du logiciel	– 10	16,67 %	2,05 %	6,94	7,94	75,04 %	45,35 %	47 972	1 152
A09 Carence des systèmes de contrôle et de journalisation	4	19,23 %	6,51 %	6,87	4,99	53,67 %	39,97 %	53 615	242
A10:2021 – A10 Falsification de requête côté serveur (SSRF)	1	2,72 %	2,72 %	8,28	6,72	67,72 %	67,72 %	9 503	385

Après analyse du tableau ci-dessus, et après avoir pris connaissance de la signification des critères présentés, nous remarquons qu'il y a une forte corrélation entre le nombre de CWE associés et le rang de chaque CWE. Cependant cette corrélation ne peut être parfaite car le classement de 8 de ces derniers (les 8 premiers) dépend du nombre de CWE associés et le taux d'incidence moyen et le nombre total d'occurrences; le classement des 2 autres a été déterminé par la communauté spécialiste.

Nous remarquons aussi que les CVE en haut du classement ont globalement un plus grand nombre total d'occurrences, nombre total de



CVE (vulnérabilités). Nous pouvons en déduire que le classement OWASP-10 est fortement corrélé à la popularité/exploitabilité des CWE.

A06:2021 est la seule catégorie à n'avoir aucune CVE répertoriée, en conséquence, les coefficients d'impact et de poids ont été renseignés à 5 par défaut. Cette catégorie progresse depuis sa 9e place en 2017; elle est un problème connu dont nous avons du mal à tester et à mesurer les risques. Son classement en 6e position est donc très peu justifiable. Cependant, auprès de la communauté il se classe 2e !

Pour **A05:2021**, avec des logiciels de plus en plus paramétrables que nous possédons, il n'est pas surprenant de voir cette catégorie prendre de l'ampleur (comparativement à son classement des années précédentes).

Avec l'augmentation exponentielle du nombre de plateformes (nécessitant authentification) et la difficulté de "patcher" cette fonctionnalité, nous avons au fil des années observé la montée de la catégorie **Contrôles d'accès défaillant** dans le classement, actuellement première.

De manière générale, nous remarquons aussi que l'inaccessibilité d'exploitations de certaines de ces catégories les fait tomber dans le classement. Cela semble naturel car la dangerosité d'une exploitation dépend du nombre d'attaquants. Or dans certains cas, seule une niche de spécialiste peut exploiter ces vulnérabilités. **A08:2021 – Manque d'intégrité des données et du logiciel** est dans ce cas car il nécessite des compétences avancées en networking pour espérer les tester/exploiter. Cela est appuyé par le fait que les 10 catégories du classement ne sont pas loin les unes des autres - niveau dangerosité. On peut le dire car leurs exploitation pondérée moyenne et impact pondéré moyen varient peu dans le tableau.

Proposition de Sujet: Vulnérabilités liées à une mauvaise gestion d'authentification (à partir de A07)

Ce sujet volontairement vaste nous garantit d'avoir un lien avec le cours à tous ses niveaux. Il est possible en effet de le lier à plusieurs catégories de vulnérabilités du classement OWASP.

Ex: Contrôle d'accès défaillant, défaillances cryptographiques, injections, conception non sécurisée, etc.

Après chaque itération du processus de développement, le livrable à fournir sera une application WEB (encore à spécifier) sur laquelle une fonctionnalité (fonctionnelle ou non) aura été ajoutée. Nous montrerons son lien avec le cours et expliquerons comment nous avons ou comptons éviter les vulnérabilités ci-dessus.

Le processus de développement en question fait allusion au fait que nous déterminerons toutes les semaines les tâches à faire pour la semaine à venir, leur assignation et les exigences liées... Suivant la satisfaction du client (professeur) nous passerons aux prochaines tâches et referons les mêmes.

Note: La présentation du livrable pourra aussi inclure les approches abordées (non nécessairement fructueuses) car étant un plus.

Traitement du sujet

1. Matching avec le cours:

- La non-utilisation d'algorithmes cryptographiques dans le cas du stockage de mot de passe par une application peut faire office de preuve de concept du cours. Une personne mal intentionnée accédant à une telle base de données pourra se faire passer pour un autre utilisateur. C'est décrit par CWE-259: Use of Hard-coded Password (4.9).
- Lorsqu'une plateforme WEB effectue un contrôle d'accès sans utiliser de certificat ssl (https), elle envoie des informations en clair dans le réseau. Ces informations peuvent alors facilement être interceptées et surtout exploitées par une personne mal intentionnée qui pourra, dès lors, se faire passer pour un autre utilisateur. De la même manière que le point précédent, cela renvoie au cours introductif à la cryptographie, notamment le quatrième Principe de Kerckhoff. Référence: CWE-290: Authentication Bypass by Spoofing (4.9). Référence du cours: Cours introductif, Attaques cryptographiques, "**L'attaque de l'homme du milieu**".
- En combinant CWE-521: Weak Password Requirements (4.9) et CWE-307: Improper Restriction of Excessive Authentication Attempts (4.9), il devient plus facile pour l'attaquant de faire une attaque par Brute Force vu que la première vulnérabilité facilite la compromission des comptes utilisateurs; et la seconde, elle ne le limite pas son nombre de tentatives de connexion. Cela est lié au cours introductif à la cryptographie, notamment à la section "**Attaque par force brute**", et aussi au chapitre sur le Hachage cryptographique, notamment à la "**Résistance des fonctions de hachage**". Si de plus, les mots de passes ont été chiffré à l'aide d'une clé, on touche alors aux **chapitre 4 et 5** suivant respectivement que cette dernière soit secrète ou publique.

2. Scénarios:

Considérons une entreprise dont les employés ont à s'authentifier via une plateforme. Si les informations de connexions n'incluent pas le poste avec lequel la connexion se fait, toute personne mal intentionnée et externe à l'entreprise, disposant des informations de connexion de l'un des employés peut se faire passer pour cet utilisateur et mener tout type d'action.

Un grand nombre d'attaques se déroulent dû à l'utilisation du mot de passe comme unique facteur d'authentification. Un attaquant pourrait donc exploiter cela, notamment à cause du fait que les bonnes pratiques de rotation de mot de passe et l'utilisation de mot de passe fort encouragent plutôt les utilisateurs à utiliser des mots de passe faible. l'attaquant pourrait donc à travers une attaque par dictionnaire ou par force brute usurper l'identité de l'utilisateur.

La réutilisation de mots de passe, l'utilisation de mots de passe connus, est une attaque classique. Supposons une application qui n'implémente pas une protection automatique contre le bourrage d'informations ou l'utilisation des mots de passe connus. Dans ce cas, l'application peut être utilisée comme un oracle pour déterminer si les mots de passe sont valides.

Prenons le cas où une interface d'authentification sensible aux SQL Injections et dont les mots de passes sont stockés en clair dans la base de données; Un attaquant peut, à partir de l'interface de connexion, injecter du code sql qui lui renvoie la liste des couples [login, password] de la base de données, et ainsi usurper leur identité. Il peut même aller plus loin et modifier ces informations si la plateforme a des droits d'écriture dans la base de données...et dans le pire des cas, supprimer toutes les données à partir d'un compte admin par exemple (identifiants découverts plus tôt).

Partie de la présentation

- La non-utilisation d'algorithmes cryptographiques dans le cas du stockage de mot de passe par une application peut faire office de preuve de concept du cours. Une personne mal intentionnée accédant à une telle base de données pourra se faire passer pour un autre utilisateur. C'est décrit par CWE-259: Use of Hard-coded Password (4.9).

En effet, afin de stocker les mots de passe dans une base de données il est recommandé de ne pas le faire en clair. Il faut donc une méthode qui va permettre aux développeurs de réaliser cela. C'est là qu'interviennent les fonctions de hachage cryptographique qui ne font que mapper une chaîne de bits de longueur aléatoire en une chaîne de bit de longueur fixe. On a plusieurs fonctions de hachages à savoir. Cependant pour qu'une fonction de hachage soit considérée comme bonne il faut qu'elle résiste aux pré-images (qui est le fait qu'à partir d'un haché H on puisse trouver un message M dont le haché est H), aux collisions (qui est le fait de trouver deux messages ou plusieurs ayant le même haché...principe des tiroirs). Donc en générale si une fonction de hachage respecte cela on peut la considérer comme correcte et utilisable.

fonctions de hachage tel que le Mac

Les fonctions de hachages vont donc nous permettre de cacher les informations secrètes de nos utilisateurs en cas de fuite des informations de la base de données. **Résistance des fonctions de hachage, Attaque par force brute**

- Lorsqu'une plateforme WEB effectue un contrôle d'accès sans utiliser de certificat ssl (https), elle envoie des informations en clair dans le réseau.

Ces informations peuvent alors facilement être interceptées et surtout exploitées par une personne mal intentionnée qui pourra, dès lors, se faire passer pour un autre utilisateur. De la même manière que le point précédent, cela renvoie au cours introductif à la cryptographie, notamment le quatrième Principe de Kerckhoff. Référence: CWE-290: Authentication Bypass by Spoofing (4.9). Référence du cours: Cours introductif,

Chiffrement

Principe de tiroir qui dit que si on a n chaussette qui occupe m tiroir et si $n > m$ alors il existe au moins un tiroir qui possède deux chaussettes ou plus

donc si une fonction de hachage possède plus d'entrées que de sortie (notons que la fonction de hachage fournit un nombre fixé de bit) si le hachage est faible alors le nombre de sortie sera inférieur au nombre de sortie dans ce cas la collision est inévitable.

(si les interactions entre deux parties ne sont pas proprement sécurisées, que cela soit un transfert de données entre un client et un serveur ou tout simplement une communication entre deux utilisateurs à travers un système de messagerie internet par exemple, une attaque de l'homme au milieu peut avoir lieu. De manière simple cette attaque consiste en ce qu'un attaquant se place entre une communication généralement entre un utilisateur et une application et essaie soit d'écouter la conversation soit se faire passer pour l'une des parties sans qu'elles ne s'en rendent compte. A la lumière de cela on peut donc déduire que si aucune méthode de cryptage n'est appliquée à l'information (par exemple les identifiants de connexion) transmise l'attaquant peut donc l'intercepter et pouvoir l'utiliser à ses propres fins. **L'attaque de l'homme du milieu**

Mise en oeuvre

Technologies

- HTML, CSS, JS pour l'implémentation
- Python pour concevoir nos "exploit" si nécessaire
- Bootstrap (potentiellement) pour le style
- Java
- docker
- Github pour le versionnement, le travail collaboratif et le suivi des différentes démos (et de leurs solutions)
- Wireshark (outil permettant d'intercepter et analyser le trafic dans le réseau)
- Metasploit (logiciel open source de tests de pénétration)
- shodan.io (moteur de recherche qui permet aux utilisateurs de rechercher différents types de serveurs connectés à internet en utilisant un éventail varié de filtres)
- Buildwith
- wappalyzer
- whatweb

Procédés

Après chaque itération du processus de développement, le livrable à fournir sera une application WEB qui présentera au moins une vulnérabilité dans notre thème , qu'on exploitera. Et une autre version de cette même application, mais cette fois-ci patchée. On expliquera potentiellement les différentes solutions qui ont pu être apportées à la vulnérabilité, et celles retenues. Dans un premier temps, suivra un scénario réaliste et qui a eu lieu.

Pour implémenter cela, nous allons essayer de mettre en place une application web avec un formulaire qui demandera à l'utilisateur de saisir ces informations personnelles , de définir un login et un mot de passe . Pour une première, ces informations seront stockées en clair dans la base de données. Ce stockage d'information dans la base de données sans chiffrement ni hachage est une vulnérabilité qu' utilisera l'attaquant pour récupérer les informations. Après avoir prouvé la vulnérabilité et l'exploitation qu'un attaquant peut en faire, on propose une version de l'application résistante à la faille.

Pour le scénario de l'homme du milieu , le formulaire soumis par l'utilisateur transitera au sein d'un réseau . L'attaquant au moyen d'un sniffeur de réseau écoutera et essaiera d'intercepter les requêtes , récupérera ainsi les messages affichés en clair ou hachés. On montrera l'exploitation dans le cas où les messages sont chiffrés (hachés et/ou chiffrés dans le cas de l'authentification) et non. Nous verrons également le cas où la méthode de chiffrement ou de hachage est faible.

Un autre type de scénario est que l'utilisateur, au lieu de remplir les différents champs du formulaire par des informations "normales " , décide d'injecter du code sql qui récupère les informations stockées dans la base de données.