CVE-2023-4634

N’oubliez pas d’aller voir le README.md du gitlab, ainsi que la vidéo de tuto pour réaliser l’exploit. Bonne lecture ;).

Introduction :

La vulnérabilité CVE-2023-4634 concerne le plugin WordPress "Media Library Assistant" (MLA) dans les versions antérieures à 3.10. Cette vulnérabilité est due à un manque de contrôles appropriés sur les chemins de fichiers fournis à un paramètre appelé "mla_stream_file" dans le fichier "mla-stream-image.php". Cette vulnérabilité peut être exploitée à distance sans authentification, ce qui signifie qu'un attaquant peut potentiellement exploiter la faille sans avoir besoin de se connecter au site WordPress.

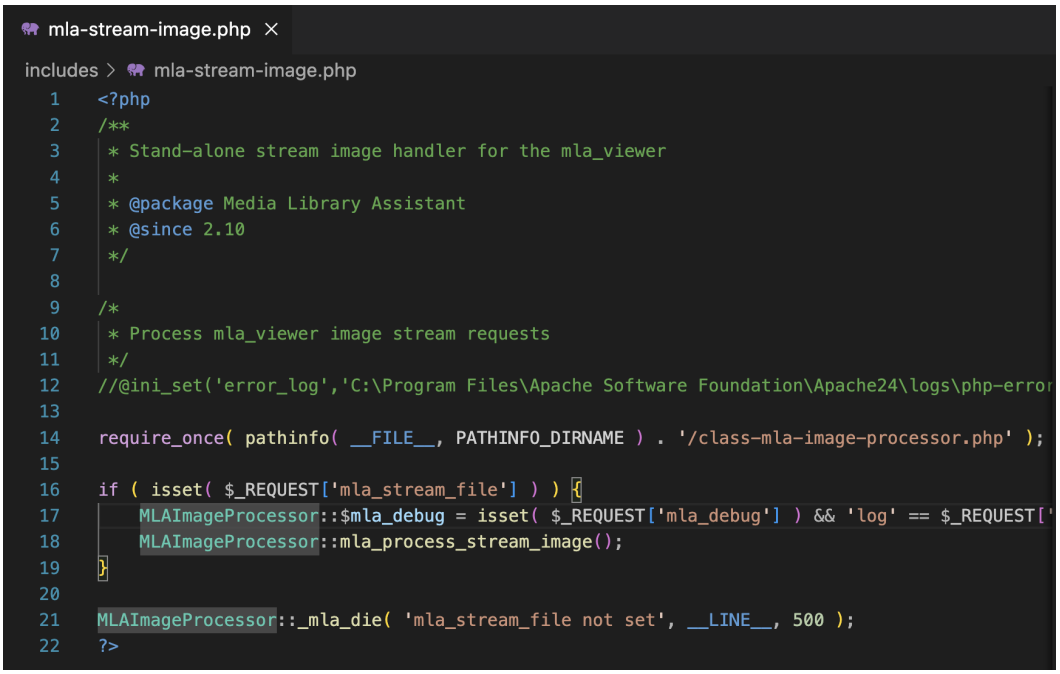
Explications de la vulnérabilité :

1. **Fonctionnalité du plugin MLA** : Le plugin "Media Library Assistant" est conçu pour aider à gérer et afficher des médias, tels que des images, dans un site WordPress. Il permet aux utilisateurs de manipuler et de générer des médias directement à partir de leur bibliothèque multimédia.

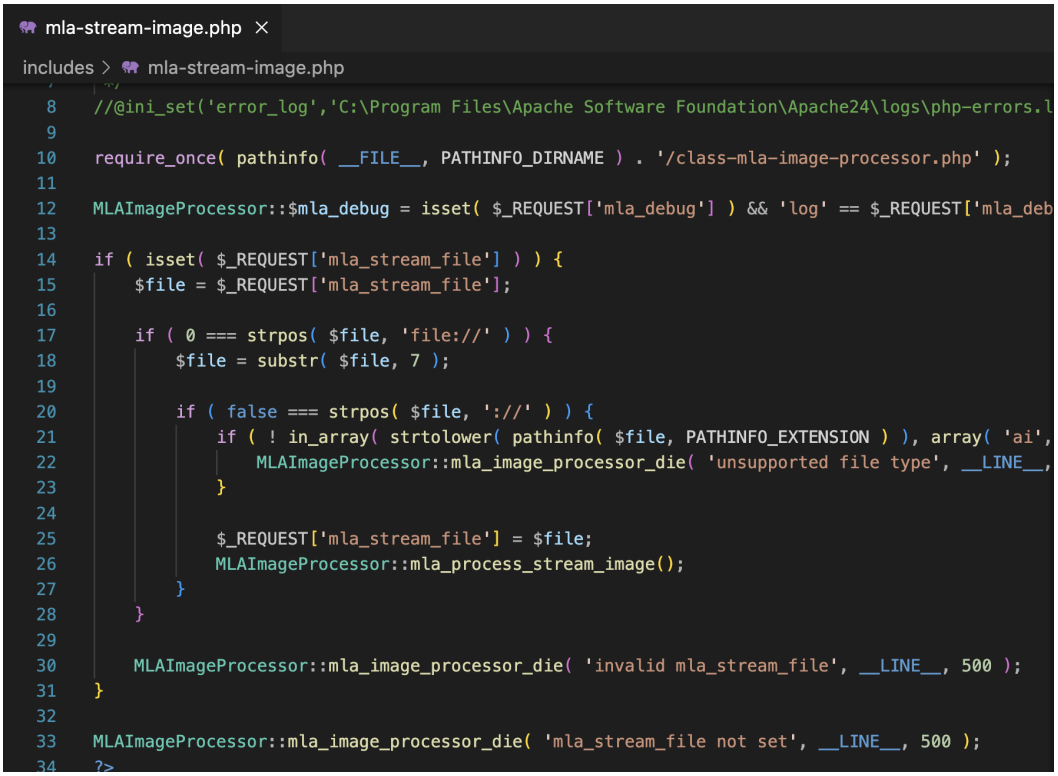
2. **Paramètre "mla_stream_file"** : Le fichier "mla-stream-image.php" est responsable du traitement des images via la bibliothèque Imagick, une bibliothèque de manipulation d'images. Le paramètre "mla_stream_file" est utilisé pour spécifier le chemi n du fichier à traiter.
3. **Absence de vérifications adéquates** : Le problème réside dans le fait que le plugin ne vérifie pas suffisamment si le chemin du fichier spécifié dans le paramètre "mla_stream_file" est valide et sécurisé. En conséquence, un attaquant peut fournir un chemin de fichier malveillant, y compris des fichiers distants accessibles via FTP.
   Dans la version corrigée, une nouvelle vérification est effectuée sur `mla_stream_file` pour détecter si elle commence par "file://". Si c'est le cas, le code supprime cette partie (les 7 premiers caractères) et vérifie s'il y a encore des "://" dans le chemin résultant. Si aucune autre occurrence de "://" n'est trouvée, le code vérifie si l'extension du fichier est dans une liste restreinte ("ai", "eps", "pdf", "ps"). Si ces conditions sont remplies, le traitement de `mla_stream_file` est effectué.
4. **Exploitation** : Un attaquant peut profiter de cette vulnérabilité en fournissant un chemin de fichier qui pointe vers un fichier malveillant stocké sur un serveur distant FTP qu'il contrôle. Cela permet à l'attaquant de charger des fichiers malveillants sur le serveur cible, de lire des fichiers sensibles ou même d'exécuter du code à distance.

En résumé, cette vulnérabilité de sécurité permet à un attaquant de manipuler le système de fichiers du serveur WordPress cible en exploitant le paramètre "mla_stream_file" du plugin "Media Library Assistant". Pour corriger cette faille, les développeurs du plugin ont publié une mise à jour vers la version 3.10 ou supérieure, qui corrige cette vulnérabilité en imposant des contrôles plus stricts sur les chemins de fichiers fournis à ce paramètre.

Avant le correctif :



Après le correctif :



Réponses aux questions :

Ce texte décrit une faille de sécurité dans le plugin WordPress "Media Library Assistant" dans sa version inférieure à 3.10. Voici les réponses à vos questions :

1. **Service ou programme compromis / Type de compromission :**
   - Service compromis : Le service ou le programme compromis est le plugin WordPress "Media Library Assistant" dans les versions antérieures à 3.10. Il s'agit d'un plugin utilisé pour gérer la bibliothèque multimédia dans les sites WordPress.
   - Type de compromission : La faille de sécurité permet une compromission de type RCE (Remote Code Execution) et LFI (Local File Inclusion). Cela signifie qu'un attaquant peut exécuter du code à distance sur le serveur cible ou inclure des fichiers locaux à partir du serveur, en fonction de la configuration d'Imagick sur ce serveur.
2. **Explication de la vulnérabilité et de son exploitation** :
   - La vulnérabilité découle d'une référence distante non authentifiée à la conversion Imagick() sur le serveur cible. Imagick est une bibliothèque utilisée pour manipuler des images. Si le serveur dispose d'Imagick configuré de manière vulnérable, un attaquant peut exploiter cette faille pour obtenir un accès non autorisé et exécuter du code à distance.

    - Pour l'exploitation de la vulnérabilité, l'attaquant doit créer un serveur FTP distant et héberger deux fichiers malveillants, "malicious.svg" et "malicious.svg[1]", sur ce serveur FTP. Ensuite, l'attaquant modifie le contenu de ces fichiers pour inclure un chemin vers un fichier sensible, par exemple, "wp-config.php". L'attaquant peut ensuite déclencher la conversion en utilisant une URL qui pointe vers le fichier vulnérable du plugin Media Library Assistant : `mla-stream-image.php`. Selon le mécanisme d'Imagick sur le serveur cible, cela peut entraîner l'inclusion de fichiers locaux ou l'exécution de code à distance.
3. **Faille concernant des machines clientes ou des machines serveurs** :
    - Cette faille concerne principalement les serveurs où le plugin WordPress "Media Library Assistant" est installé. L'attaque est dirigée vers le serveur cible qui exécute WordPress avec ce plugin vulnérable. Les machines clientes qui visitent un site WordPress utilisant cette version vulnérable du plugin pourraient être affectées indirectement si l'attaquant exploite la faille pour compromettre le serveur et y exécuter du code malveillant. Cependant, l'exploitation elle-même est effectuée sur le serveur, ce qui en fait principalement une faille affectant les machines serveurs.
4. **Architecture typique** :

Pour décrire une architecture typique du système d'information qui pourrait être impliquée dans l'exploitation de ces failles, nous allons utiliser une approche simplifiée sous forme de texte. Voici une représentation schématique des éléments impliqués :

Architecture Typique du Système d'Information :

1. **Réseau Internet** :
    - C'est l'environnement extérieur où les attaquants se trouvent.
2. **Pare-feu (Firewall)** :
    - Un pare-feu est placé entre Internet et le réseau local de l'organisation. Il surveille et contrôle le trafic entrant et sortant, agissant comme une barrière de sécurité.
3. **Réseau Local de l'Organisation**
4. **Serveurs Web** :
    - Les serveurs web hébergent les sites WordPress vulnérables. Dans ce cas, le serveur Web exécute WordPress et le plugin "Media Library Assistant."
5. **Serveur FTP** :
    - Un serveur FTP distant est utilisé pour héberger les fichiers malveillants, y compris les fichiers "malicious.svg" et "malicious.svg[1]" qui seront exploités.
6. **Serveur HTTP (Polyglot PNG/PHP)** :
    - Un serveur HTTP est utilisé pour héberger le fichier polyglot PNG/PHP malveillant, que l'attaquant souhaite exécuter sur le serveur cible.
7. **Serveur d'Application** :
    - Le serveur d'application WordPress interagit avec la base de données et exécute le code WordPress.
8. **Bases de Données** :
    - La base de données stocke les données de WordPress, y compris les configurations, les articles, les pages, etc.

9. **Imagick Server** :
    - Imagick est une bibliothèque de manipulation d'images. Dans ce contexte, il peut s'agir d'un composant logiciel qui traite les images envoyées au serveur, ce qui est essentiel pour exploiter la faille.
10. **Postes Clients** :

- Les postes clients sont des ordinateurs depuis lesquels les utilisateurs accèdent au site Web vulnérable. Ils sont principalement des clients naviguant sur Internet.

Flux de Données et Interactions :

- Les attaquants, situés sur Internet, ciblent le serveur Web WordPress vulnérable. Ils exploitent la faille en utilisant des fichiers malveillants hébergés sur un serveur FTP distant.
- Lorsque l'exploitation réussit, le code malveillant est exécuté sur le serveur cible. Le serveur d'application, le serveur de base de données et le serveur Imagick peuvent tous être impliqués dans le processus, en fonction de la nature de l'exploitation.

Veuillez noter que cette architecture est simplifiée pour illustrer les principaux acteurs impliqués. Dans un environnement réel, il peut y avoir plus de couches de sécurité, de serveurs et de composants. La sécurité du système d'information joue un rôle critique pour empêcher de telles exploitations.

5. **Limiter impact de l'exploitation de ces failles / Empêcher qu'elles ne puissent être exploitées** :

En tant qu'administrateur système sur un réseau contenant des machines pouvant être affectées par ces failles, vous pouvez préconiser ce qui suit dans le rapport :

Pour limiter l'impact de l'exploitation de ces failles :

- **Mises à jour et patchs** : Assurez-vous que tous les logiciels, y compris WordPress, les plugins et les bibliothèques tierces comme Imagick, sont régulièrement mis à jour avec les derniers correctifs de sécurité. Les mises à jour doivent être appliquées dès qu'elles sont disponibles.
- **Restrictions de Réseau** : Utilisez un pare-feu pour restreindre l'accès à des ports spécifiques, tels que le port FTP, depuis des sources externes non autorisées. Cette restriction peut empêcher l'attaquant d'accéder au serveur FTP distant.
- **Accès Minimisé** : Restreignez les privilèges d'accès sur le serveur. Assurez-vous que les utilisateurs ont uniquement les autorisations nécessaires pour effectuer leurs tâches. Par exemple, ne donnez pas d'accès FTP inutile.
- **Surveillance du Trafic** : Mettez en place des outils de surveillance du trafic réseau pour détecter toute activité suspecte, notamment des tentatives d'accès non autorisées à des services sensibles.
- **Système de Détection d'Intrusion (IDS)** : Utilisez un IDS pour surveiller les activités du serveur et alerter en cas de comportement anormal ou d'exploitations de failles.
- **Audits de Sécurité Réguliers** : Effectuez des audits de sécurité réguliers pour identifier et résoudre les vulnérabilités potentielles. Ces audits peuvent inclure des tests de pénétration pour vérifier la résistance aux attaques.

- **Sécurité des Services FTP** : Si l'utilisation de FTP est nécessaire, assurez-vous qu'il est configuré de manière sécurisée, avec des mots de passe forts et des contrôles d'accès appropriés.

Pour empêcher l'exploitation de ces failles :

- **Validation des Données** : Dans le développement, appliquez une validation stricte des données d'entrée pour éviter les vulnérabilités de type LFI et RCE. Ne jamais faire confiance aux données d'entrée non vérifiées.
- **Limitation des Bibliothèques Tierces** : Minimisez l'utilisation de bibliothèques tierces et de plugins. Utilisez uniquement ce qui est nécessaire, et assurez-vous que ces composants sont réputés et mis à jour régulièrement.
- **Formation des Développeurs** : Formez les équipes de développement à la sécurité des applications. Les développeurs doivent comprendre les vulnérabilités courantes et les bonnes pratiques de codage sécurisé.
- **Tests de Sécurité** : Intégrez des tests de sécurité, y compris des tests d'intrusion automatisés, dans le processus de développement pour identifier les vulnérabilités avant la mise en production.
- **Code Review** : Mettez en place des revues de code pour passer en revue le code source à la recherche de vulnérabilités potentielles. Un autre regard peut révéler des problèmes de sécurité.
- **Examen des Configurations Imagick** : Si Imagick est utilisé, assurez-vous que les configurations sont sécurisées. Restreignez les opérations possibles avec Imagick pour limiter les risques.
- **Utilisation d'Outils de Sécurité** : Intégrez des outils de sécurité, tels que des analyseurs de code statique, dans le processus de développement pour identifier les problèmes de sécurité dès le début.
- **Gestion des Vulnérabilités** : Mettez en place un processus de gestion des vulnérabilités pour suivre et résoudre rapidement les problèmes de sécurité identifiés.

6. **Référencer parmi les bonnes pratiques, celles qu'il faudrait utiliser pour limiter cette menace :**

Pour limiter cette menace, les bonnes pratiques suivantes devraient être utilisées :

- Mises à jour régulières : Maintenir tous les logiciels, systèmes d'exploitation et applications à jour avec les derniers correctifs de sécurité.
- Pare-feu et Filtrage des Ports : Utiliser un pare-feu pour restreindre le trafic non autorisé vers des ports sensibles.
- Privilèges Minimaux : Attribuer des privilèges et des accès aux utilisateurs et aux systèmes de manière minimale, en fonction de leurs besoins.
- Surveillance Active : Surveiller activement le trafic réseau et les journaux d'activité pour détecter les comportements suspects.
- Éducation en Sécurité : Sensibiliser les utilisateurs et le personnel à la sécurité informatique et aux bonnes pratiques.

7. **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 :**

Voici ce qui aurait dû être mis en place dans les équipes de développement pour limiter l'apparition d'une telle faille :

- **Validation et Filtrage des Entrées** : Les développeurs doivent valider et filtrer toutes les données d'entrée pour éviter les failles d'inclusion de fichiers (LFI) et d'exécution de code à distance (RCE).
- **Pratiques de Codage Sécurisé** : Les équipes de développement doivent être formées aux meilleures pratiques de codage sécurisé, y compris l'échappement des données, la validation des entrées, la gestion des sessions sécurisées, etc.
- **Révision de Code** : Mettre en place des revues de code pour examiner le code source à la recherche de vulnérabilités potentielles, y compris les vulnérabilités liées à la sécurité.
- **Tests de Sécurité Automatisés** : Intégrer des outils de test automatisés de sécurité dans le processus de développement pour identifier les vulnérabilités avant la mise en production.
- **Gestion des Vulnérabilités** : Établir un processus de gestion des vulnérabilités pour suivre, évaluer et corriger les vulnérabilités identifiées.
- **Contrôles d'Accès Appropriés** : Mettre en place des contrôles d'accès appropriés pour les services et les données sensibles, en limitant l'accès aux utilisateurs autorisés.
- **Gestion des Dépendances** : Surveiller et mettre à jour les dépendances logicielles, y compris les bibliothèques tierces, pour s'assurer qu'elles sont à jour et sécurisées.
- **Éducation Continue** : Assurer une éducation continue des développeurs en matière de sécurité informatique pour rester au fait des dernières menaces et des meilleures pratiques de sécurité.
- **Développement Axé sur la Sécurité** : Adopter une approche axée sur la sécurité dès le début du développement, en intégrant la sécurité dans le processus de développement plutôt qu'en l'ajoutant en aval.

8. **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, …) :**

Imaginons un contexte au sein d'une entreprise où la faille CVE-2023-4634 dans le plugin WordPress "Media Library Assistant" est présente. Cette entreprise exploite un site web critique qui repose sur WordPress pour la gestion de ses médias et de son contenu. La

PSSI (Politique de Sécurité des Systèmes d'Information) devrait être mise à jour pour aborder cette faille spécifique. Voici un extrait de la PSSI à envisager :

Criticité de la Faille :

La faille CVE-2023-4634 présente un risque de sécurité élevé pour notre entreprise. Elle permet à un attaquant distant non authentifié d'exploiter des vulnérabilités de type LFI (Inclusion de Fichier Local) et RCE (Exécution de Code à Distance) sur notre serveur WordPress. En cas d'exploitation réussie, l'attaquant peut compromettre la confidentialité, l'intégrité et la disponibilité des données et du système. Les conséquences pourraient inclure la divulgation de données sensibles, la perturbation des opérations commerciales et la compromission de la sécurité du site web.

Actions Préventives :

1. **Mise à jour du Plugin :** Assurez-vous que le plugin "Media Library Assistant" est mis à jour vers une version ne présentant pas la faille CVE-2023-4634. Si une mise à jour est indisponible, envisagez de remplacer le plugin par une alternative plus sécurisée.
2. **Surveillance en Temps Réel :** Mettez en place un système de surveillance en temps réel pour détecter toute tentative d'exploitation de la faille. Les alertes de sécurité devraient être générées en cas d'activité suspecte.
3. **Restrictions de Réseau :** Utilisez un pare-feu pour restreindre l'accès aux ports sensibles, en particulier le port FTP, depuis des sources non autorisées.
4. **Mise en place de Correctifs :** En cas de vulnérabilité détectée, mettez en place rapidement des correctifs et des correctifs de sécurité pour corriger la faille.
5. **Formation en Sécurité :** Fournissez une formation en sécurité informatique pour les administrateurs du site web et le personnel chargé de la gestion des plugins. Cette formation doit comprendre des pratiques de sécurité pour éviter l'exploitation de failles similaires à l'avenir.

Actions Curatives :

1. **Isolation du Serveur :** En cas d'exploitation réussie de la faille, isolez immédiatement le serveur compromis du réseau pour éviter toute propagation ultérieure de l'attaque.
2. **Analyse de l'Impact :** Réalisez une analyse complète de l'impact de l'exploitation de la faille, y compris la vérification de la compromission des données et des systèmes.
3. **Nettoyage et Restauration :** Nettoyez le serveur compromis et effectuez une restauration à partir de sauvegardes saines. Assurez-vous que le serveur est exempt de malware ou de code malveillant.
4. **Évaluation de la vulnérabilité :** Réalisez une évaluation détaillée de la vulnérabilité pour comprendre comment elle a été exploitée et comment elle peut être évitée à l'avenir.
5. **Communication :** Communiquez l'incident à toutes les parties prenantes concernées, y compris les autorités compétentes et les utilisateurs du site web, en fournissant des informations sur l'incident et les mesures prises pour le résoudre.
6. **Révision de la PSSI :** Après l'incident, révisez la PSSI pour inclure des mesures spécifiques pour prévenir de telles failles à l'avenir, notamment des procédures de

test de sécurité, des politiques de mise à jour et de gestion des plugins, et des formations de sensibilisation à la sécurité pour le personnel.

7. **Suivi et Rapport :** Mettez en place un processus de suivi continu de la sécurité et de rapports sur les incidents pour assurer une réponse rapide aux menaces futures.

Formation des Utilisateurs :

Les utilisateurs du site web, en particulier le personnel administratif et les gestionnaires de contenu, devraient être formés aux pratiques de sécurité de base. Ils doivent comprendre les risques associés à l'exploitation de failles de sécurité et être conscients des mesures préventives à prendre, telles que l'importance de maintenir les plugins et les logiciels à jour.

La sécurité des systèmes d'information doit être une priorité continue, et les actions préventives et curatives doivent être mises en œuvre de manière proactive pour atténuer les risques associés à cette faille spécifique.

Explication vulnérabilité :

Étape 1 : Génération des fichiers SVG

- Dans cette première étape, le script génère des fichiers SVG/MSL (Multi-Size Layer) polyglots. L'objectif est de créer des fichiers spécialement conçus pour exploiter une vulnérabilité dans le plugin WordPress Media-Library-Assistant.
- L'utilisateur fournit des paramètres tels que `svg_polyglot_name`, `svg_exploiter_names`, `remotehttp`, `png_polyglot_name`, `webserverpath`, et `exploitname`.
- Le script crée un fichier SVG/MSL principal (poly_svg) avec les paramètres fournis par l'utilisateur. Ce fichier contient des opérations d'ImageMagick destinées à être exécutées ultérieurement.
- Le script crée un répertoire nommé remote_ftp s'il n'existe pas déjà. Il enregistre le fichier SVG/MSL principal (poly_svg) avec le nom `svg_polyglot_name` dans ce répertoire. Une version avec le nom `svg_polyglot_name[0]` est également créée.
- Le script génère des fichiers VID bruteforcers en remplaçant le marqueur "FUZZ" dans le nom du fichier `svg_exploiter_names` par une lettre brute (par exemple, "exploiter_A.svg", "exploiter_B.svg"). Ces fichiers contiennent des opérations d'ImageMagick similaires destinées à être exécutées ultérieurement.
- Les fichiers VID bruteforcers sont enregistrés dans le même répertoire `remote_ftp` avec le nom approprié, et une version avec "[0]" à la fin de leur nom est également créée.

Étape 2 : Création des fichiers PNG/PHP

- Dans cette deuxième étape, le script génère des fichiers PNG/PHP polyglots malveillants. Ces fichiers contiennent du code PHP malveillant qui sera exécuté sur le serveur cible.
- Pour créer ces fichiers, l'utilisateur utilise l'option `--generatepng` et fournit un payload PHP en utilisant l'argument `payload`.
- Le code PHP est intégré dans des métadonnées d'une image PNG standard nommée "sample.png", qui doit être préalablement placée dans le répertoire "exploit-png".
- Ces fichiers PNG/PHP sont conçus pour être référencés à partir des fichiers SVG/MSL créés dans l'étape précédente.

Lien entre les deux étapes :

- Le lien essentiel entre ces deux étapes réside dans le fait que le fichier PNG/PHP malveillant généré est incorporé dans le code des fichiers SVG/MSL polyglots. Lorsque ces fichiers SVG/MSL sont utilisés pour exploiter la vulnérabilité dans le plugin WordPress, ils référencent et déclenchent l'exécution du code PHP contenu dans le fichier PNG/PHP.
- En résumé, les fichiers SVG/MSL servent de vecteur d'exploitation en incorporant le fichier PNG/PHP qui exécute le code malveillant sur le serveur cible. C'est ce lien qui permet à l'attaquant de prendre le contrôle du serveur vulnérable.

Etape 3 : Exploitation

- Le script commence par vérifier si tous les arguments nécessaires pour exploiter la cible ont été correctement configurés par l'utilisateur.
- Une première vérification est effectuée pour s'assurer que l'option `--svg_exploiter_names` contient le marqueur "FUZZ", qui sera utilisé dans l'exploitation.
- Ensuite, le script effectue des vérifications sur la cible, notamment pour déterminer si le plugin WordPress Media-Library-Assistant est installé. Il vérifie également si la version du plugin est vulnérable (inférieure à 3.10).
- Il procède ensuite à des vérifications des fichiers hébergés sur un serveur FTP distant. Il vérifie si le fichier SVG/MSL principal (`ftp_target`) et sa version avec [0] (`ftp_target_frame`) sont accessibles.
- De plus, le script vérifie la disponibilité d'un échantillon de fichier exploiteur VID sur le serveur FTP distant (`ftp_target_exploiter` et `ftp_taget_exploiter_frame`).
- Enfin, le script vérifie la disponibilité du fichier PNG/PHP malveillant sur un serveur HTTP distant (`remote_virus`).
- Après avoir effectué toutes ces vérifications, le script lance l'exploitation. Il envoie des requêtes vers le plugin WordPress Media-Library-Assistant en utilisant le fichier SVG/MSL principal avec le paramètre `mla_stream_file`. Le nombre de requêtes simultanées est contrôlé par la variable `concurrency`.
- Après avoir envoyé les requêtes, le script attend 5 secondes pour permettre au fichier d'être créé sur le serveur cible.

- Ensuite, le script commence à envoyer des requêtes d'exploitation en utilisant différents fichiers VID bruteforcers en remplaçant "FUZZ" par des lettres brutes.
- Après l'envoi de ces requêtes d'exploitation, le script attend encore 5 secondes.
- Il vérifie si le fichier PHP malveillant (`pwned.php`) a été déposé sur le serveur cible. Si c'est le cas, l'exploitation a réussi, et le script se termine avec un message de réussite.
- Si l'exploitation ne réussit pas, le script continue à vérifier périodiquement si le fichier est déposé sur le serveur. S'il échoue neuf fois consécutivement, le script se termine avec un message d'échec.
- Si des erreurs surviennent pendant l'exploitation, le script génère des messages d'erreur appropriés.
- Si des arguments essentiels sont manquants, le script génère un message d'erreur et se termine.