

Africa Verify

Rapport SEO

Réaliser par : Wassim Fathallah

Fais-le : 1-mars-21

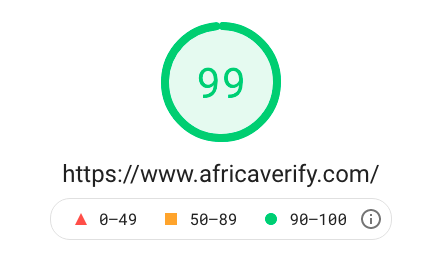
La vitesse du site web

La vitesse du site sur :

**Uptrends**

**page speed insights**

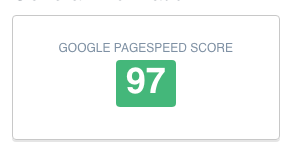
Ordinateur Mobile

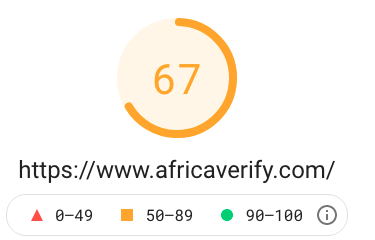


• 0 à 49 (rouge) : faible

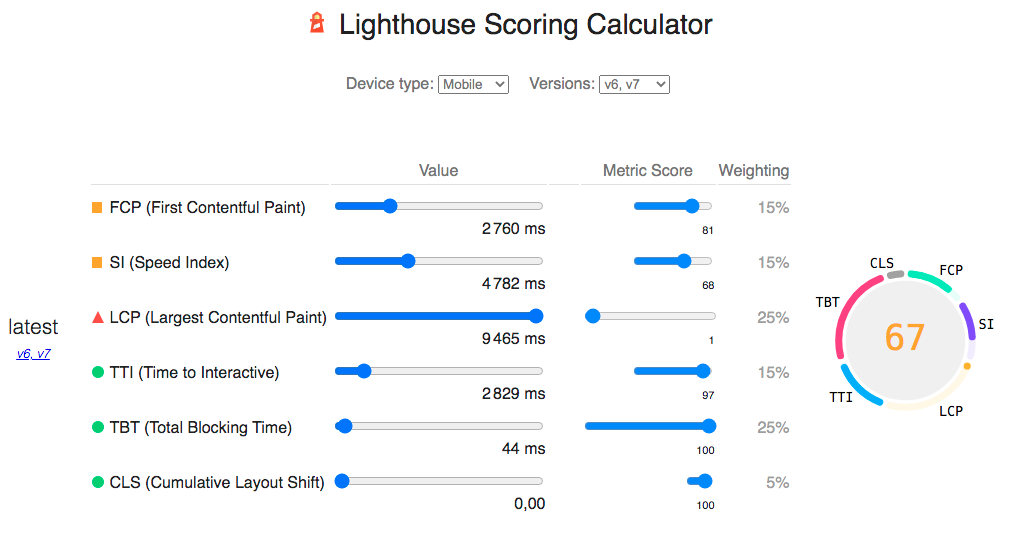
• 50 à 89 (orange) : à améliorer

• 90 à 100 (vert) : bon



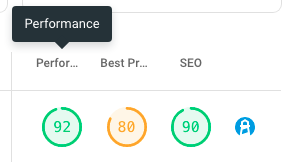


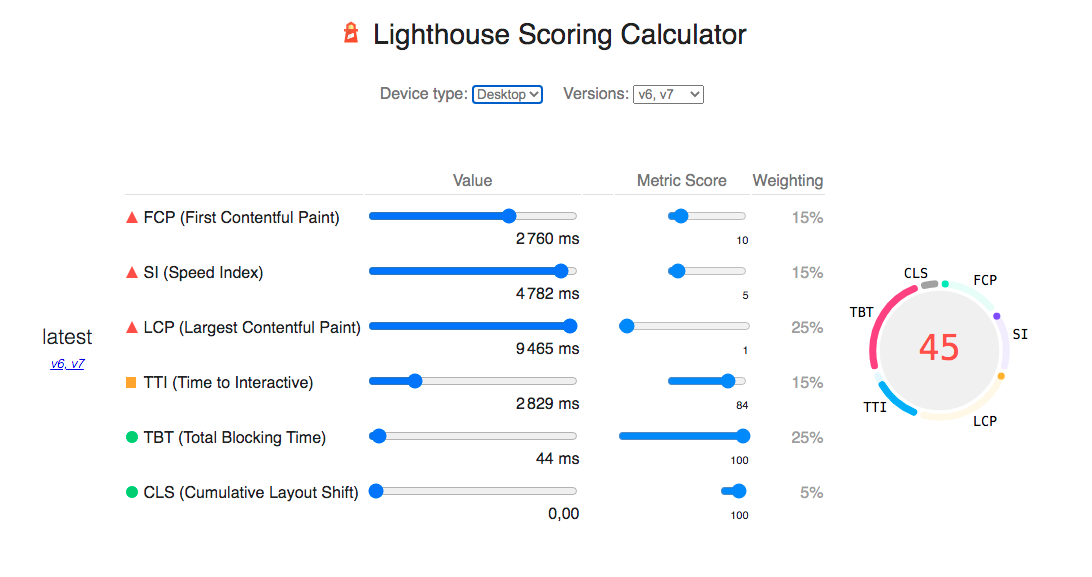
La Performance du site web



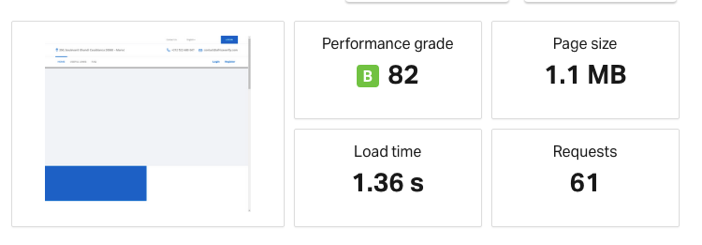
**Uptrends**

**Page speed insights**





**pingdom**



*Pour Améliorer les performances de la page*

|  |  |
| --- | --- |
| Le problème |  |
| La solution | * Faire moins de requêtes HTTP   La diminution du nombre de composants sur une page réduit le nombre de requêtes HTTP requises pour afficher la page, ce qui accélère le chargement des pages. Voici quelques moyens de réduire le nombre de composants: combiner des fichiers, combiner plusieurs scripts en un seul script, combiner plusieurs fichiers CSS dans une seule feuille de style et utiliser des sprites CSS et des images maps.   * Lorsque le navigateur demande une image statique et envoie des cookies avec la demande, le serveur ignore les cookies. Ces cookies sont du trafic réseau inutile. Pour contourner ce problème, assurez-vous que les composants statiques sont demandés avec des demandes sans cookie en créant un sous-domaine et en les y hébergeant. |

La Validation du site web

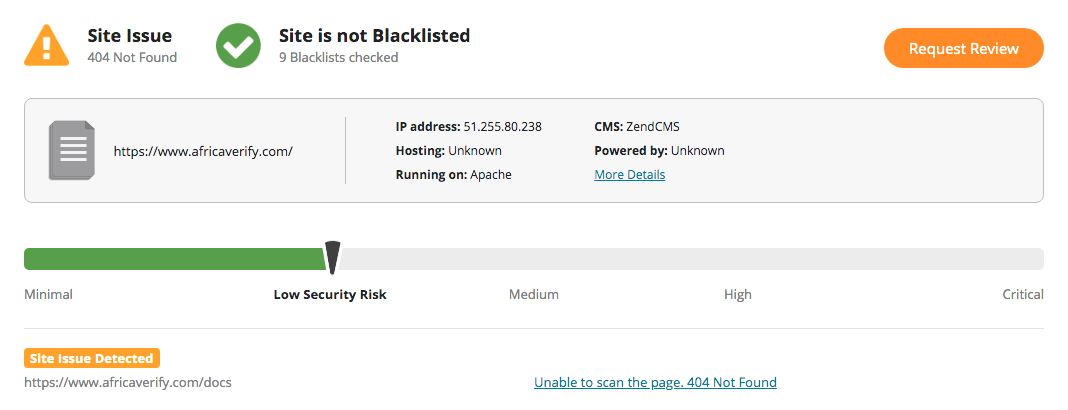
|  |  |
| --- | --- |
| Le Problème | La Solution |
|  | Le lien est rompu. Vérifiez à nouveau que vous n'avez commis aucune faute de frappe ou erreur de copier-coller. Si le lien pointe vers une ressource qui n'existe plus, vous souhaiterez peut-être supprimer ou corriger le lien. |
| Lines: 80, 604[**http://www.facebook.com/**](http://www.facebook.com/)  Lines: 83, 605[**http://www.twitter.com/**](http://www.twitter.com/) | Il faut lier Les liens avec le profil du site web |
|  | * Ajouter des balises d'en-tête : H6 * Jouter plus de texte sur le site. * Dupliquer les mots-clés au moins 3 fois * Mettre les mots clés en gras, en italique ou en soulignant. * Minimiser le nombre de requêtes HTTP. * Ajouter un bouton Tweet à le site Web. * Ajouter un bouton J'aime Facebook à le site Web. * Ajouter un bouton Google+ au site Web. * Valider le code HTML par le W3C. * Supprimer le JavaScript en ligne. * Supprimer le CSS en ligne. * Ajouter des balises ‘ alt ’ à toutes les images. * Supprimer les traits de soulignement des noms de fichiers d'image. |

Validation selon : validator.w3.org

La Sécurité du site web

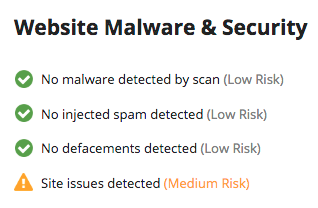
Selon le site : sitechecksucuri.net

L’analyse automatisée a détecté un problème sur certaines pages de notre site Web Africaverify



|  |  |
| --- | --- |
| Le Problème |  |
| La solution | Le lien :www.africaverify.com/docs  A aucune route ne correspond à la demande pour cela il faut le supprimer |

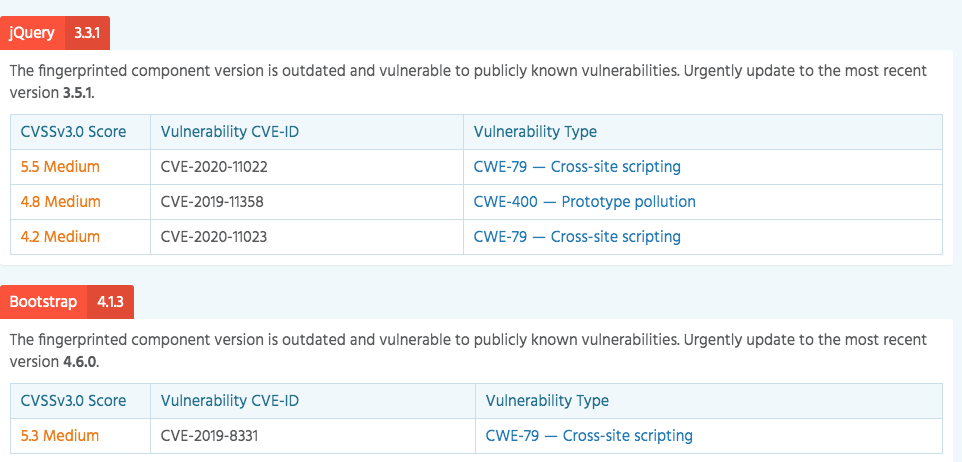
Statut du BlackList & malware & Security





Selon le site : immuniweb

L’analyse automatisée a détecté des vulnérabilités sur le site Web Africaverify



|  |  |
| --- | --- |
| Le Problème | La solution |
|  | Il faut changer la version du Bootstrap to Bootstrap V5.0.0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Le Problème | | | La solution | |
|  | | |  | |
|  | | |
|  | Pour résoudre ce problème voilà la solution :  https://resources.infosecinstitute.com/topic/securing-cookies-httponly-secure-flags/ | |

Selon le site : observatory.mozilla.org

|  |  |
| --- | --- |
| Le problème | La solution |
|  |  |

Selon un attacker : youssef\_OURGOUS

On trouvent :

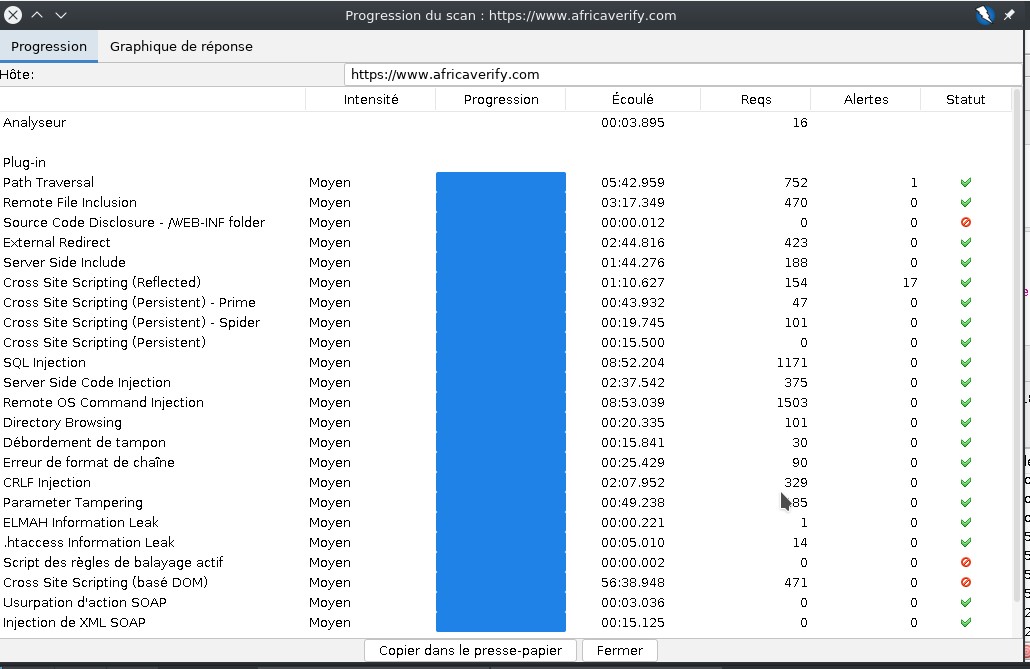
|  |  |
| --- | --- |
| Le problème | La solution |
|  | Avant d’envoyer un formulaire il faut vérifier les champs pour éviter les injection SQL.  Ou utiliser des ORM eloquent |

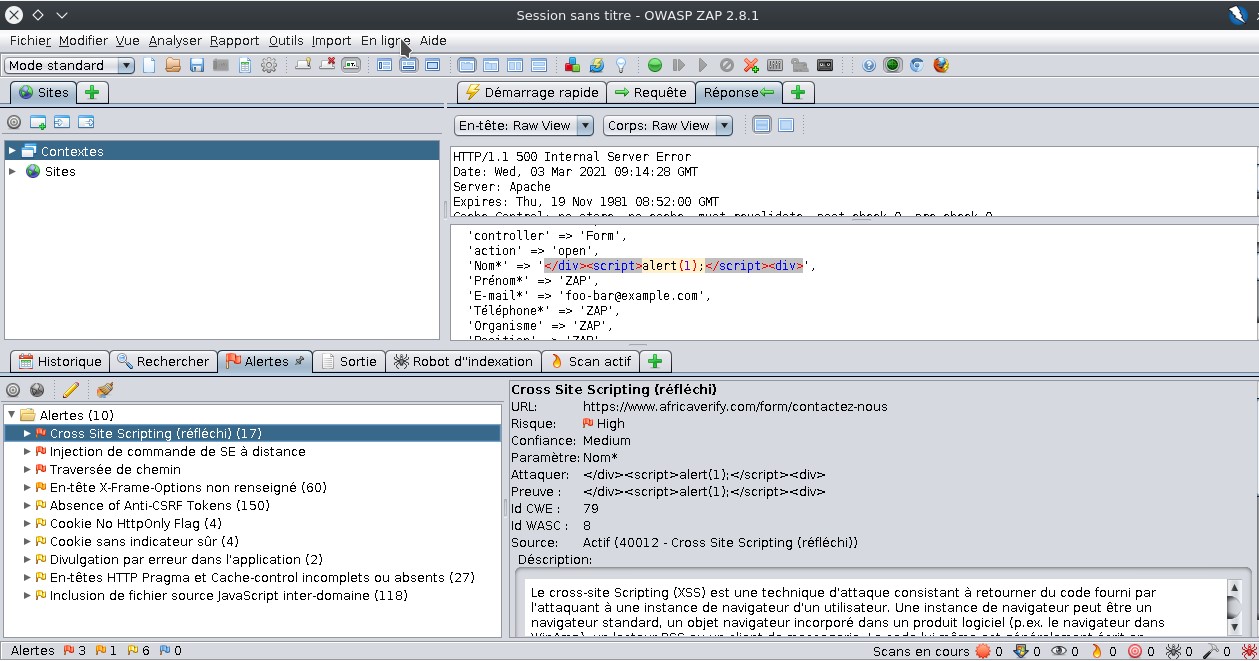
#### **Test de pénétration sur le site**

**Target site =🡺** [**www.africaverify.com**](http://www.africaverify.com)

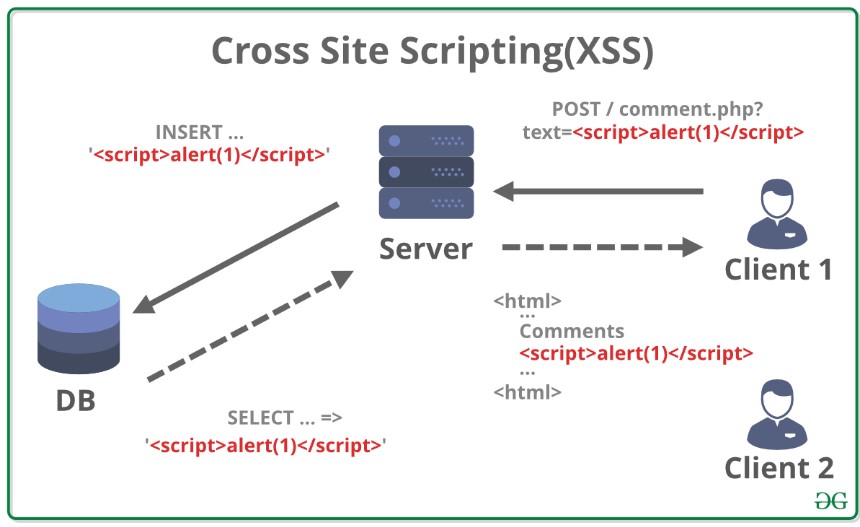
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **­OPENPORTS** | **STATE** | **SERVICE** | **VERTION** | **L'outil utilisé** | **Exploit** |  |
| 21 | open | ftp | Pure-FTPd | Nmap + Metasploit | Aucune charge utile configurée, par défaut sous Linux |  |
| 26 | open | smtp | Exim smypd 4.93 | Nmap + Metasploit | Exploit a échoué: une ou plusieurs options n'ont pas pu être validées |  |
| 53 | open | domain | ISC BIND 9.11.4-P2 (RHEL’s) | Nmap + Metasploit | Échec de l'opération |  |
| 80 | open | http | Apache httpd | Nmap + Metasploit | Ce serveur est configuré pour l'anonymat ne peut uniquement et le code de porte dérobée pas être atteint |  |
| 110 | open | pop3 | Dovecot pop3d | Nmap + Metasploit | Échec de l'opération |  |
| 143 | open | imap | Dovecot imapd | Nmap + Metasploit | Échec de l'opération |  |
| 443 | open | ssl/http | Apache httpd | Nmap + Metasploit | exploit abandonné en raison d'un échec: non vulnérable: la cible n'est pas vulnérable |  |
| 465 | open | ssl/smtp | Exim smtpd 4.93 | Nmap + Metasploit |  |  |
| 587 | open | smtp | Exim smtpd 4.93 | Nmap + Metasploit |  |  |
| 993 | open | ssl/imaps? | vide | Nmap + Metasploit |  |  |
| 995 | open | ssl/pop3s? | vide | Nmap + Metasploit |  |  |
| 3306 | open | mysql | MySQL 5.5.5-10.1.48-MariaDB | Nmap + Metasploit |  |  |
| 5432 | open | postgresql | PostgreSQL DB | Nmap + Metasploit |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

* Vérification complète du site par outil OWASP ZAP 2.8.1
* Résultats de scan





* Cross site scripting (réfléchi)(17)



<Déscription>

Le cross-site Scripting (XSS) est une technique d'attaque consistant à retourner du code fourni par l'attaquant à une instance de navigateur d'un utilisateur. Une instance de navigateur peut être un navigateur standard, un objet navigateur incorporé dans un produit logiciel (p.ex. le navigateur dans WinAmp), un lecteur RSS ou un client de messagerie. Le code lui-même est généralement écrit en HTML/JavaScript, mais peut être aussi en VBScript, ActiveX, Java, Flash ou toute autre technologie supportée par les navigateurs.

Lorsqu'un pirate parvient à faire exécuter son code par le navigateur d'un utilisateur, le code s'exécute dans le contexte de sécurité (ou zone) du site web hébergeur. Avec ce niveau de privilège, le code a la capacité de lire, modifier et transmettre toutes les données sensibles accessibles par le navigateur. Un utilisateur sujet au Cross-site Scripting pourrait voir son compte piraté (vol de cookie), son navigateur redirigé vers un autre site, ou éventuellement voir apparaître du contenu frauduleux envoyé par le site internet qu'ils visitent. Les attaques par Cross-site Scripting compromettent fondamentalement la relation de confiance entre un utilisateur et le site web. Les applications utilisant des instances d'objet de navigateur, et qui chargent du contenu depuis le système de fichiers, peuvent exécuter du code dans le périmètre de confiance de l'ordinateur, mettant ainsi en danger le système.

Il existe trois types d'attaques par Cross-site Scripting : non persistant, persistant et basé sur les DOM.

Les attaques non persistantes et celles basées sur les DOM nécessitent qu'un utilisateur visite un lien incorporant un code malveillant, ou visite une page web malveillante contenant un formulaire internet, qui, une fois publié sur le site vulnérable, permettra l'attaque proprement dite. Un formulaire malveillant sera souvent utilisé lorsque la ressource vulnérable n'accepte que les requêtes HTTP POST. Dans un tel cas, le formulaire peut être envoyé automatiquement, sans que la victime n'en ait conscience (p. ex. à l'aide de JavaScript). En cliquant sur le lien malveillant ou en soumettant le formulaire malveillant, le code XSS s'affichera en retour et sera interprété et exécuté par le navigateur de l'utilisateur. Une autre technique pour envoyer des requêtes presque arbitraires (GET et POST) consiste à utiliser un client intégré, tel que Adobe Flash.

Les attaques persistantes se produisent lorsque le code malveillant est envoyé à un site internet où il est stocké pendant un certain temps. Des exemples de cibles favorites des pirates sont les publications sur les tableaux de messages, les messages de courrier électronique et les logiciels de tchat internet. Aucune interaction avec un quelconque site/lien supplémentaire (par exemple, un site de pirate ou un lien malveillant envoyé par e-mail) n'est requis de la part de l'utilisateur sans méfiance, il suffit de visualiser la page internet contenant le code.

<Solution>

Phase: Architecture et Design

Utilisez une librairie ou un framework approuvé qui ne permet pas cette vulnérabilité, ou qui contient des fonctionnalités permettant d'éviter plus facilement cette vulnérabilité.

Quelques exemples de bibliothèques et de frameworks facilitant l'encodage approprié des pages internet sont: la bibliothèque Microsoft Anti-XSS, le module d'encodage OWASP ESAPI, et Apache Wicket.

Phases: Implémentation; Architecture et Design

Tenez compte du contexte dans lequel vos données seront utilisées et de l'encodage qui sera attendu. Ceci est particulièrement important lors de la transmission de données entre les différents composants, ou lors de la génération de sorties qui peuvent contenir plusieurs encodages en même temps, comme des pages internet ou des messages électroniques multi-parties. Étudiez tous les protocoles de communication et les représentations de données attendus pour déterminer les stratégies de codage requis.

Pour toutes les données qui seront affichées sur une autre page web, en particulier toutes les données qui ont été reçues de l'extérieur, utilisez l'encodage approprié pour tous les caractères non alphanumériques.

Consultez le XSS Prevention Cheat Sheet pour plus de détails sur les types d'encodage et d'échappement dont vous pouvez avoir besoin...

Phase : Architecture et Design

Pour tous les contrôles de sécurité effectués du côté client, veillez à ce que ces contrôles soient réitérés du côté serveur également, afin d'éviter la faille CWE-602. Les agresseurs peuvent contourner les contrôles côté client en modifiant les valeurs après que ces contrôles aient été effectués, ou en changeant le client afin d'en retirer complètement les contrôles. Ensuite, ces valeurs modifiées seraient soumises au serveur.

Le cas échéant, utilisez des mécanismes structurés appliquant automatiquement la séparation entre le code et les données. Ces mécanismes peuvent être en mesure de fournir automatiquement la présentation, le codage et la validation adéquat, au lieu de se fier au développeur pour réaliser ces fonctionnalités pour chaque champ de sortie généré.

Phase: Implémentation

Pour chaque page internet générée, utilisez et spécifiez un encodage de chaînes de caractères comme ISO-8859-1 ou UTF-8. Quand aucun codage n'est spécifié, le navigateur web tente de deviner quel encodage est effectivement utilisé dans la page web et peut choisir un codage erroné. Ceci peut pousser le navigateur internet à traiter spécialement certaines séquences, exposant ainsi le client à de subtiles attaques XSS. Voir CWE-116 pour des mesures de réduction de risque plus liées à l'encodage / l'échappement.

Pour aider à réduire les attaques XSS contre le cookie de session de l'utilisateur, fixez la valeur du cookie à HttpOnly. Dans les navigateurs prennant en charge la fonctionnalité HttpOnly (telles que les versions les plus récentes d'Internet Explorer et de Firefox), cet attribut peut éviter que le cookie de session de l'utilisateur soit accessible à des scripts malveillants utilisant document.cookie côté client. Ce n'est toutefois pas une solution complète, car HttpOnly n'est pas supporté par tous les navigateurs. Plus important encore, XMLHTTPRequest et d'autres puissantes technologies de navigateur fournissent un accès en lecture aux en-têtes HTTP, y compris à l'en-tête Set-Cookie, dans laquelle la balise HttpOnly est définie.

Supposez que toutes les entrées sont malveillantes. Utilisez une stratégie de validation des entrées basée sur le principe "n'accepter que le bon", c'est-à-dire utilisez une liste blanche des entrées acceptables se conformant strictement aux spécifications. Rejetez toute entrée ne se conformant pas strictement aux spécifications, ou transformez-la en une valeur qui soit conforme. Ne vous fiez pas exclusivement à la recherche d'entrées malveillantes ou incorrectes (par exemple, ne comptez pas sur une liste noire). Toutefois, les listes noires peuvent être utiles pour détecter les attaques potentielles ou pour déterminer quelles entrées sont si mal formées qu'elles devraient être rejetées purement et simplement.

Lorsque vous effectuez une validation d'entrée, considérez toutes les propriétés potentiellement pertinentes, comme la longueur, le type d'entrée, la gamme complète des valeurs acceptables, les entrées manquantes ou défectueuses, la syntaxe, la cohérence dans des domaines connexes et la conformité aux règles métier. Comme exemple de logique de règle métier, "bateau" peut être syntaxiquement valide car elle ne contient que des caractères alphanumériques, mais elle n'est pas valide si des couleurs sont attendues, telles que "rouge" ou "bleu".

Veillez à effectuer une validation d'entrée à des interfaces bien définies au sein de l'application. Ceci aidera à protéger l'application, même si un composant est réutilisé ou déplacé ailleurs.

🡺 Injection de commande de SE à distance

Déscription :

Technique d'attaque utilisée pour l'exécution non autorisée de commandes du système d'exploitation. Cette attaque est possible lorsqu'une application accepte des entrées douteuses pour générer des commandes de système d'exploitation vulnérables, sans assainissement correct des données et/ou sans appel correct de programmes externes.

Solution:

Si possible, utilisez des appels de bibliothèque plutôt que des processus externes pour recréer les fonctionnalités souhaitées.

Exécutez votre code dans un "bac à sable" ou un environnement similaire, qui impose des limites strictes entre le processus et le système d'exploitation. Cela peut restreindre efficacement quels fichiers sont accessibles dans un répertoire particulier ou lesquels peuvent être exécutés par votre logiciel.

Au niveau du système d'exploitation, on peut citer les exemples pour Unix: chroot, AppArmor et SELinux. En général, le code géré (managed code) peut offrir une certaine protection. Par exemple, java.io.FilePermission dans le SecurityManager de Java permet de spécifier des restrictions sur les opérations de fichiers.

Ceci peut ne pas être une solution adéquate. Par ailleurs, elle limite l'impact au seul système d'exploitation; le reste de votre application peut encore faire l'objet de vulnérabilités.

Autant que possible, ne donnez pas à une tierce partie le contrôle sur des données utilisées pour générer une commande à exécuter. Par exemple, dans les applications internet, il vaudra mieux stocker la commande localement dans la session plutôt que l'envoyer au client dans un champ de formulaire masqué.

Utilisez une bibliothèque ou un framework certifié qui ne permet pas à cette faille d'apparaitre ou qui fournit des mécanismes aidant à éviter cette faille.

Par exemple, envisagez d'utiliser le contrôle d'encodage ESAPI ou un outil, une bibliothèque ou un framework semblable. Ceux-ci aideront le programmeur à encoder les sorties d'une manière moins sujette aux erreurs.

Si vous avez besoin d'utiliser des chaînes de requête ou des commandes générées dynamiquement en dépit du risque, insérez correctement les arguments entre des guillemets et échappez tous les caractères spéciaux dans ces arguments. L'approche la plus prudente consiste à échapper ou à filtrer tous les caractères ne figurant pas dans une liste blanche extrêmement stricte (par exemple, tout ce qui n'est pas un caractère alphanumérique ou un espace). Si certains caractères spéciaux sont encore nécessaires, tels que des espaces, entourez chaque argument par des guillemets après l'étape de filtrage ou d'échappement. Prenez garde à l'injection d'arguments.

Si le programme à exécuter permet de spécifier les arguments dans un fichier d'entrée ou par l'entrée standard, alors envisagez d'utiliser ce même mode pour passer des arguments plutôt que par la ligne de commande.

Le cas échéant, utilisez des mécanismes structurés appliquant automatiquement la séparation entre le code et les données. Ces mécanismes peuvent être en mesure de fournir automatiquement la présentation, le codage et la validation adéquat, au lieu de se fier au développeur pour réaliser ces fonctionnalités pour chaque champ de sortie généré.

Certains langages offrent de multiples fonctions pouvant être utilisées pour lancer des commandes. Si possible, identifiez toute fonction appelant un interpréteur de commandes à l'aide d'une seule chaîne, et remplacez-la par une fonction requérant des arguments individuels. Ces fonctions effectuent généralement une mise entre guillemets et un filtrage approprié des arguments. Par exemple, en C, la fonction system() accepte une chaîne qui contient la commande entière à exécuter, tandis que execl(), execve() et d'autres demandent un tableau de chaînes, une pour chaque argument. Dans Windows, CreateProcess() accepte uniquement une seule commande à la fois. En Perl, si un tableau d'arguments est fourni à la fonction system(), alors la fonction mettra entre guillemets chacun des arguments.

Supposez que toutes les entrées sont malveillantes. Utilisez une stratégie de validation des entrées basée sur le principe "n'accepter que le bon", c'est-à-dire utilisez une liste blanche des entrées acceptables se conformant strictement aux spécifications. Rejetez toute entrée ne se conformant pas strictement aux spécifications, ou transformez-la en une valeur qui soit conforme. Ne vous fiez pas exclusivement à la recherche d'entrées malveillantes ou incorrectes (par exemple, ne comptez pas sur une liste noire). Toutefois, les listes noires peuvent être utiles pour détecter les attaques potentielles ou pour déterminer quelles entrées sont si mal formées qu'elles devraient être rejetées purement et simplement.

Lorsque vous effectuez une validation d'entrée, considérez toutes les propriétés potentiellement pertinentes, comme la longueur, le type d'entrée, la gamme complète des valeurs acceptables, les entrées manquantes ou défectueuses, la syntaxe, la cohérence dans des domaines connexes et la conformité aux règles métier. Comme exemple de logique de règle métier, "bateau" peut être syntaxiquement valide, car la chaîne ne contient que des caractères alphanumériques, mais elle n'est pas valide si vous attendez des couleurs telles que "rouge" ou "bleu".

Lors de la construction des chaînes de commande d'OS, utilisez des listes blanches rigoureuses qui limitent le jeu de caractères en fonction de la valeur attendue du paramètre dans la requête. Vous limiterez ainsi indirectement la portée d'une attaque, mais cette technique est moins importante qu'un bon encodage et un bon échappement.

Notez que le codage de sortie, l'échappement, et la mise entre guillemets est la solution la plus efficace pour prévenir les injections de commandes d'OS, même si la validation d'entrée peut fournir une certaine défense en profondeur. La raison est qu'elle limite efficacement ce qui apparaîtra en sortie. La validation des entrées n'empêchera pas toujours l'injection de commande d'OS, surtout si vous êtes tenu de supporter des champs de texte libre qui peuvent contenir des caractères arbitraires. Par exemple, lors de l'appel à un logiciel de messagerie, vous devrez accepter que le champ objet contienne des entrées potentiellement dangereuses, comme les caractères ";" et ">", qui devraient être échappés ou traités autrement. Dans ce cas, supprimer ce caractère pourrait réduire le risque d'injection de commande d'OS, mais il produirait un comportement incorrect, parce que le champ objet ne serait pas traité de la façon dont l'utilisateur l'attend. Cela peut sembler être un inconvénient mineur, mais il pourrait être plus important lorsque le programme s'appuie sur des lignes d'objet bien structuré afin de transmettre des messages à d'autres composants.

Même si vous faites une erreur dans votre validation (par exemple, oublier un champ de saisie parmi 100 autres), un codage approprié sera toujours susceptible de vous protéger contre les attaques basées sur l'injection. Tant qu'elle n'est pas utilisée seule, la validation des entrées reste une technique utile, car elle peut significativement réduire votre surface d'attaque, vous permettre de détecter certaines attaques et fournir d'autres bénéfices de sécurité que même un codage approprié n'apporte pas.

🡺Traversée de chemin:

Déscription:

La technique d'attaque par traversée de chemin d'accès (path traversal) permet à un pirate d'accéder aux fichiers, répertoires et commandes résidant potentiellement en dehors du répertoire racine des documents internet. Un agresseur peut manipuler une URL de telle sorte que le site internet exécutera ou révélera le contenu de fichiers

arbitraires situés n'importe où sur le serveur internet. Tout dispositif qui expose une interface HTTP est potentiellement vulnérable à la traversée de chemin.

La plupart des sites internet restreignent l'accès de l'utilisateur à une partie spécifique du système de fichiers, généralement appelée la "racine des documents internet" ou le répertoire "Racine CGI". Ces répertoires contiennent les fichiers destinés à l'accès des utilisateurs et les exécutables nécessaires au fonctionnement des applications internet. Pour accéder aux fichiers ou exécuter des commandes situées n'importe où dans le système de fichiers, les attaques par traversée de chemin utiliseront les capacités de séquences de caractères spéciaux.

L'attaque par traversée de chemin la plus élémentaire utilise la séquence de caractères spéciaux "../ " pour modifier l'emplacement de la ressource demandée dans l'URL. Bien que la plupart des serveurs internet les plus populaires empêcheront de sortir de la racine des documents internet par cette technique, des encodages alternatifs de la séquence "../" peuvent aider à contourner les filtres de sécurité. Ces variations de la méthode incluent le codage Unicode valide et non valide ("..%u2216" ou "..%c0%af ») de la barre oblique, des barres obliques inverses ("..\") sur des serveurs Windows, des caractères en codage d'URL ("%2e%2e%2f"), et le double codage d'URL ("..%255c") de la barre oblique.

Solution:

Supposez que toutes les entrées sont malveillantes. Utilisez une stratégie de validation des entrées basée sur le principe "n'accepter que le bon", c'est-à-dire utilisez une liste blanche des entrées acceptables se conformant strictement aux spécifications. Rejetez toute entrée ne se conformant pas strictement aux spécifications, ou transformez-la en une valeur qui soit conforme. Ne vous fiez pas exclusivement à la recherche d'entrées malveillantes ou incorrectes (par exemple, ne comptez pas sur une liste noire). Toutefois, les listes noires peuvent être utiles pour détecter les attaques potentielles ou pour déterminer quelles entrées sont si mal formées qu'elles devraient être rejetées purement et simplement.

Lorsque vous effectuez une validation d'entrée, considérez toutes les propriétés potentiellement pertinentes, comme la longueur, le type d'entrée, la gamme complète des valeurs acceptables, les entrées manquantes ou défectueuses, la syntaxe, la cohérence dans des domaines connexes et la conformité aux règles métier. Comme exemple de logique de règle métier, "bateau" peut être syntaxiquement valide car l'entrée ne contient que des caractères alphanumériques, mais elle n'est pas valide si vous attendez des couleurs telles que "rouge" ou "bleu".

Pour les noms de fichiers, utilisez des listes blanches rigoureuses qui limitent le jeu de caractères à utiliser. Si possible, ne permettez qu'un seul caractère "." dans le nom de fichier pour éviter les failles et excluez les séparateurs de répertoire comme "/". Utiliser une liste blanche des extensions de fichier autorisées.

Avertissement: si vous essayez de nettoyer vos données, faites-le alors de manière à ce que le résultat final ne soit pas sous une forme qui puisse être dangereuse. Un mécanisme d'assainissement peut supprimer des caractères tels que '.' et ';', qui peuvent être nécessaires pour certains exploits. Un agresseur peut essayer de tromper le mécanisme d'assainissement en transformant les données sous une forme dangereuse. Supposons que le pirate injecte un '.' à l'intérieur d'un nom de fichier (par exemple "sensi.tiveFile"), et le mécanisme d'assainissement supprimera le caractère, résultant en un nom de fichier valide, "sensitiveFile". Si les données d'entrée sont maintenant supposées sûres, alors le fichier peut être compromis.

Les entrées devraient être décodées et rendues canoniques selon la représentation interne actuelle de l'application avant d'en effectuer la validation. Assurez-vous que votre application ne décode pas la même entrée deux fois. De telles erreurs pourraient être utilisées pour contourner les contrôles par liste blanche en introduisant des entrées dangereuses après qu'elles aient été contrôlées.

Utilisez une fonction prédéfinie de canonisation du chemin (par exemple la fonction realpath() en C), qui produit la version canonique du nom de chemin, qui élimine efficacement les séquences ".." et les liens symboliques.

Exécutez votre code à l'aide des droits d'accès les plus réduits possible pour accomplir les tâches nécessaires. Si possible, créez des comptes isolés avec des privilèges limités, qui ne sont utilisés que pour une seule tâche. De cette façon, une attaque réussie ne donnera pas immédiatement accès au reste du logiciel ou de son environnement à l'agresseur. Par exemple, les applications de base de données ne nécessitent que rarement de fonctionner en tant qu'administrateur de base de données, en particulier dans les activités quotidiennes.

Lorsque l'ensemble des objets admissibles, tels que les noms de fichier ou les URLs, est limité ou connu, créez une correspondance à partir d'un ensemble de valeurs d'entrée fixe (tels que des ID numériques) sur les noms de fichiers réels ou sur les URLs, et rejetez toute autre entrée.

Exécutez votre code dans un "bac à sable" ou un environnement similaire, qui impose des limites strictes entre le processus et le système d'exploitation. Cela peut restreindre efficacement quels fichiers sont accessibles dans un répertoire particulier ou lesquels peuvent être exécutés par votre logiciel.

Au niveau du système d'exploitation, on peut citer les exemples pour Unix: chroot, AppArmor et SELinux. En général, le code géré (managed code) peut offrir une certaine protection. Par exemple, java.io.FilePermission dans le SecurityManager de Java permet de spécifier des restrictions sur les opérations de fichiers.

Ceci peut ne pas être une solution adéquate. Par ailleurs, elle limite l'impact au seul système d'exploitation; le reste de votre application peut encore faire l'objet de vulnérabilités.

🡺En-téte X-Frame-Option non renseigné

Déscription:

L'en-tête X-Frame-Options n'est pas incluse dans la réponse HTTP pour protéger contre les attaques de 'ClickJacking'.

Solution:

La plupart des navigateurs internet modernes supportent l'en-tête HTTP X-Frame-Options. Assurez-vous que celle-ci est renseignée sur toutes les pages internet retournées par votre site (si cette page ne doit être encadrée que par des pages de votre serveur (p.ex. fait partie d'un FRAMESET), alors utilisez la valeur SAMEORIGIN, sinon choisissez la valeur DENY quand la page n'est jamais encadrée. ALLOW-FROM permet à des sites spécifiques d'encadrer la page, pour autant que le navigateur le supporte).

🡺Absence Of Anti-CSRF Tokens

Déscription:

No Anti-CSRF tokens were found in a HTML submission form.

La contrefaçon de requête intersites (Cross Site Request Forgery - CSRF) est une attaque qui consiste à forcer une victime à envoyer une requête HTTP vers une destination cible, sans qu'elle n'en aie ni connaissance ni intention, afin d'effectuer une action en se faisant passer pour la victime. La cause originelle est que les fonctionnalités de l'application sont appelées à l'aide d'URL ou d'actions de formulaires prévisibles et reproductibles. La nature de l'attaque est que le CSRF exploite la confiance qu'un site internet accorde à un utilisateur. En revanche, le cross-site scripting (XSS) exploite la confiance que l'utilisateur porte à un site internet. Comme XSS, les attaques CSRF ne sont pas nécessairement multi-sites, mais elles peuvent l'être. La contrefaçon de requête intersite est également connue sous les noms CSRF, XSRF, attaque en un clic (one-click attack), session riding, confused deputy et sea surf.

Les attaques CSRF sont efficaces dans de nombreuses situations, notamment:

\* quand la victime a une session active sur le site cible.

\* quand la victime est authentifiée via HTTP auth sur le site cible.

\* quand la victime est sur le même réseau local que le site cible.

CSRF a d'abord été utilisée pour effectuer une action contre un site cible en utilisant les privilèges de la victime, mais des techniques récentes permettent d'avoir accès à des renseignements en accédant à la réponse. Le risque de divulgation d'informations est considérablement augmenté lorsque le site cible est vulnérable aux XSS, parce que XSS peut être utilisé comme une plateforme pour CSRF, permettant à l'attaque d'opérer dans les limites de la politique de même origine.

Autre informations:

No known Anti-CSRF token [anticsrf, CSRFToken, \_\_RequestVerificationToken, csrfmiddlewaretoken, authenticity\_token, OWASP\_CSRFTOKEN, anoncsrf, csrf\_token, \_csrf, \_csrfSecret] was found in the following HTML form: [Form 1: "emailsubscribe" "subscriber" ].

solution:

Phase: Architecture et Design

Utilisez une librairie ou un framework approuvé qui ne permet pas cette vulnérabilité, ou qui contient des fonctionnalités permettant d'éviter plus facilement cette vulnérabilité.

Utilisez par exemple des librairies anti-CSRF telles que CSRFGuard de l'OWASP.

Phase: Implémentation

Assurez-vous que votre application soit exempte de problèmes de cross-site scripting, parce que la plupart des défenses contre le CSRF peuvent être contournées en utilisant des scripts contrôlés par le pirate.

Phase: Architecture et Design

Générez une valeur à usage unique pour chaque formulaire, placez-la dans le formulaire et vérifiez-la à la réception du formulaire. Assurez-vous que cette valeur unique ne soit pas prévisible (CWE-330).

Notez que ceci peut aussi être contourné en utilisant XSS.

Identifiez les opérations particulièrement dangereuses. Quand l'utilisateur exécute une opération dangereuse, envoyez une requête de confirmation distincte pour vérifier que l'utilisateur veut effectivement effectuer cette opération.

Notez que ceci peut aussi être contourné en utilisant XSS.

Utilisez la librairie de gestion de session ESAPI.

Cette librairie comprend un composant pour le contrôle de CSRF.

N'utilisez pas la méthode GET pour les requêtes entraînant un changement d'état.

Phase: Implémentation

Vérifiez l'en-tête HTTP Referer pour voir si la requête provient d'une page attendue. Ceci pourrait toutefois restreindre la fonctionnalité de l'application, car les utilisateurs ou les serveurs proxy pourraient avoir désactivé le renvoi du HTTP Referer pour des raisons de confidentialité.

+Cookie No HttpOnly flag :

Déscription:

Un cookie a été défini sans le flag HttpOnly, ce qui signifie que le cookie peut être accessible via JavaScript. Si un script malveillant peut être exécuté sur cette page, alors le cookie sera accessible et pourra être transmis à un autre site. Si il s’agit d’un cookie de session, alors le détournement de session pourra être possible.

solution:

Veiller que le flag HttpOnly soit défini pour tous les cookies.

+Cookie sans indicateur :

Déscription :

Un cookie a été défini sans le flag sécurisé, ce qui signifie que le cookie est accessible via des connexions non chiffrées.

solution :

Chaque fois qu’un cookie contient des informations sensibles ou est un jeton de session, il doit toujours être passé à l’aide d’un canal chiffré. Veiller à ce que l’indicateur sécurisé (secure flag) soit défini pour les cookies contenant des informations sensibles.

+ divulgation par erreur dans l'application:

Déscription:

Cette page contient un message d’erreur/avertissement qui peut divulguer des informations sensibles, comme l’emplacement du fichier qui a produit l’exception non gérée. Cette information peut servir à lancer d'autres attaques contre l’application web. L’alerte pourrait être un faux positif si le message d’erreur se trouve à l’intérieur d’une page de documentation.

solution:

Examinez le code source de cette page. Mettez en place des pages d’erreur personnalisées. Envisagez d’implémenter un mécanisme fournissant une référence/identificateur unique de l'erreur au client (navigateur), tout en journalisant les détails du côté serveur, les cachant ainsi à l’utilisateur.

+En-tétes HTTP Pragma et Cache-control incomplets ou absents:

Déscription:

Les en-têtes HTTP cache-control et pragma n’ont pas été définis correctement ou manquent, permettant au navigateur et aux proxys de mettre en cache le contenu.

solution:

Chaque fois que possible, assurez-vous que l'en-tête HTTP cache-control est renseignée à no-cache, no-store, must-revalidate, private; et que l’en-tête pragma HTTP est renseignée à no-cache.

+Inclusion de fichier source JavaScript inter-domaine

Déscription:

La page inclut un ou plusieurs fichiers de script d’un autre domaine.

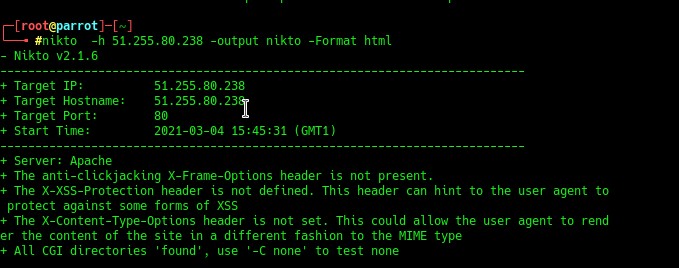
solution:

S’assurer que les fichiers source JavaScript soient chargés uniquement à partir de sources fiables, et que les sources ne peuvent être contrôlés par les utilisateurs de l’application.

* Date de fin de validité SSL



Scan<NIKTOR >

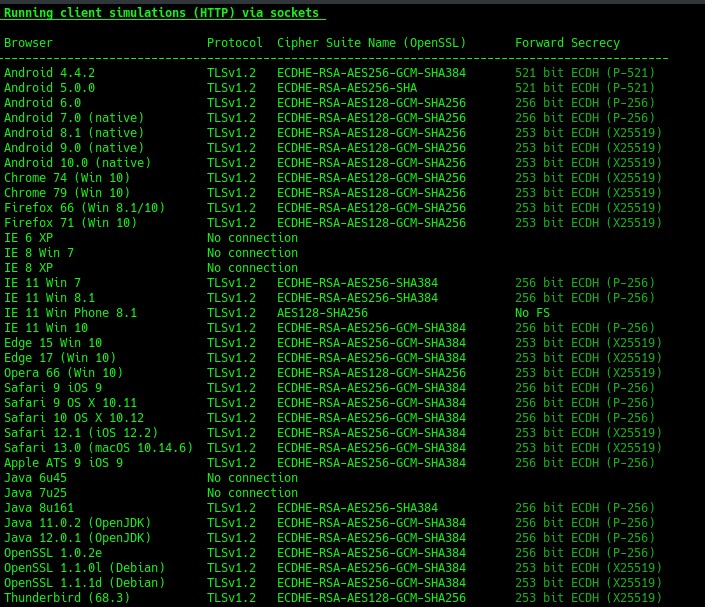
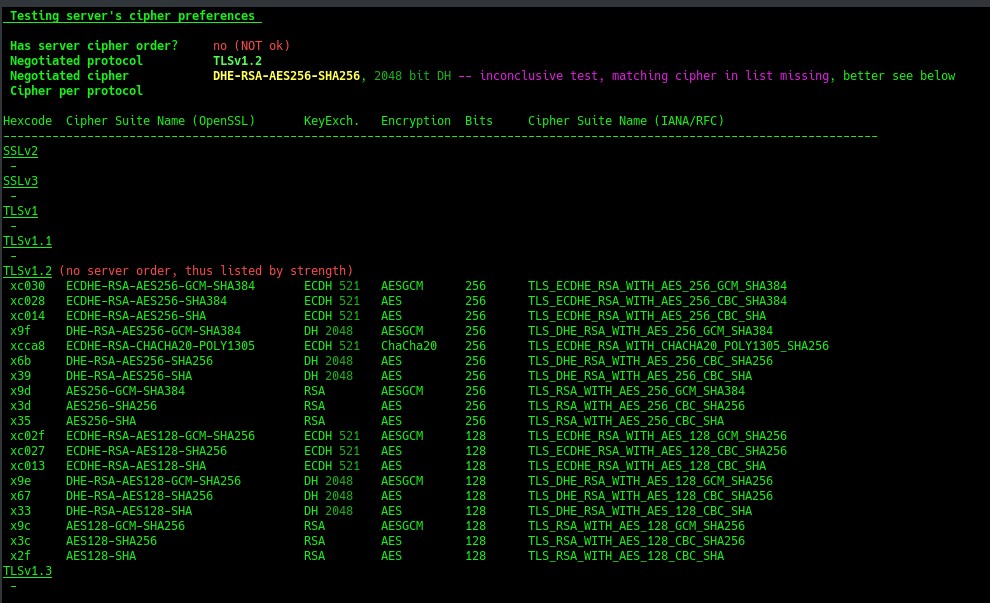
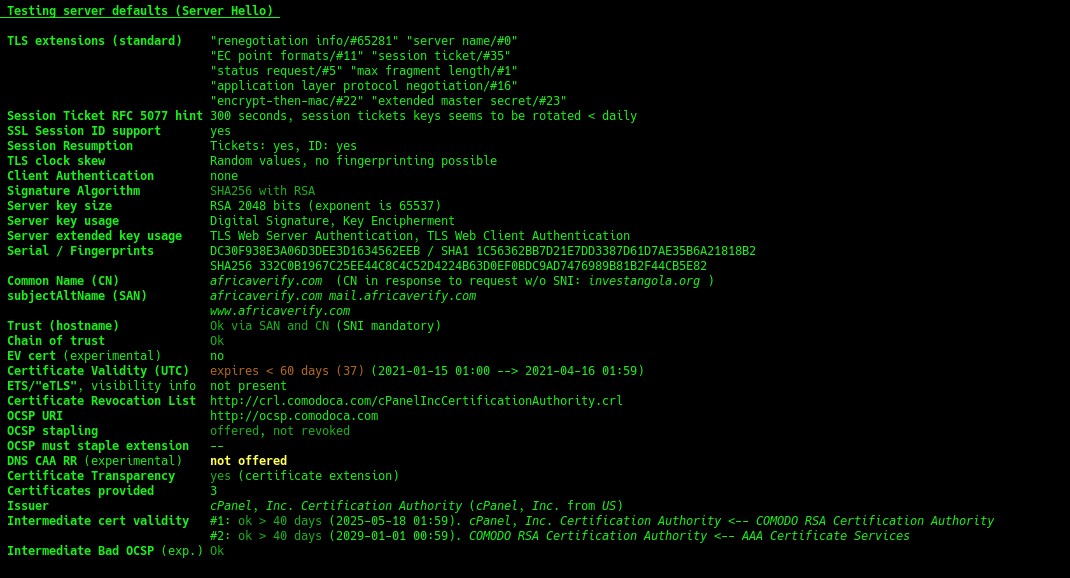
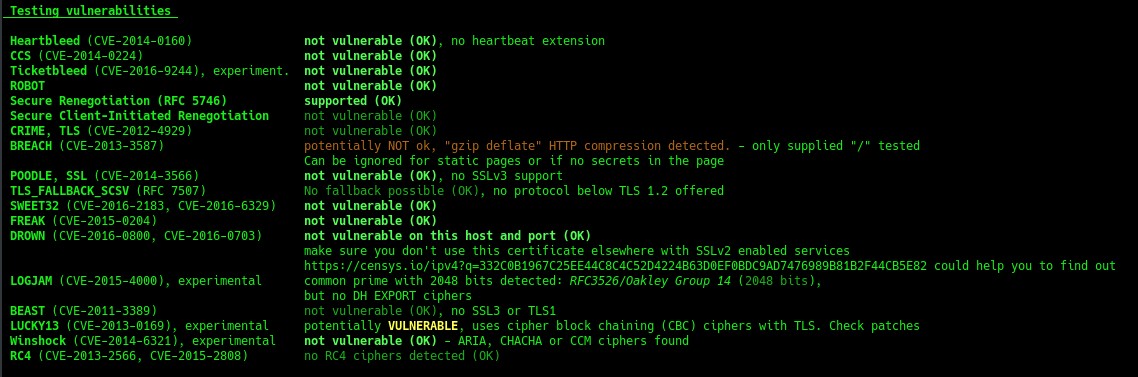
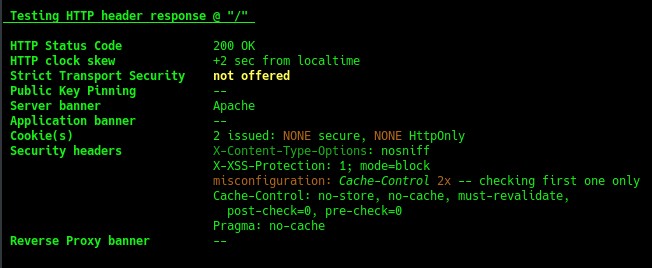
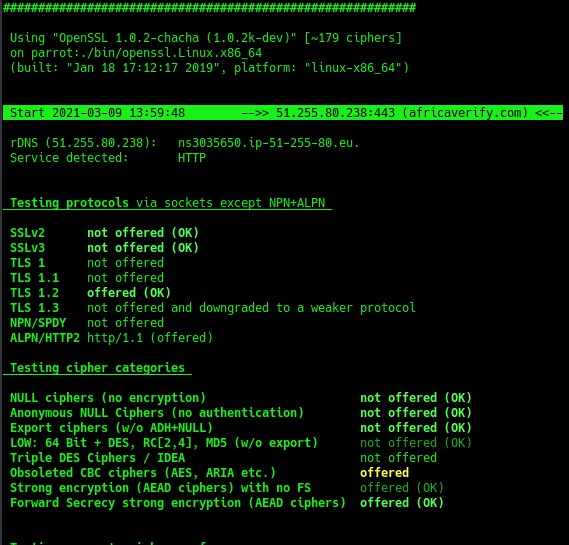


Scan testssl :

testssl.sh est un outil de ligne de commande gratuit qui vérifie le service d'un serveur sur n'importe quel port pour la prise en charge des chiffrements TLS / SSL, des protocoles ainsi que des failles cryptographiques et bien plus encore.

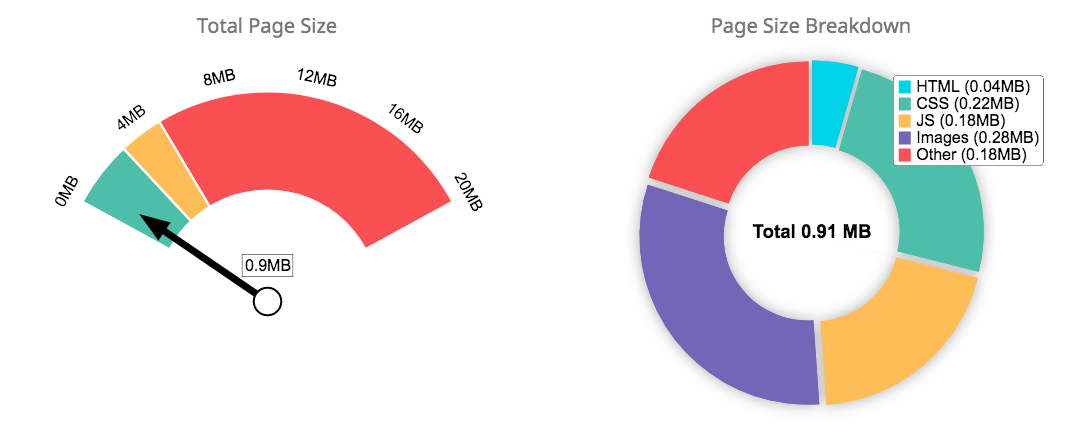
La sortie évalue les résultats par couleur (écran) ou par gravité (sortie de fichier) afin que vous puissiez savoir si quelque chose est bon ou mauvais. La sortie (écran) comporte plusieurs sections dans lesquelles des classes de vérifications sont effectuées. Pour faciliter la lisibilité à l'écran, il aligne et indente correctement la sortie.

Vous seul voyez le résultat. Vous pouvez également l'utiliser en interne sur votre réseau local. Sauf les recherches DNS ou à moins que vous ne demandiez à testssl.sh de vérifier la révocation des certificats, il n'utilise aucun autre hôte ni même des tiers pour aucun test.



La taille du site web

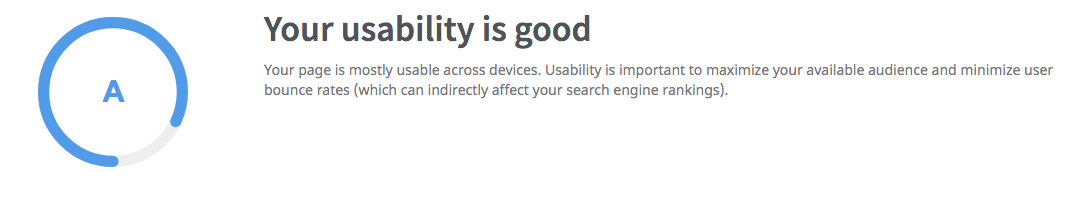
La taille du fichier de votre page est raisonnablement faible, ce qui est bon pour la vitesse de chargement de la page et l'expérience utilisateur.



Les mots-clés du site web

|  |  |
| --- | --- |
| Mots-clés existe déjà | Mots-clés à ajouter |
|  |  |

Expérience d’utilisateur (UX)



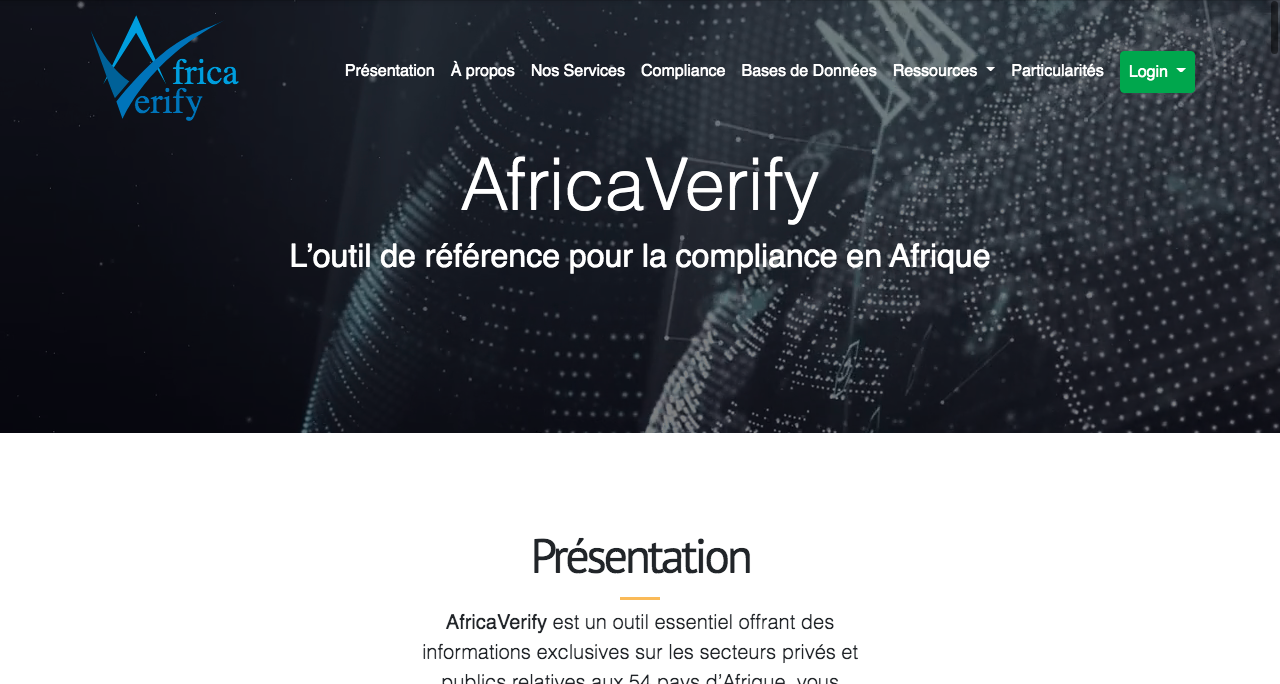
Votre page est principalement utilisable sur tous les appareils. La convivialité est importante pour maximiser votre audience disponible et minimiser les taux de rebond des utilisateurs.

Mobile



Ordinateur







Logiciels & Framework

