

Analyse de faille - CVE 2024-22416

Arvind Candassamy
Théo Rozier

Table des matières

Table des matières	1
Description de la faille	1
Exploitation de la faille	1
Architecture typique	2
Mesures de protection	3
Cible de sécurité	4
Démo	5
Glossaire	5
Sources/Références	6

Description de la faille

La faille affecte le logiciel pyLoad. pyLoad est un gestionnaire de téléchargement open-source permettant le téléchargement de fichiers depuis Internet. Avec son interface web user-friendly, il prend en charge le téléchargement via des protocoles tels que HTTP, HTTPS, FTP, etc. Le logiciel est écrit en Python et peut être exécuté sur différents OS (dont Linux, Windows). Il est souvent utilisé pour gérer les téléchargements de manière centralisée sur des serveurs dédiés (NAS [1]). En effet, les utilisateurs peuvent contrôler et surveiller leurs téléchargements à distance via l'interface web de pyLoad. Le logiciel comprend différents niveaux d'utilisateurs, gérés par une fonction d'authentification. C'est bien cette fonction qui est touchée par notre faille.

Cette faille est notée par sa page de rapport GitHub comme ayant une sévérité de 9.6, ce qui est très élevé. En effet, cette faille permet d'usurper l'administrateur seulement via le chargement d'une page malveillante, à condition que cette dernière connaisse en avance l'adresse de l'instance pyLoad de la victime. Ce qui n'est pas totalement improbable si la victime n'a pas changé le port et héberge localement son instance pyLoad.

Exploitation de la faille

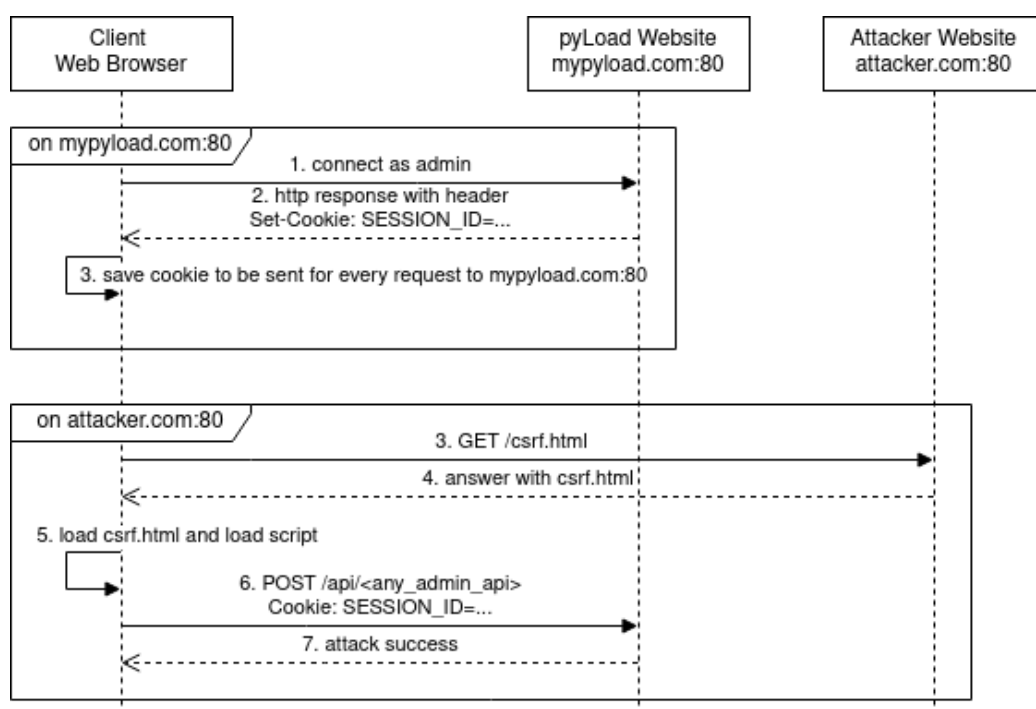
La faille permet les attaques de type CSRF [2] (Cross-Site Request Forgery), permettant à un attaquant non authentifié d'usurper l'identité d'un utilisateur connecté et de faire des actions qui n'étaient possibles que par l'utilisateur

connecté. La faille permet une attaque d'apparence simple et s'explique assez facilement, mais peut être dévastatrice pour un logiciel qui se fait exploiter.

De l'utilisateur en lecture simple à l'administrateur avec tous les droits, l'ampleur de l'attaque dépend du niveau de privilèges détenu par l'utilisateur usurpé. Dans le cas où l'utilisateur touché est un administrateur, les conséquences peuvent être graves :

1. Création d'administrateurs malveillants : L'attaquant peut utiliser la faille CSRF pour créer un nouvel utilisateur avec des privilèges administrateur. Cela donne à l'attaquant un contrôle complet sur la gestion des téléchargements, la configuration de l'application, et d'autres fonctionnalités sensibles ;
2. Suppression ou modification de téléchargements : Avec un accès administrateur, l'attaquant a accès à la gestion des téléchargements. Il pourrait supprimer des téléchargements en cours ou terminés, entraînant une perte de données pour l'utilisateur. Voir pire : il pourrait modifier un téléchargement, en injectant du code malveillant dans les fichiers téléchargés à l'insu de l'utilisateur initial. Permettant la propagation d'un malware à tous les utilisateurs du fichier téléchargé ;
3. Suppression et modification des fichiers de configuration des serveurs pouvant entraîner panne, malfonctionnement, etc. ;
4. Divulcation d'informations sensibles : Il pourrait faire fuiter la configuration des serveurs, le contenu des fichiers téléchargés, etc.

Architecture typique



Mesures de protection

L'administrateur réseau peut mettre en place différentes manœuvres pour limiter l'action de la faille.

1. Pour faire face aux altérations/suppressions douteuses de l'attaquant, on pourrait mettre en place des outils de surveillance du trafic réseau. Le déploiement d'outils comme les IDS (Systèmes de détection d'intrusion) et/ou les IPS (Système de prévention d'intrusion) permettent d'examiner le trafic réseau en temps réel. Via ces outils, l'administrateur système peut détecter les schémas d'activité suspects (ex : tentatives répétées d'accès à des téléchargements/ressources sensibles). Ainsi, on pourrait lever des alertes basées sur l'analyse du trafic en temps réel pour permettre une réponse rapide.
2. L'implémentation de couches de sécurité comme des pare-feu [3], ou des filtres de paquets réseaux [4] pourrait aussi permettre de filtrer le trafic réseau.

S'agissant d'une faille de développement, les développeurs auraient pu bloquer cette attaque en amont de différentes manières.

1. En configurant de manière correcte les cookies, ils auraient dû passer l'attribut SameSite : Strict [5]. De cette manière, le cookie d'authentification ne pourra être utilisé que sur l'interface pyLoad et non par site de phishing de l'attaquant.
2. Mettre en place le jeton anti-CSRF au lieu du cookie. Un jeton anti-CSRF est une valeur unique et aléatoire associée à une session utilisateur, il est inclus dans les requêtes API, et le serveur vérifie sa présence et son authenticité pour être sûr que l'on arrive bien de l'interface web pyLoad. Le jeton étant unique à chaque session et non enregistré sur le navigateur, on aurait pu permettre d'éviter les attaques CSRF.
3. Malgré toutes ces mesures, si on est encore victime d'attaques, on pourrait envisager de mettre en place une authentification deux facteurs (2FA) [6] pour les opérations administrateurs de Pyload.

Cible de sécurité

Dans cette partie, nous allons tenter de résumer tout ce qu'on a évoqué précédemment en quatre parties : Utilisateurs, biens à protéger, menaces et fonctions de sécurité.

Utilisateurs :

Administrateurs système : Ils font la configuration, la gestion et la surveillance des activités sur pyLoad.

Utilisateurs : Ils gèrent leurs téléchargements et accèdent aux fonctionnalités de l'application.

Biens à protéger :

Données utilisateurs : Les informations personnelles des utilisateurs, les paramètres de compte, historique de téléchargement, préférences de téléchargement, contenu des téléchargements etc.

Fonctionnalités administrateur : Les privilèges qui permettent aux administrateurs de configurer le système, d'ajouter des utilisateurs, de gérer les téléchargements, etc.

Menaces :

Attaques CSRF (Cross-Site Request Forgery) : Attaques qui exploitent les requêtes non authentifiées pour effectuer des actions non autorisées au nom d'un utilisateur authentifié.

Fuites d'Informations : Le risque de divulgation non autorisée d'informations sensibles : données utilisateur, configurations système etc.

Élévation de Privilèges : Les tentatives d'un utilisateur malveillant pour élever ses privilèges, en créant de nouveaux administrateurs ou en modifiant les configurations système par exemple.

Attaques sur les Téléchargements : Les menaces dans le contenu des téléchargements eux-mêmes, comme l'injection de malware dans les fichiers.

Fonctions de sécurité :

Authentification Robuste : Mise en place d'un mécanisme d'authentification complémentaire (2FA)

Jetons Anti-CSRF : Utilisation de jetons anti-CSRF pour prévenir les attaques CSRF et garantir que les requêtes proviennent bien de l'utilisateur depuis son interface web.

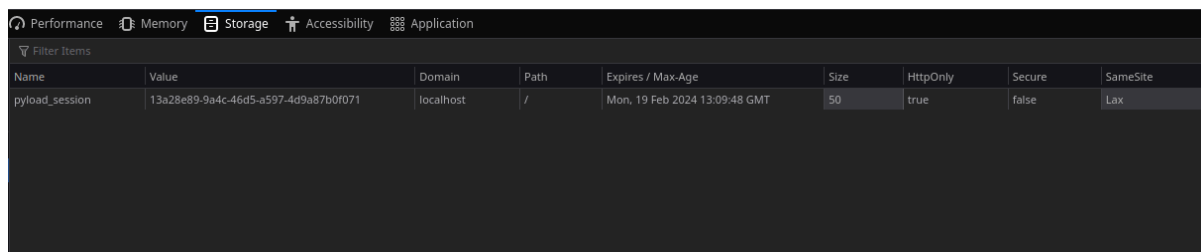
Surveillance et journalisation : Implémentation de mécanismes de surveillance en temps réel pour détecter et réagir rapidement aux comportements suspects.

Démo

Pour cette démo, nous avons mis en place un dépôt git qui contient une recette docker-compose qui met en place à la fois un serveur pyLoad et un serveur attaquant avec une seule page web "csrf.html" pour effectuer l'attaque.

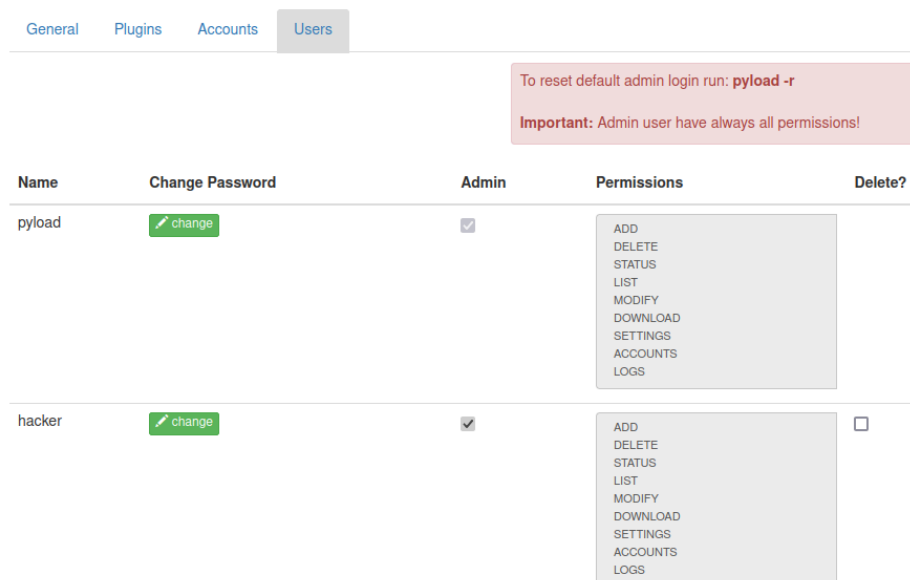
Pour faire le test vous-même, il faut déjà être sûr d'avoir installé docker-compose en plus de docker, et ensuite exécuter la commande "docker compose up" à la racine du dépôt.

Une fois que le docker compose est lancé, rendez-vous premièrement sur la page de connexion de pyLoad, à l'adresse "localhost:8000" et connectez vous avec l'utilisateur par défaut "pyload" et le mot de passe "pyload". L'utilisateur par défaut étant administrateur, cela va permettre l'exploit, on peut même observer le cookie qui a été ajouté dans les outils de développement de votre navigateur, on observe ainsi que l'attribut "SameSite" n'est pas défini à "Strict".



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
pyload_session	13a28e89-9a4c-46d5-a597-4d9a87b0f071	localhost	/	Mon, 19 Feb 2024 13:09:48 GMT	50	true	false	Lax

Ensuite, allez sur la page attaquant à l'adresse "localhost:8001/csrf.html", cela va instantanément produire une requête vers l'API du serveur pyLoad, l'exploit vient d'être fait avec succès (affichant "true"). On peut aller vérifier que cela à bien fonctionner en allant dans l'onglet Settings/Users de l'interface pyLoad, comme sur la figure suivante.



Name	Change Password	Admin	Permissions	Delete?
pyload	change	<input checked="" type="checkbox"/>	<div>ADD DELETE STATUS LIST MODIFY DOWNLOAD SETTINGS ACCOUNTS LOGS</div>	<input checked="" type="checkbox"/>
hacker	change	<input checked="" type="checkbox"/>	<div>ADD DELETE STATUS LIST MODIFY DOWNLOAD SETTINGS ACCOUNTS LOGS</div>	<input type="checkbox"/>

Glossaire

[1] NAS - "Network Attached Storage" (stockage en réseau attaché) ; Périphérique de stockage connecté au réseau, permettant le partage de fichiers entre les utilisateurs et offrant un moyen centralisé de stocker et d'accéder à des données.

[2] Attaques CSRF : Tentatives malveillantes d'un attaquant non authentifié de forcer un utilisateur à exécuter une action indésirable dans un contexte auquel il est authentifié. Cela se fait en exploitant la confiance que le site a envers l'utilisateur.

[3] Pare-feu : Un pare-feu est un dispositif de sécurité qui contrôle le flux de données entre un réseau privé et des réseaux externes, comme Internet. Il vise à bloquer ou autoriser le trafic en fonction de règles prédéfinies pour renforcer la sécurité du réseau.

[4] Filtre de paquets réseaux : Un filtre de paquets est une fonctionnalité d'un pare-feu qui examine et contrôle le trafic réseau en fonction de règles définies. Il filtre les paquets de données en fonction de critères tels que l'adresse IP, le protocole et le numéro de port.

[5] Attribut "Same-Site : strict" : L'attribut "Same-Site: strict" est une directive utilisée dans les cookies web. Il indique au navigateur de ne pas envoyer le cookie avec les requêtes de site tiers, renforçant ainsi la sécurité en réduisant le risque d'attaques basées sur les cookies.

[6] Authentification 2FA : L'authentification 2FA est un processus de sécurité qui demande deux méthodes distinctes pour vérifier l'identité d'un utilisateur. Cela peut inclure quelque chose que l'utilisateur sait (comme un mot de passe) et quelque chose qu'il possède (comme un code généré par une application sur son téléphone). Cela renforce la sécurité en ajoutant une couche supplémentaire de protection.

Sources/Références

<https://github.com/pyload/pyload/security/advisories/GHSA-pgpi-v85q-h5fm>

<https://developer.mozilla.org/fr/docs/Web/HTTP/Headers/Set-Cookie>

<https://github.com/mindstorm38/ensimag-secu3a-cve-2024-22416>