Sécuriser une application Web 

Sommaire

Définition d’une faille

Dans le domaine de la sécurité informatique, une **vulnérabilité** ou **faille** est une faiblesse dans un système informatique permettant à un attaquant de porter atteinte à l'intégrité de ce système, c'est-à-dire à son fonctionnement normal, à la confidentialité ou à l'intégrité des données qu'il contient.

Ces vulnérabilités sont la conséquence de faiblesses dans la conception, la mise en œuvre ou l'utilisation d'un composant matériel ou logiciel du système, mais il s'agit souvent d'anomalies logicielles liées à des erreurs de programmation ou à de mauvaises pratiques.

Ces dysfonctionnements logiciels sont en général corrigés à mesure de leurs découvertes, mais l'utilisateur reste exposé à une éventuelle exploitation tant que le correctif (temporaire ou définitif) n'est pas publié et installé.

C'est pourquoi il est important de maintenir les logiciels à jour avec les correctifs fournis par les éditeurs de logiciels. La procédure d'exploitation d'une vulnérabilité logicielle est appelée exploit.

Les vulnérabilités informatiques proviennent souvent de la négligence ou de l'inexpérience d'un programmeur.

Il peut y avoir d'autres causes liées au contexte comme l'évolution des technologies, notamment en cryptographie. Une vulnérabilité permet généralement à l'attaquant de duper l'application, par exemple en outrepassant les vérifications de contrôle d'accès ou en exécutant des commandes sur le système hébergeant l'application.

Quelques vulnérabilités surviennent lorsque l'entrée d'un utilisateur n'est pas contrôlée, permettant l'exécution de commandes ou de requêtes SQL (connues sous le nom d'injection SQL). D'autres proviennent d'erreurs d'un programmeur lors de la vérification des buffers de données (qui peuvent alors être dépassés), causant ainsi une corruption de la pile mémoire (et ainsi permettre l'exécution de code fourni par l'attaquant).

1.1 Les injections SQL

L'injection SQL est une des attaques les plus classiques qu'elle en est devenue la plus populaire. Son principe est tellement simple mais ingénieux et en 2017, elle paraît encore à la tête du TOP 10 OWASP.

L'attaque consiste à forger son propre code SQL que l'on soumet au serveur. Ce code là sera mélangé à du code SQL propre à l'application (ou site Web) ce qui conduira au détournement du fonctionnement attendu. Les sites qui peuvent être vulnérables à cette attaque sont ceux qui se basent sur une base de données SQL. Donc si le site ne repose pas sur une base de données, il ne présente aucun risque vis-à-vis de l'injection SQL.
Quand cette attaque réussit, elle peut avoir des conséquences désastreuses sur le site Web comme par exemple:

- L'accès à un espace non autorisé via une fausse authentification
- Suppression des données d'une manière frauduleuse
- Vol de données confidentielles enregistrées dans la base de données
- Destruction ou atteinte à l'intégrité de la base de données

L'attaque repose sur l'exploitation des entrées d'un site Web comme les champs de formulaires ou la barre d'URL. Par exemple, quand on demande à l'utilisateur de s'authentifier en entrant un login et un mot de passe, si celui-ci était un pirate aux mauvaises intentions, alors au lieu de saisir du texte normal il saisit des séquences de codes SQL qu'il soumettait au serveur. Si le site présente une vulnérabilité à l'injection SQL alors il se peut que le code SQL injecté s'exécute et donne lieu à un comportement inattendu comme avoir accès illégalement à l'espace confidentiel.
Par exemple, imaginez que le code SQL (encapsulé dans le code PHP) qui permet de vérifier si un login et un mot de passe sont valides est le suivant:

```
select * from table_users where login='$_POST["login"]' and pass='$_POST["pass"]'
```

Notez que **$_POST["login"]** et **$_POST["pass"]** représentent les textes que l'utilisateur a saisi dans les champs de formulaire dédiés à cet effet.Cette requête SQL retourne toutes les lignes (en général y'en aura qu'une seule) où les champs 'login' et 'pass' contiendront respectivement le login et le mot de passe saisis par le client.

1.2 La faille Upload

Beaucoup de sites Web offrent la possibilité aux clients d'y uploader des fichiers comme des photos, des vidéos, des CV... Donc il ne s'agit pas vraiment d'une vulnérabilité, mais c'est le fait de ne pas contrôler ce que le client charge sur le serveur qui constitue une vulnérabilité très dangereuse.

Le principe de l'attaque est très simple. Le pirate essaie d'uploader un fichier qui contient du code malveillant ou un code PHP de sa création. Si la faille est là alors le fichier finira pas atterrir sur le serveur. Il suffit ensuite au pirate d'appeler son fichier pour que celui-ci s'exécute.

Bien entendu une telle attaque peut avoir de graves conséquences comme par exemple:

- Espionnage des fichiers et dossiers du site
- Accès au fichiers systèmes et fichiers confidentiels
- Destruction ou altération de données existantes sur le serveur
- Prise de contrôle du serveur

Imaginons que le site Web contient un champ d'upload de fichiers qui permet aux utilisateurs de charger leurs photos de profil. Si la vulnérabilité est là alors le pirate peut créer un document PHP du nom de **crawler.php** qui contient à titre d'exemple le code suivant:

```
$pt = opendir('/');
while($entree = readdir($pt)){
   echo $entree;
}
closedir($pt);
```

Comment s'en protéger?

Au niveau du code PHP

Comme d'habitude, la vulnérabilité est due au mauvais contrôle des entrées de l'utilisateur, alors qu'il suffisait de vérifier si le type/Mime du fichier uploadé correspond bien à une image JPEG ou PNG en utilisant le code suivant par exemple:

```
<?php

   if(preg_match("#jpeg|png#",$_FILES["photo"]["type"]))

      // Accépter l'upload

   else

      echo "Format du fichier invalide.";

?>
```

Il faut également penser à isoler les fichiers chargés dans un dossier à part pour minimiser les risques de rebond au cas où il s'agit d'un fichier malveillant.

Renommer les fichiers chargés sera aussi d'une grande utilité car le pirate aura du mal à appeler son fichier s'il ne connaît pas son chemin et son nom.

1.3 Le Cross-Site Scripting (XSS)

XSS est un terme utilisé pour décrire une classe d'attaque qui permet à l'attaquant d'injecter des scripts, exécutés côté-client, *au travers* du site web pour viser le navigateur web des autres utilisateurs. Comme le code injecté provient du site web, le navigateur web le considère comme sûr, il peut de ce fait faire des choses comme transmettre le cookie d'authentification de l'utilisateur à l'attaquant. Une fois que l'attaquant obtient ce cookie il peut se connecter sur le site comme si il était l'utilisateur attaqué et peut faire tout ce que l'utilisateur pourrait faire. En fonction du site sur lequel l'attaque se produit, cela peut inclure l'accès aux détails de carte bancaire, les informations des contacts, la modification du mot de passe, etc.

Notes: Les vulnérabilités xss ont historiquement été les plus courantes.

*Il y a deux manières principales pour demander au site de retourner un script injecté vers un navigateur web — elles sont désignées en tant que vulnérabilités XSS *réfléchie* et *persistante*.

- Une vulnérabilité XSS réfléchie se produit quand le contenu de l'utilisateur transmis au serveur est immédiatement retourné, sans avoir été modifié, pour être affiché dans le navigateur — tous les scripts présents dans le contenu d'origine seront exécutés quand la nouvelle page sera chargée!
  On prendra par exemple une fonction de recherche dans un site où les termes recherchés sont encodés en tant que paramètres dans l'URL, et que ces termes sont affichés en permanence avec les résultats. Un attaquant peut construire un lien de recherche contenant un script malicieux en tant que paramètre (ex: `http://mysite.com?q=beer<script%20src="http://sitedangereux.com/malicieux.js"></script>`) et le transmettre par e-mail à un autre utilisateur. Si l'utilisateur ciblé clique sur ce lien intéressant, le script sera exécuté quand les résultats de la recherche seront affichés. Comme vu auparavant, cela donne à l'attaquant toute les informations dont il a besoin pour se connecter sur le site avec le compte de la victime — potentiellement faire des achats en tant que cet utilisateur ou accéder à la liste de contacts..
- Une vulnérabilité XSS *persistante* sera celle où le script malicieux est stocké sur le site web puis affiché, sans modification, un peu plus tard par les autres utilisateurs et exécuté à leur insu.
  Par exemple, un écran de discussion qui accepte les commentaires contenant du code HTML pur peuvent stocker le script malicieux de l'attaquant. Quand les commentaires sont affichés, le script est exécuté et peut ensuite transmettre à l'attaquant les informations nécessaires pour accéder au compte de l'utilisateur. Cette méthode d'attaque est extrêmement courante et efficace, parce que l'attaquant n'a pas besoin d'avoir une relation directe avec les victimes.

  Alors que l'envoi de données avec `POST` ou `GET` est la source la plus commune de vulnérabilité XSS, toute donnée en provenance du navigateur web est potentiellement vulnérable (cela inclut l'affichage des données des cookies par le navigateur, ou les fichiers de l'utilisateur qui sont chargés et affichés).La meilleure défense contre les vulnérabilités XSS est de supprimer ou désactiver toutes les balises qui peuvent potentiellement contenir des instructions pour exécuter du code. Pour HTML cela inclut les tags comme `<script>`, `<object>`, `<embed>`, et `<link>`.

Il est nécessaire de traiter les données saisies par l'utilisateur pour être sûr qu'il ne puisse ni exécuter de scripts ni pertuber le fonctionnement normal du site (ce procédé est appelé *input sanitization* en anglais). De nombreux frameworks proposent par défaut cette vérification sur les entrées des formulaires.

1.4 La faille Include

Rappelez-vous bien que Include() fait deux choses : tout d'abord elle "rapatrie" le code contenu dans le fichier passé en paramètre, puis elle l'exécute. C'est justement l'exécution de ce code, liée à une autre subtilité, qui va constituer la faille. En eet, imaginez l'appel suivant sur une page index.php :

Ce bout de code regarde si la variable "page" est contenue dans l'URL, et si c'est le cas, elle inclut le chier de même nom que le contenu de la variable. Sinon, elle inclut une page par défaut. Par exemple, si vous appelez : index.php?page=test.php Le script inclura (et exécutera) la page "test.php".

Jusqu'ici, rien de bien compliqué. Maintenant, plaçons nous dans la peau d'un visiteur malveillant. Imaginons qu'il se soit rendu compte de la présence d'un appel à Inclure() par un quelconque moyen, et qu'il peut contrôler (comme c'est le cas ici) le paramètre passé. S'il appelle la page : index.php?page=http://www.google.fr Alors Google s'achera sur le site !
En effet, une subtilité d'include() permet d'inclure des pages ne se trouvant pas sur le même serveur que la page appelant ! A présent, le visiteur peut inclure n'importe quelle page. Elle sera de toute façon exécutée sur le serveur. Dans ce cas, il peut très bien placer une page PHP qu'il a fait lui-même sur son serveur personnel, et l'inclure ! Imaginez que cette page en question soit une backdoor...

L'inclusion se fera de la même manière, et il disposera d'une backdoor sur le site vulnérable ! Il pourra alors lister tous les cahiers, les éditer, en créer, accéder à la base de données, etc... Nous y reviendrons en temps voulu.

1.5 Falsification de requête

Les attaques CSRF permettent à un utilisateur malveillant d'éxécuter des actions à l' idée des identifiants d'un autre utilisateur sans que cet utilisateur ne soit informé ou consentant.
Ce type d'attaque s'explique mieux avec un exemple. John est l'utilisateur malveillant qui sait qu'un site particulier permet à des utilisateurs authentifiés d'envoyer de l'argent vers un compte particulier en utilisant des requêtes HTTP POST qui incluent le numéro de compte et le montant. John construit un formulaire qui inclut son numéro de compte et un montant dans des champs cachés (invisibles) et le transmet à un autre utilisateur du site (avec le bouton de validation déguisé en un lien vers un site "pour devenir riche".

Si un utilisateur clique sur le bouton de validation, une requête HTTP POST, contenant les informations de transaction, va être transmise au serveur ainsi que le cookie que le navigateur web associé au site (l'ajout à la requête du cookie associé au site est le comportement normal du navigateur). Le serveur va vérifier le cookie d'authentification, et l'utiliser pour déterminer si l'utilisateur est ou n'est pas connecté et donc permet ou non la transaction.

Au final, tout utilisateur qui va cliquer sur le bouton de validation, alors qu'il sera connecté sur le site d'échange d'argent, va autoriser la transaction. John va devenir riche !

Note : l'astuce ici est que John n'a pas besoin d'accéder aux cookies de l'utilisateur (ou à ses identifiants), le navigateur web stocke cette information et l'inclut automatiquement dans toute les requêtes destinées au serveur associé.

Un moyen de prévenir ce type d'attaque est que le serveur demande que chaque requête POST possède un secret généré par le serveur et spécifique à l'utilisateur (le secret serait transmis par le serveur lors de l'envoi du formulaire de transaction).

Cette approche empêche John de créer son propre formulaire car il n'est pas capable de connaître le secret que le serveur fournit à l'utilisateur. Même si il venait à trouver ce secret et créer un formulaire pour un utilisateur particulier, il ne pourrait pas utiliser ce formulaire pour attaquer d'autres utilisateurs

Les framework web incluent souvent des mécanismes afin de prévenir les attaques CSRF.

2.1 Test unitaire

En programmation informatique, le **test unitaire** (ou « **T.U.** », ou « **U.T.** » en anglais) ou **test de composants** est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module »).

Dans les applications non critiques, l'écriture des tests unitaires a longtemps été considérée comme une tâche secondaire.

Utilité

On écrit un test pour confronter une réalisation à sa spécification. Le test définit un critère d'arrêt (état ou sorties à l'issue de l'exécution) et permet de statuer sur le succès ou sur l'échec d'une vérification. Grâce à la spécification, on est en mesure de faire correspondre un état d'entrée donné à un résultat ou à une sortie.

Trouver les erreurs rapidement

La méthode XP préconise d'écrire les tests en même temps, ou même avant la fonction à tester (Test Driven Development). Ceci permet de définir précisément l'interface du module à développer. Les tests sont exécutés durant tout le développement, permettant de visualiser si le code fraîchement écrit correspond au besoin.

Sécuriser la maintenance

Lors d'une modification d'un programme, les tests unitaires signalent les éventuelles régressions. En effet, certains tests peuvent échouer à la suite d'une modification, il faut donc soit réécrire le test pour le faire correspondre aux nouvelles attentes, soit corriger l'erreur se situant dans le code.

Documenter le code

Les tests unitaires peuvent servir de complément à l'API, il est très utile de lire les tests pour comprendre comment s'utilise une méthode. De plus, il est possible que la documentation ne soit plus à jour, mais les tests eux correspondent à la réalité de l'application.

2.2 Test Fonctionnel

Un test fonctionnel est le test qui servira à tester **automatiquement** toutes les fonctionnalités de votre application. "Toutes", ça veut dire : les fonctionnalités qui étaient demandées dans le cahier des charges du projet (ou autres spécifications). Par exemple, vous pourrez avoir à tester qu'un membre peut bien s'inscrire, se connecter, se déconnecter…

Voici la liste des étapes qu'il faut respecter a minima pour effectuer un test fonctionnel :

1. Créer un client HTTP.
2. Effectuer une requête HTTP sur la page que nous devons tester.
3. S'assurer que les éléments sur la page testée sont bien présents (écrire des assertions).

Il est également possible d'établir un scénario d'utilisation pour arriver à un résultat attendu… En fait, il **faut** le faire : vous allez chercher à imiter le comportement d'un utilisateur lorsque celui-ci teste une fonctionnalité. Dans tous les cas, les assertions qu'il faudra écrire seront toujours liées aux besoins exprimés avant le développement.

Intérêts des tests fonctionnels

L'écriture de tests fonctionnels amène logiquement à connaître le fonctionnement d'une application. Au-delà de la connaissance du code, vous devrez connaître le contenu attendu de la réponse HTTP (ce qui sera affiché sur la page).

Allez, faisons une mise en situation.

À la découverte d'une application inconnue

Disons que vous arrivez sur un projet sur lequel on vous demande de travailler. L'une des premières étapes pour commencer à contribuer est de découvrir l'ensemble des pages web de ce site.

Dans le meilleur des cas, des tests fonctionnels ont été écrits, et cela vous permet d'avoir une documentation précise à disposition !

Dans le cas contraire, c'est une excellente raison d'établir une documentation.

Assurer sereinement les évolutions de l'application

Par ailleurs, au fur et à mesure que vous ferez évoluer l'application, les tests fonctionnels permettent de s'assurer que ces modifications n'impactent pas les autres fonctionnalités du site. Il n'y a rien de plus chronophage que d'avoir à afficher l'ensemble des pages d'un site web à la main pour s'assurer que ce que nous venons d'écrire n'a rien cassé !

Un test fonctionnel est un test automatisé. Vous allez tester une fonctionnalité, vous assurez à chaque lancement de l'ensemble des tests que celle-ci n'est pas altérée et qu'elle correspond bien aux besoins exprimés.

Principes des tests fonctionnels

Dans ce cours, nous allons écrire nos tests fonctionnels dans le cadre d'une application Symfony. Je vais vous fournir une application déjà développée, ne vous inquiétez pas. Je vais vous expliquer comment la tester fonctionnellement.

Les étapes pour écrire un test fonctionnel simple sont les suivantes :

1. Créer un client HTTP.
2. Émettre une requête HTTP à l'aide du client HTTP sur une page que vous souhaitez tester.
3. Récupérer la réponse HTTP.
4. Tester si le contenu attendu est bien présent.

Comment les mettre en place

De la même manière que pour les tests unitaires, vous pouvez implémenter les tests fonctionnels à différents moments du projet :

- avant d'implémenter une nouvelle fonctionnalité (TDD),
- juste après avoir implémenté une nouvelle fonctionnalité,
- au moment où vous découvrez une anomalie,
- au moment où vous

3 Les principales failles humaines

3.1 Fishing

C'est quoi?

Via votre messagerie ou votre boîte mail, certaines personnes mal intentionnées tentent de mettre la main sur vos données personnelles en utilisant des techniques d'hameçonnage (phishing) ou d'escroquerie de type fraude 419 (scam) ! Ces techniques d'attaque évoluent constamment. Les conseils suivants vous aideront à déterminer si un message est légitime ou non.

Comment repérer une arnaque reçue dans votre messagerie ou votre boîte mail ?

Est-ce que le message/courriel vous est réellement destiné ?
Généralement, les messages malveillants sont envoyés à destination d'un grand nombre de cibles, ils ne sont pas ou peu personnalisés.
Le message évoque un dossier, une facture, un thème qui ne vous parle pas ? Il s'agit certainement d'un courriel malveillant.
Attention aux expéditeurs inconnus : soyez particulièrement vigilants sur les courriels provenant d'une adresse électronique que vous ne connaissez pas ou qui ne fait pas partie de votre liste de contact.
Soyez attentif au niveau de langage du courriel : même si cela s'avère de moins en moins vrai, certains courriels malveillants ne sont pas correctement écrits. Si le message comporte des erreurs de frappe, des fautes d'orthographe ou des expressions inappropriées, c'est qu'il n'est pas l'œuvre d'un organisme crédible (banque, administration …).
Vérifiez les liens dans le courriel : avant de cliquer sur les éventuels liens, laissez votre souris dessus*. Apparaît alors le lien complet. Assurez-vous que ce lien est cohérent et pointe vers un site légitime. Ne faites pas confiance aux noms de domaine du type impots.gouv.fr, impots.gouv.fr.biz, infocaf.org au lieu de www.caf.fr.
** A noter : cette manipulation est impossible à effectuer depuis un écran de smartphone.*
Méfiez-vous des demandes étranges : posez-vous la question de la légitimité des demandes éventuelles exprimées. Aucun organisme n'a le droit de vous demander votre code carte bleue, vos codes d'accès et mots de

passe. Ne transmettez rien de confidentiel même sur demande d'une personne qui annonce faire partie de votre entourage.
L'adresse de messagerie source n'est pas un critère fiable : une adresse de messagerie provenant d'un ami, de votre entreprise, d'un collaborateur peut facilement être usurpée. Seule une investigation poussée permet de confirmer ou non la source d'un courrier électronique. Si ce message semble provenir d'un ami - par exemple pour récupérer l'accès à son compte - contactez-le sur un autre canal pour vous assurer qu'il s'agit bien de lui !

Comment réagir ?

Si vous avez un doute sur un message reçu, il y a de fortes chances que celui-ci ne soit pas légitime :

N'ouvrez surtout pas les pièces jointes et ne répondez-pas ;
Si l'escroquerie que vous souhaitez signaler vous est parvenue par un spam (pourriel), rendez-vous sur www.signal-spam.fr ;
Supprimez le message puis videz la corbeille ;
S'il s'agit de votre compte de messagerie professionnel : transférez-le au service informatique et au responsable de la sécurité des systèmes d'information de votre entreprise pour vérification. Attendez leur réponse avant de supprimer le courrier électronique.

Comment s'en prémunir ?

1-Utilisez un logiciel de filtre anti-pourriel ou activer l'option d'avertissement contre le filoutage présent sur la plupart des navigateurs.

2-Installez un anti-virus et mettez-le à jour.
Désactivez le volet de prévisualisation des messages.
Lisez vos messages en mode de texte brut.

3.2 Clé USB

Pourtant, les attaques informatiques provenant de périphériques USB sont de plus en plus nombreuses et devraient nous pousser à être vigilants !Seconds du classement des cybermenaces les plus dangereuses, les périphériques USB sont rapidement devenus un des principaux vecteurs d'attaque ! Or, malgré cette ascension, on minimise trop souvent le danger que représentent les clés USB (et autres périphériques).En effet, en 2016, un chercheur en sécurité de Google a tenté une expérience en abandonnant 300 clés USB sur un campus. Les résultats de cette recherche parlent d'eux-mêmes.

Elles ont presque toutes été ramassées par des étudiants, et la moitié d'entre elles a été insérée dans un ordinateur !Cela prouve donc que les périphériques USB sont encore trop peu perçus comme des menaces.

Quel danger représentent les clés USB ? Quels sont les risques dont il faut se méfier ?

Le contenu du périphérique

Lorsqu'on connecte une clé USB à votre ordinateur, on ne connaît pas forcément la totalité des fichiers qui y sont enregistrés. Parmi tous les fichiers présents, il pourrait se trouver un virus. Qu'on l'ait infectée volontairement ou non, dans tous les cas, ce virus peut infecter votre ordinateur**.**

Le comportement du périphérique

En plus du contenu de la clé USB, son comportement une fois connecté à votre ordinateur peut s'avérer lui-même dangereux.

Autorun

Les clés USB peuvent profiter d'une fonctionnalité offerte par les systèmes d'exploitation Windows : *l'autorun*. Cette fonctionnalité permet d'automatiser certaines tâches lors de la connexion, entre autres, d'une clé USB.
Il suffit d'un simple fichier déposé sur la clé USB pour exploiter cette fonctionnalité et exécuter une action malveillante. Cette action pourra , selon le cas, s'exécuter automatiquement dès la connexion de la clé, ou lors de son ouverture par l'utilisateur.

USB Killer

Il existe des clés USB qui permettent de « griller » les cartes mères des ordinateurs sur lesquels on les connecte. Ce sont les USB killer.
Leur fonctionnement est simple : elles contiennent des condensateurs et des convertisseurs de courant. Une fois branchée, la clé utilise le courant qu'elle reçoit pour charger les condensateurs.
Lorsque ces condensateurs sont chargés, ils permettent l'envoi d'une tension électrique élevée qui va griller les composants électriques de la carte mère. Ce cycle de charge/décharge est répété plusieurs fois par seconde jusqu'à ce que la carte mère ne fonctionne plus. En général, une fraction de seconde suffit pour que votre ordinateur ne soit plus utilisable.

BadUSB

La plupart des clés USB ont une faille appelée BadUSB. Cette faille permet de reprogrammer le firmware de la clé (programme intégré dans la clé USB lui permettant de fonctionner), afin de la faire passer pour un autre type de périphérique : clavier, carte réseau, lecteur CD, etc.
L'utilisation de cette faille peut permettre à un individu malveillant d'installer un virus sur votre ordinateur, de vous voler certaines informations – données sensibles, mots de passe, identifiants, etc. -, ou de verrouiller vos données.
Comment limiter, voire éviter ces dangers ?

Les conseils et bonnes pratiques pour éviter les risques

Certains réflexes limitent les risques d'être infecté par un virus informatique et amenuisent le danger que représentent les périphériques USB. Il est par exemple nécessaire d'avoir un ordinateur à jour, d'appliquer tous les correctifs de sécurité, de mettre à niveau son antivirus ou de forcer l'analyse lorsqu'on branche une clé USB.
Besoin d'un bon antivirus ? Nous vous conseillons *ESET Internet Security* pour protéger efficacement tous vos appareils !

OPPENS a testé et validé

Une fois la clé ouverte, il faut prendre soin de ne pas cliquer sur n'importe quel contenu. Il faut ouvrir de préférence des fichiers connus (photos, vidéos, PDF) et éviter les programmes dont vous ignorez le rôle.

Par ailleurs, la principale source d'infection chez les PC particuliers reste le piratage d'applications ou de jeux. Par conséquent, récupérer un jeu vidéo ou un logiciel « cracké par un ami » est (en plus d'être illégal et immoral) extrêment dangereux pour votre ordinateur qui risque d'avoir une infection virale !

Les mesures de sécurité pour faire face aux dangers des clés USB

Pour limiter les risques et le danger des périphériques USB, il faut s'assurer de bien gérer les droits utilisateurs. En effet, une clé USB branchée récupère les mêmes droits que l'utilisateur courant sous Windows (c'est-à-dire vous !). Utiliser au quotidien un compte utilisateur ne disposant pas des droits administrateur permet donc de limiter les actions que la clé USB peut effectuer sur le système.
Vous pouvez également désactiver la fonctionnalité « autorun ». Cette fonctionnalité n'est en général pas indispensable. La désactiver est une bonne pratique permettant d'empêcher l'exécution d'applications de manière automatique lors de la connexion d'une clé USB.
N'oubliez pas de verrouiller votre poste de travail ! Sous Windows, lorsqu'on verrouille l'écran, l'insertion d'un périphérique USB ne provoque pas son installation.
Si vous prenez pour habitude de verrouiller votre session lorsque vous quittez votre ordinateur, vous vous protégez partiellement contre les attaques par clé USB.
Enfin, cela va de soi, mais ne connectez pas de clé USB inconnue ! En évitant de connecter des clés USB inconnues sur votre poste de travail, vous vous protégez contre de nombreuses menaces – volontaires ou non -.

Quelles sont les bonnes pratiques d'utilisation des clés USB pour garantir la sécurité en entreprise ?

Il est conseillé de fournir des clés USB aux collaborateurs pour éviter qu'ils utilisent des clés USB de l'extérieur, et de renouveler cette flotte afin d'éviter la propagation de malwares.
Un autre bon réflexe à avoir est de ne pas brancher une clé USB dont on ne connaît pas la provenance à un poste de travail, et de contrôler systématiquement les clés USB dès lors qu'elles ont été branchées dans un environnement non sécurisé.
Enfin, pour veiller à la sécurité de votre entreprise, nous vous conseillons de brancher uniquement des clés préalablement décontaminées dans les PC industriels !
Plus que jamais, la sensibilisation des collaborateurs aux questions de sécurité informatique est vitale afin qu'ils appliquent les procédures mises en place.

4 Les liens utiles

4.1 Les attaques les plus courantes et leur parades:

https://www.chiny.me

https://developer.mozilla.org/fr/docs/Learn/Server-side/Premiers_pas/Website_security

4.2 Référencement des attaques:

https://en.wikipedia.org/wiki/Category:Web_security_exploits

https://owasp.org/www-community/attacks/

4.2 Référencement des test:

https://fr.wikipedia.org/wiki/Test_de_performance#:~:text=Cette%20d%C3%A9finition%20est%20donc%20tr%C3%A8s,la%20charge%20d'utilisateurs%20simultan%C3%A9s.