

Sécurité des systèmes d'information

Devoir 2 - Cross Site Request Forgery

Equipe : Chaimaa Zegoumou, Ayoub Nasr, Hamza Tamenaoul.

CVE : CVE-2019-1010054

Score CVSS : 6.8

1 Rapport

1 - Indiquer quel est le service ou le programme compromis, quel est le type de compromission

1 - *Service compromis* : Dolibarr en version 7.0.0.

Il s'agit d'une application Web qui permet d'offrir une solution de gestion d'une entreprise, association ou institution.. Elle tourne grâce aux technologies PHP, MySQL et serveur web HTTP.

2 - *Type de compromission* : Cross Site Request Forgery.

3 - *Explication de la compromission* :

Un **cross-site request forgery** est une injection de requête(s) illégitime(s) par rebond. Cela signifie alors qu'un attaquant peut effectuer des opérations sur un site sans le consentement d'un utilisateur.

En fait, ce type d'attaque se produit lorsqu'un serveur Web est conçu pour recevoir une requête d'un client sans aucun mécanisme permettant de vérifier qu'elle a été envoyée intentionnellement ; il peut alors être possible pour un attaquant de tromper un client afin qu'il fasse une requête involontaire au serveur Web qui sera traité comme une requête authentique. Cela peut être fait via une URL, une charge d'image, XMLHttpRequest, etc. et peut entraîner l'exposition de données ou l'exécution de code involontaire.

2 - Expliquer la vulnérabilité, décrire le mécanisme permettant de l'exploiter.

La vulnérabilité en question, décrite dans ce lien https://github.com/lucasgcilento/CVE/blob/master/Dolibarr_CSRF, existe dans les deux fichiers 'user/card.php' et 'admin/security.php', et est plus en détail exemplifié parfaitement dans 3 cas de figure :

1. On peut changer le mot de passe de n'importe quel utilisateur avec un mot de passe aléatoire en insérant un id via http://192.168.33.10/dolibarr/user/card.php?id=&action=confirm_password&confirm=yes.
2. On peut supprimer n'importe quel utilisateur en insérant un id via http://192.168.33.10/dolibarr/user/card.php?id=&action=confirm_delete&confirm=yes.
3. On peut désactiver l'encryption des mots de passe dans la base des données correspondante à l'appli Dolibarr via http://192.168.33.10/dolibarr/admin/security.php?action=disable_encrypt.

En effet, en inspectant le code source dans 'card.php' et 'security.php', on trouve que la raison principale pour laquelle ce logiciel est vulnérable, est qu'on ne s'est tout simplement pas servi des mécanismes de protection contre le CSRF, on détaillera ces derniers dans les questions qui suivent.

3 - Cette faille concerne-t-elle des machines clientes ou des machines serveurs ?

Cette faille est effectuée en remote : en effet, l'attaquant transmet d'une façon ou d'une autre (courrier contenant les URLs indiqués ci-haut ou les inclut dans un site..) à un administrateur connecté. Ainsi, la requête malveillante sera exécutée sur la machine de l'administrateur et affectera les serveurs hébergeant Dolibarr et ceux de la base des données.

4 - Décrire une architecture typique du système d'information qui pourrait être impliquée dans l'exploitation de ces failles. Cela peut prendre la forme d'un schéma où sont décrits :

- les services mis en oeuvre,
- les machines concernées (clients et serveurs),
- les équipements réseaux,
- les réseaux d'interconnexion.

Dans les deux figures ci-dessous, on présente deux cas typiques impliqués dans l'exploitation de la faille en question. Dans la figure 1, on présente le cas d'un administrateur (victime) qui reçoit du courrier malveillant provenant d'un attaquant, et qui l'ouvre, ce qui ne conduit à l'exploit voulu par ce dernier que s'il est bien **connecté** à Dolibarr. Dans la figure 2, on décrit un cas de figure où l'administrateur, connecté à Dolibarr, visite un site malveillant qui contient les URLs vulnérables et clique dessus, par conséquent, l'exploitation de la faille est effectuée.

Services mis en oeuvre : Serveur Apache hébergeant Dolibarr, serveur de bases de données.

Machines concernées : les serveurs Apache et des bases de données.

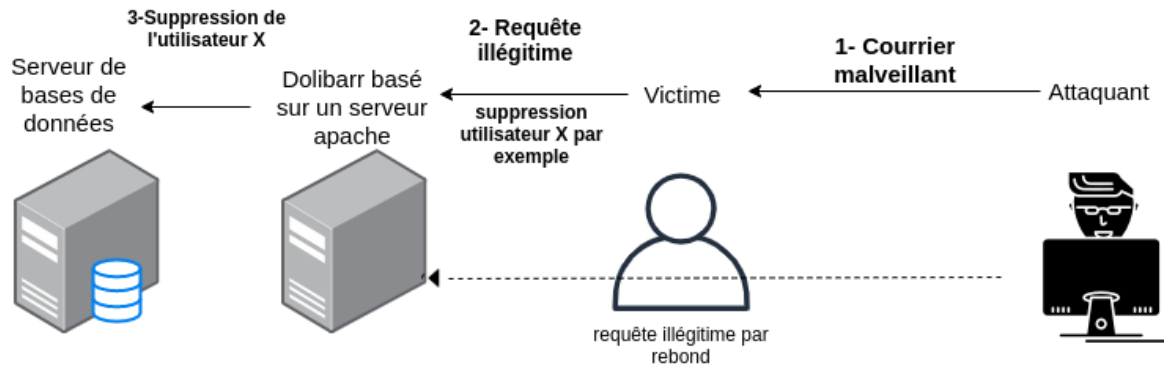


FIGURE 1 – Attaquant envoyant un courriel malveillant contenant une requête illégime.

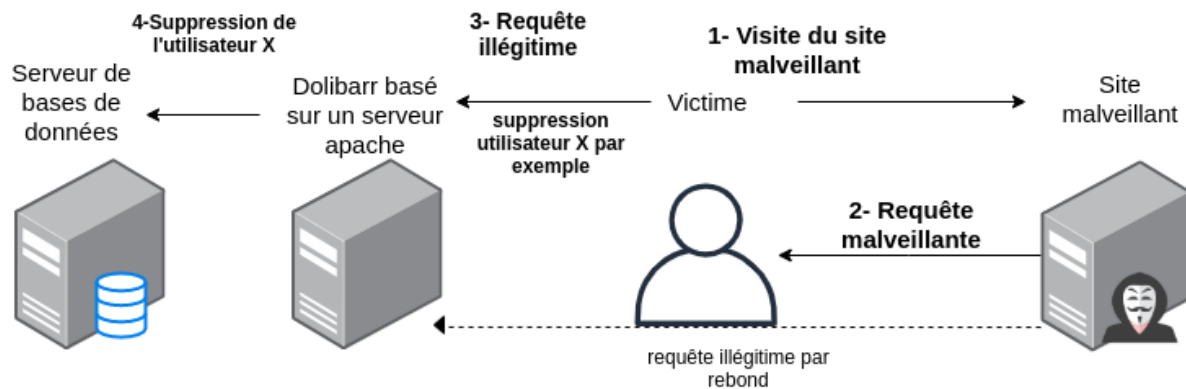


FIGURE 2 – Visite d'un site malveillant provoquant une requête illégime.

Équipements réseaux : Ports HTTP ouverts du serveur Apache et du client administrateur (victime), Serveur Apache connecté au serveur de bases de données, client admin **connecté** au serveur Apache.

Réseaux d'interconnexion : Réseau de communication entre client et serveur Apache (Wifi/Ethernet) et entre serveur Apache et serveur de bases de données.

5 - Référencer parmi les bonnes pratiques, celles qu'il faudrait utiliser pour limiter cette menace.

D'abord, il faut mentionner quelques mesures qui sont, contrairement aux idées reçues, complètement inutiles quant à la protection contre cette faille : vérification par cookie secret, utilisation de POST au lieu de GET, utilisation de l'URL rewriting/réécriture d'URL.

Vu que cette faille est une faille de type *CSRF*, il existe un certain nombre de bonnes pratiques qu'il est très conseillé d'implémenter afin de se prémunir contre cette faille :

- **Implémentation de l'authentification par jeton :** Le jeton est une chaîne de caractère unique par session utilisateur et générée aléatoirement de préférence par un CSPRNG (cryptographically secure pseudorandom number generator). Sa

propriété la plus importante est d'être imprévisible, sinon, cette méthode ne sera d'aucun effet. A chaque fois qu'un utilisateur se connecte à notre application, on lui attribue un nouveau jeton. Ce dernier doit être présent dans toutes les requêtes HTTP de l'utilisateur, ce qui nous permettra de vérifier avant de traiter toute nouvelle requête que cette dernière a été demandée par cet utilisateur et non suite à une requête lancée comme dans notre cas par un site ou une personne tiers. Il est possible de rajouter ce jeton dans un champ caché ou dans une en-tête, mais il faut faire attention au fait que le jeton ne soit divulgué dans les logs du serveur ou dans l'URL.

- **Demande de confirmation :** Bien que cette méthode nuit à l'UX (user experience), une demande de confirmation reste l'une des meilleurs méthodes afin de se prémunir contre des failles CSRF, et cette affirmation est vraie même dans notre cas. En effet, une demande de confirmation ne permet non seulement de prévoir simplement le cas où l'utilisateur a demandé de réaliser une action juste par erreur, mais elle permet également de prévoir une redirection malicieuse. Vu qu'à la base cette dernière demande une action supplémentaire de la part de l'utilisateur, elle bloque toute tentative d'exécution de requêtes à l'insu de l'utilisateur, et permet de se rendre compte qu'une tentative d'exploitation vient d'avoir lieu.

6 - S'il s'agit d'une faille issue d'un développement, indiquer ce qu'il aurait fallu mettre en place dans les équipes de développement pour limiter l'apparition d'une telle faille.

La faille *CSRF* est une faille peut être complètement prévenue au niveau de la phase de développement, et ce grâce aux bonnes pratiques citées plus haut. En effet, en implémentant un système de confirmation plus poussé, et non comme l'actuel, basé sur une simple variable envoyée par *GET*, et en utilisant aussi, les jetons dans toutes les opérations de l'utilisateur, et principalement dans les opérations critiques, on pourra facilement bloquer l'exploitation de ces failles *CSRF* dans cette application.

7 - De manière à mettre à jour la politique de sécurité des systèmes d'information, imaginer un contexte au sein d'une entreprise où cette faille pourrait être présente et rédiger un extrait de la PSSI (criticité de la faille, action préventive et curative, formation des utilisateurs à envisager, ...)

Description de l'entreprise : Une petite entreprise se sert de l'application Dolibarr afin de digitaliser sa gestion. Elle y stocke notamment tous ses rapports de comptabilité, ses factures et ses crédits. Elle se sert également de cette application pour gérer sa relation client, en affectant à chaque client un compte sur cette application.

Niveau de sensibilité : Très sensible vu qu'un attaquant peut causer la suppression des données décrites plus haut, l'altération d'informations de clients ou même la suppression de leurs comptes, provoquant ainsi la suppression de toutes les données qui leur sont associées.

Topologie des données : Les données sont stockées en intégralité sur le serveur de bases de données connecté au serveur Apache sur lequel l'application de l'entreprise est hébergée.

Niveau de criticité : Perte de données dommageables et affaiblissement de la relation client.

Actions à prendre :

- Si possible, éteindre le serveur hébergeant l'application Dolibarr en attendant un nouveau release contenant une version où cette faille sera corrigée.
- Impliquer les employés, les clients, les fournisseurs et toute entité possédant un compte sur cet espace et les sensibiliser quant à l'existence d'une telle faille dans le but d'éviter qu'un utilisateur ne clique sur un courrier malveillant quand il est connecté sur cet espace. Privilégier même que la connexion à cette espace soit l'unique tâche ouverte sur le navigateur d'un utilisateur, qu'il ne clique sur aucun lien lors de sa session.
- Le DSI de l'entreprise ne doit hésiter de s'impliquer dans la prévention contre cette faille, en contribuant au projet open source Dolibarr et en implémentant les préconisations mentionnées plus haut dans ce rapport.

2 Mise en évidence de la vulnérabilité

Afin de mettre en évidence l'exploitation de cette faille, on a mis en place une machine virtuelle dont la création est automatisée par le biais de Vagrant. On utilise comme image source celle d'Ubuntu, on y autorise l'accès en réseau privé par le biais de l'adresse 192.168.33.10 et on partage les fichiers du dossier *dev* avec cette VM. On installe également sur cette VM tous les logiciels nécessaires à cette manipulation et on démarre à la fin le serveur Apache. On vous dicte les étapes suivantes pour reproduire facilement l'exploitation de cette faille.

1. Créer VM

```
vagrant up
```

En local, exécuter les commandes suivantes dans le dossier 'dev' :

```
rm index.html
mv backup_index.html index.html
```

2. Se connecter à la VM créée en SSH

```
vagrant ssh
```

3. Visiter l'URL : <http://192.168.33.10/dolibarr> pour configurer Dolibarr :

- (a) Dans la section 'Dolibarr Database' : changer le login et le password de 'admin' et 'admin', et dans la section 'Database server - Superuser access' : changer le login à 'root' sans rajouter de password.

Dolibarr install or upgrade - Configuration file

You use the Dolibarr setup wizard from a Linux package (Ubuntu, Debian, Fedora...), so values proposed here are already optimized. Only the password of the database owner to create must be completed. Change other parameters only if you know what you do.

Web server

Directory where web pages are stored	<input type="text" value="/usr/share/dolibarr/html"/>	Without the slash "/" at the end Examples: • /var/www/dolibarr/html • C:\wwwroot\dolibarr\html
Directory to store uploaded and generated documents	<input type="text" value="/var/lib/dolibarr/documents"/>	Without the slash "/" at the end It is recommended to use a directory outside of your directory of your web pages. Examples: • /var/lib/dolibarr/documents • C:\My Documents\dolibarr\
URL Root	<input type="text" value="http://192.168.33.10/dolibarr"/>	Examples: • http://localhost/ • http://www.mysite.com:8180/dolibarr

Dolibarr Database

Database name	<input type="text" value="dolibardebien"/>	Database name
Driver type	<input type="text" value="mysqli (MySQL >= 4.1.0)"/>	Database type
Server	<input type="text" value="localhost"/>	Name or ip address for database server, usually 'localhost' when database server is hosted on same server than web server
Port	<input type="text" value="3306"/>	Database server port. Keep empty if unknown.
Database prefix table	<input type="text" value="llx_"/>	Database prefix table
Create database	<input checked="" type="checkbox"/>	Check box if database does not exist and must be created. In this case, you must fill the login/password for superuser account at the bottom of this page.
Login	<input type="text" value="admin"/>	Login for Dolibarr database owner.
Password	<input type="password" value="admin"/>	Password for Dolibarr database owner.
Create owner	<input checked="" type="checkbox"/>	Check box if database owner does not exist and must be created. In this case, you must choose its login and password and also fill the login/password for the superuser account at the bottom of this page. If this box is unchecked, owner database and its passwords must exist.

Database server - Superuser access

Login	<input type="text" value="root"/>	Login of the user allowed to create new databases or new users, mandatory if your database or its owner does not already exists.
Password	<input type="password"/>	Leave empty if user has no password (avoid this)

FIGURE 3 – 4- a) 1ère configuration

(b) Configurer le Dolibarr admin login à 'admin' et 'admin' comme password.

(c) se connecter avec les identifiants :

```
login:admin , password: admin
```

(d) Créer un utilisateur en cliquant sur 'Users & Groups' puis sur 'new User' dans la barre gauche de la page.

FIGURE 4 – 4- b) 2ème configuration

FIGURE 5 – 4- d) Création d'utilisateur

RESTER TOUJOURS CONNECTÉ en tant qu'admin pour voir l'effet de la faille CSRF.

4. Dans un autre onglet, visiter l'URL <http://192.168.33.10/> pour choisir l'attaque qu'on souhaite exécuter.
 - (a) Changer le mot de passe de l'utilisateur : Pour voir l'effet de cette faille plus en détail, se connecter à MySQL dans la session SSH lancée par 'vagrant ssh' en lançant :

```
sudo mysql
use dolibarrdebian ;
select rowid, login, pass, pass_crypted from llx_user;
```

En entrant l'id 2 dans ce champ, on remarque que la valeur de password du user 'test' a changé à une valeur complètement aléatoire. **Ne pas entrer la valeur**

1 juste pour assurer qu'on garde la session admin cohérente pour le reste de l'expérimentation.

- (b) Supprimer un utilisateur : On entre l'id 2, et on remarque en vérifiant sur mysql que l'utilisateur 'test' a été supprimé de la base de données.
- (c) Désactiver le cryptage de données : on est redirigé vers une page où l'on trouve que les permissions par défaut quant au cryptage des données ont été modifiées.

3 Glossaire

- **CSRF** : le cross-site request forgery (parfois prononcé sea-surf en anglais) ou XSRF, est un type de vulnérabilité des services d'authentification web.
- **Apache** : logiciel de serveur web gratuit et open-source qui alimente environ 46% des sites web à travers le monde.
- **logs** : fichiers où l'on stocke un historique horodaté et ordonné en fonction du temps des événements attachés à un logiciel
- **URL rewriting** : ou réécriture d'URL, est une méthode utilisée sur les sites web dynamique permettant de présenter des adresses plus propre et plus lisible.
- **CSPRNG** : ou générateur de nombres pseudo aléatoires cryptographiquement sécurisé est un générateur de nombres pseudo aléatoires doté de propriétés qui le rendent approprié pour une utilisation en cryptographie.

4 Références

- lien de la CVE : <https://www.cvedetails.com/cve/CVE-2019-1010054/>
- Repo Git détaillant la vulnérabilité : https://github.com/lucasgcilento/CVE/blob/master/Dolibarr_CSRF
- Explications du CSRF : <https://www.cert.ssi.gouv.fr/information/CERTA-2008-INF-003/>, [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF))
- Préventions contre le CSRF : https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html , <https://www.tothenew.com/blog/the-csrfcross-site-request-forgery-vulnerability/>