1 A2 Cryptographic Failures

1.1 Criptografía Básica

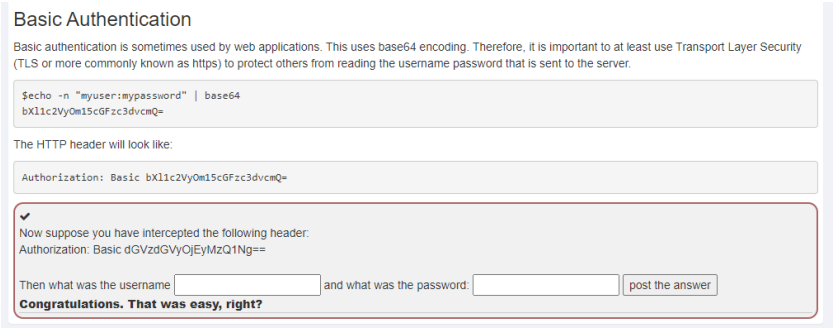
Revisa los conceptos relacionados con

- Encoding
- Hashing
- Encryption
- Signing
- Keystores
- Security defaults
- Post quantum crypto

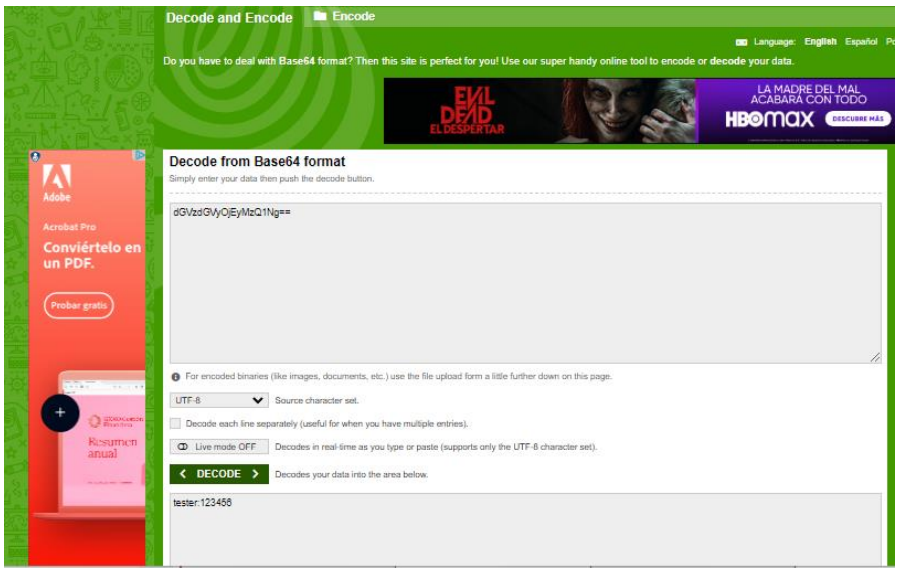
1.2 Base64 Encoding

1.2.1 Basic Authentication

El ejercicio consiste en decodificar un HTTP header codificado en Base64.

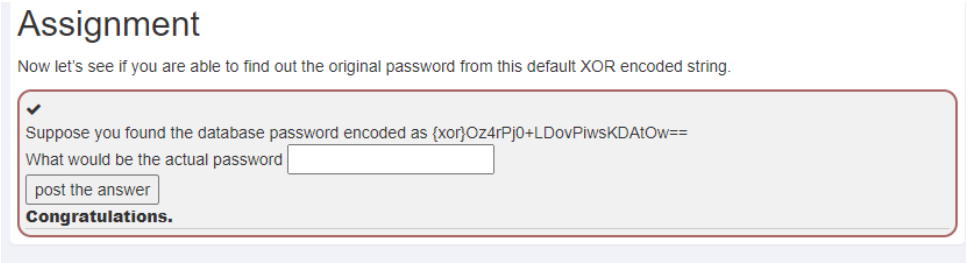


Decodifica el código con Base 64 Decode & Encode: https://www.base64decode.org/



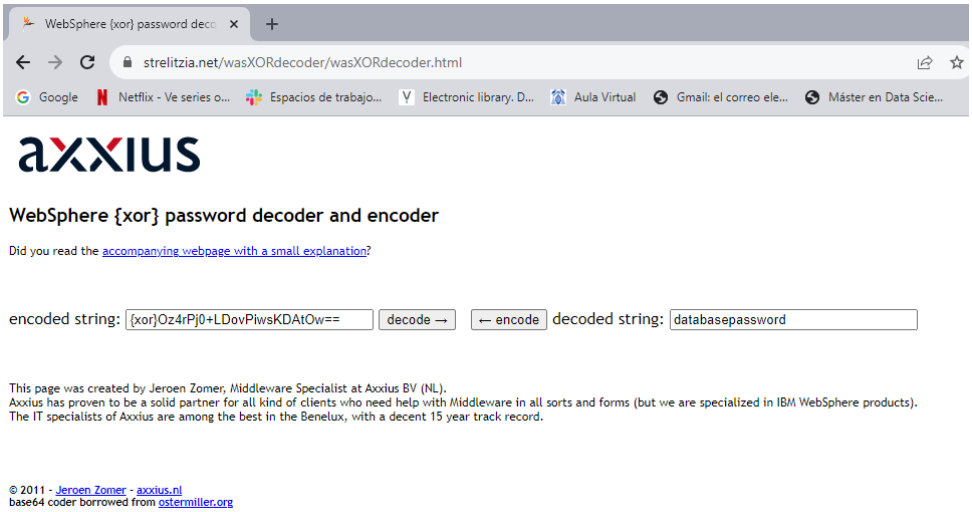
1.2.2 Other Encoding

En este punto del ejercicio tiene que decodificar el password codificado con {XOR}.



Usa el decodificador AXXIUS

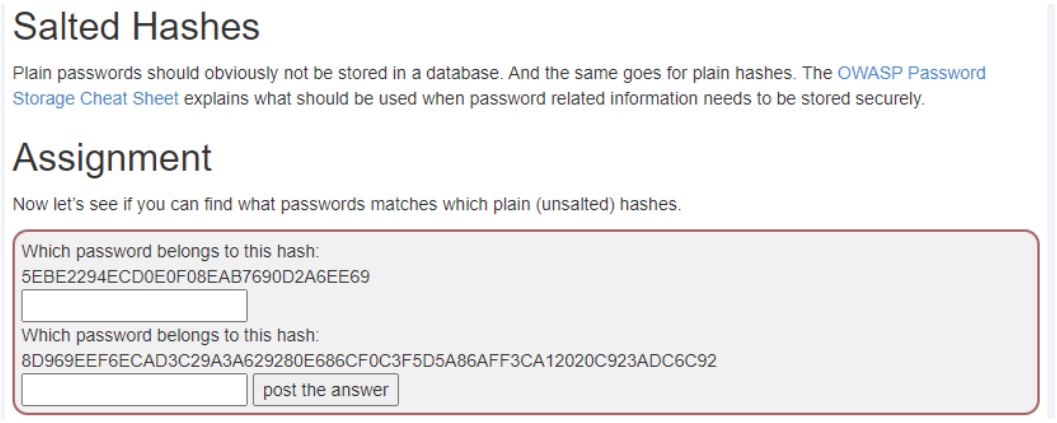
https://strelitzia.net/wasXORdecoder/wasXORdecoder.html



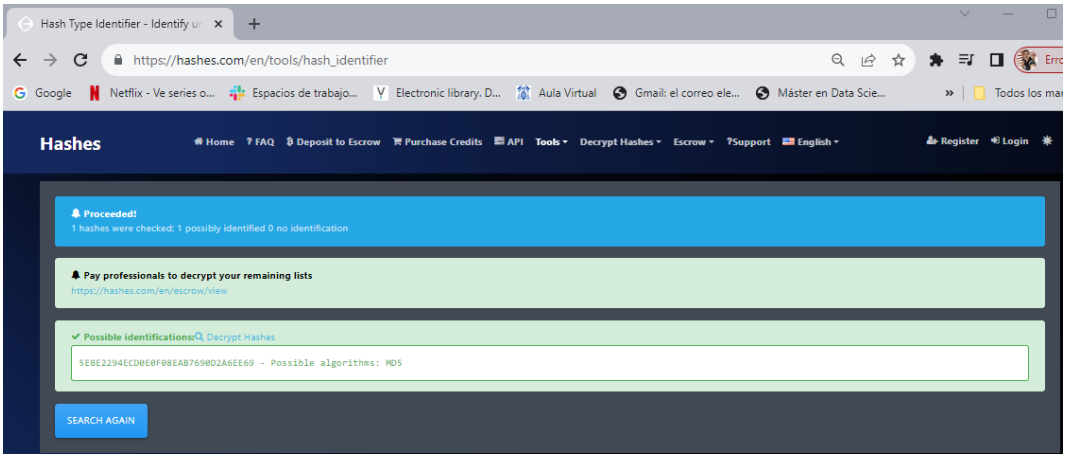
1.2.3 Plain Hashing

En este ejercicio tienes que decodificar dos hashes. Usa Hashes.com para identificar el tipo de algoritmo usado para codificar el hash.

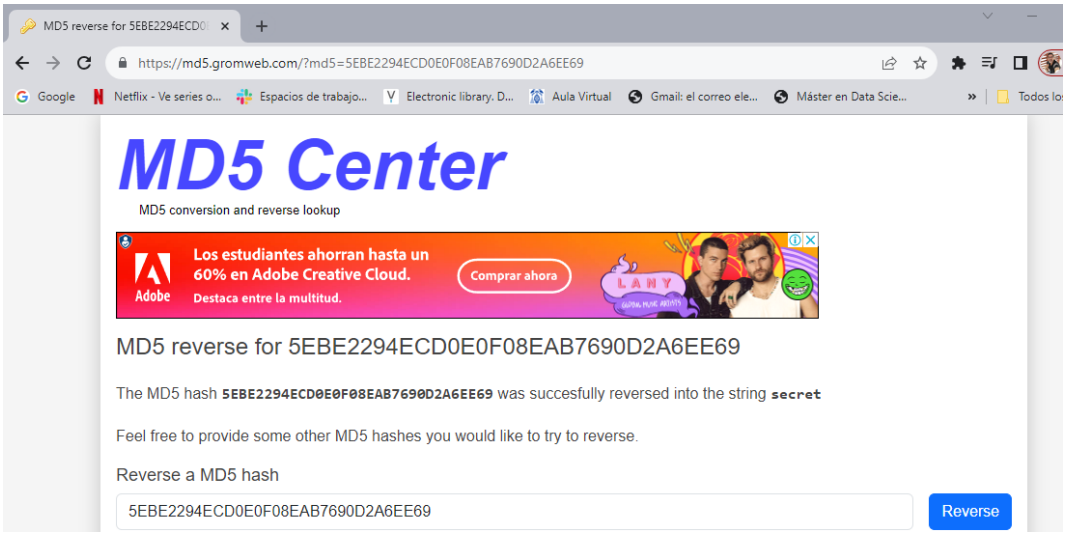
https://hashes.com/en/tools/hash_identifier



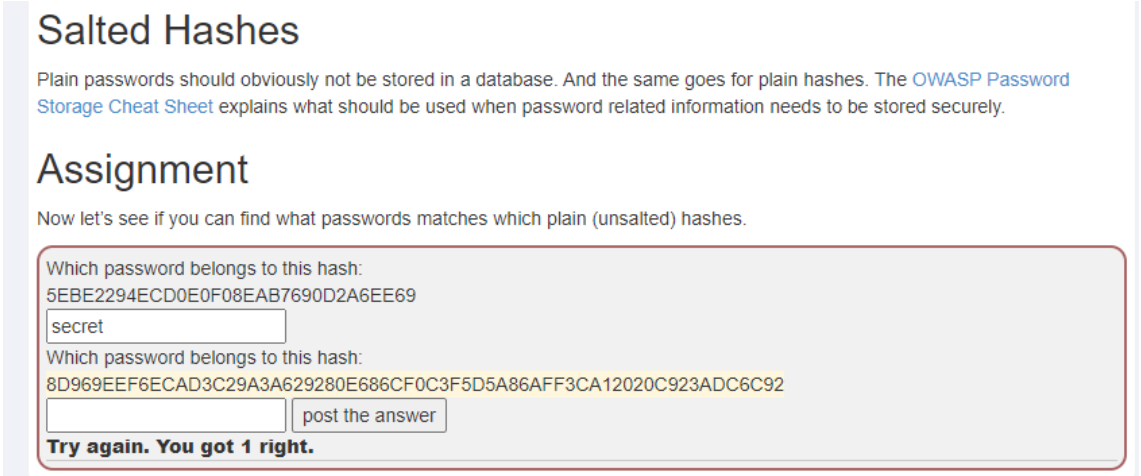
Identifica el tipo de algoritmo.



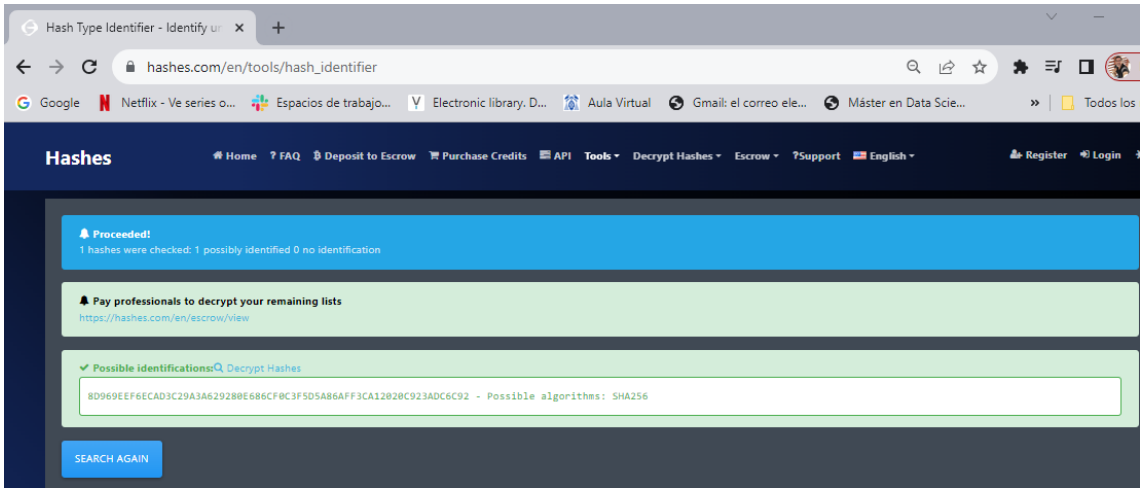
Sabemos que el algoritmo es MD5 decodifica con https://md5.gromweb.com/ y obtén el texto claro.



Prueba el texto



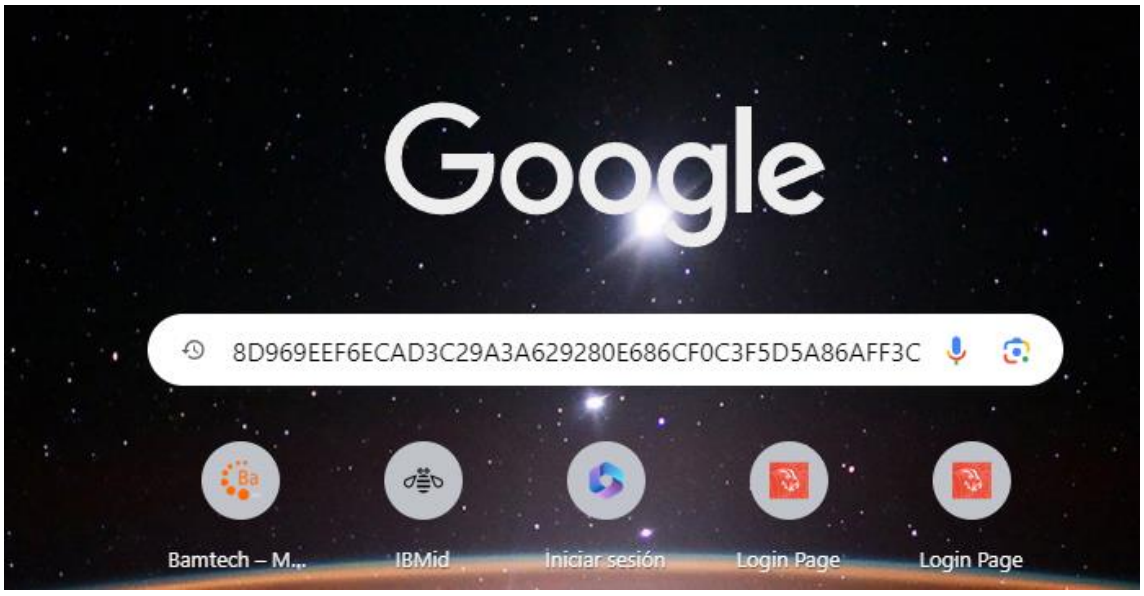
Falta el otro hash, identifica e algoritmo del hash.



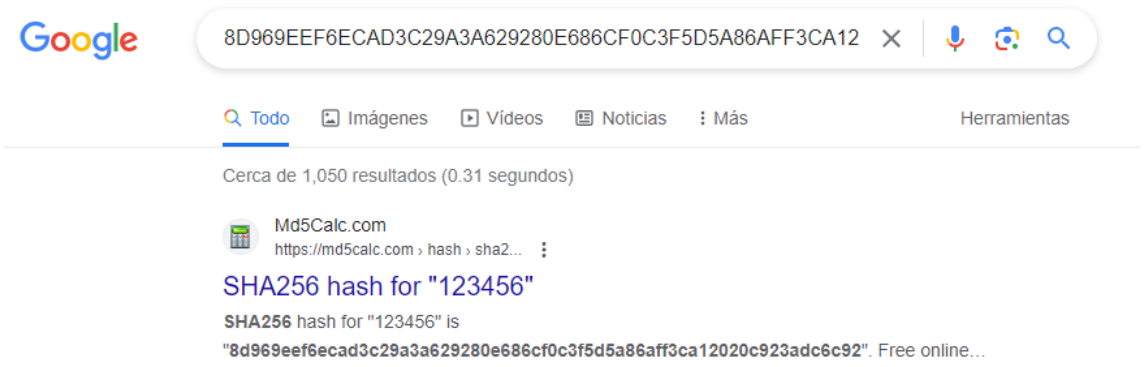
Podemos decodificarlo con el buscador de Google. Colocamos en el buscador el valor del hash y el algoritmo. Ejemplo:

8D969EEF6ECAD3C29A3A629280E686CF0C3F5D5A86AFF3CA12020C923ADC6C92

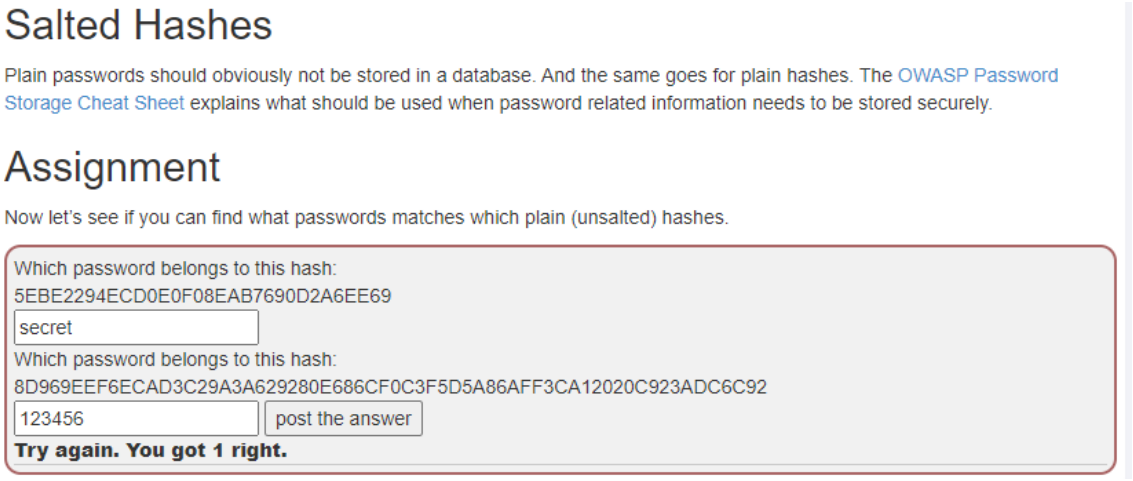
SHA256



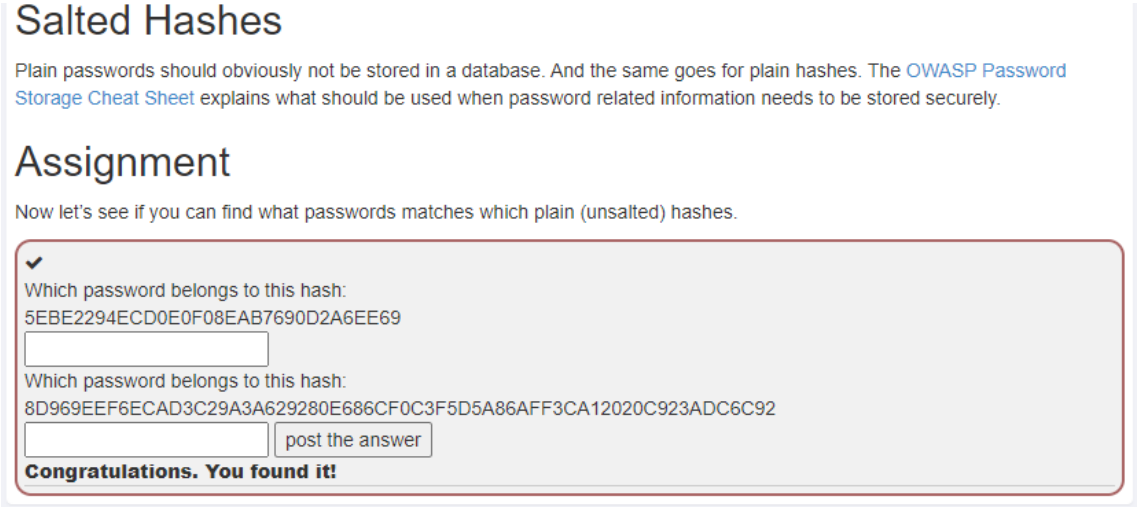
El texto claro es “123456”.



Escribimos en las cajas.



Verificamos que paso la prueba.



1.3 RAW Signatures

En este ejercicio se requiere experiencia en OpenSSL, se trata de obtener el modulus y el signature a partir de una llave privada.

Crea los siguientes archivos

get_signature.java

```
import java.nio.charset.Charset;

import java.security.KeyFactory;

import java.security.PrivateKey;

import java.security.Signature;

import java.security.spec.PKCS8EncodedKeySpec;
```

```
import java.util.Base64;

public class get_signature{

 public static void main(String[] args) throws Exception{

      String message = System.getProperty("mod");

      String privateKey = System.getProperty("key");

     final PrivateKey key = getPrivateKeyFromPEM(privateKey);

     final Signature instance = Signature.getInstance("SHA256withRSA");

     instance.initSign(key);

     instance.update(message.getBytes("UTF-8"));

     final String signature = new String
(Base64.getEncoder().encode(instance.sign()),Charset.forName("UTF-8"));

     String header = "";

     for (int i=0; i<20; i++) {

       header += "-";

     }

     System.out.println();

    System.out.println(header + "New Signature" + header +"\n" + signature + "\n" +  header + "End
Signature" + header);


 }
```

```
 public static PrivateKey getPrivateKeyFromPEM(String  privateKeyPem) throws Exception{

       privateKeyPem = privateKeyPem.replace("-----BEGIN PRIVATE KEY-----","");

       privateKeyPem = privateKeyPem.replace("-----END PRIVATE KEY-----","");

       privateKeyPem = privateKeyPem.replace("\n","");

        final byte [] decoded = Base64.getDecoder().decode(privateKeyPem);

        final PKCS8EncodedKeySpec spec= new PKCS8EncodedKeySpec(decoded);

        final KeyFactory kf = KeyFactory.getInstance("RSA");

        return kf.generatePrivate(spec);
```

```
 }

}
```

Luego crea el script

get_modulus.sh

```
#!/bin/bash

privateKey=$1

privateKey=$(echo $privateKey | tr  -d '[:space:]')

echo -----BEGIN PRIVATE KEY----- > private.key

echo $privateKey | sed -e "s/.\{64\}/&\n/g" >> private.key

echo -----END PRIVATE KEY----- >> private.key

echo

echo

cat private.key

echo

echo

sleep 1

mod=$(openssl rsa -in private.key  -noout -modulus | cut -d'=' -f2)

echo Modulus: $mod

sleep 1

echo

echo Starting Java Signature Process ...

echo

java -Dkey=$privateKey -Dmod=$mod get_signature.java
```

Ejecuta el script.

```
[root@centos-s-1vcpu-1gb-35gb-intel-sfo3-01 ~]# sh get_modulus.sh MIIEuwIBADANBgkqhkiG9w0BAQEFAASCBKUwggShAgEAAoIBAQCAo0TMP0Ps5/cngFKRaOr
37gXhUPvRMQ1xQT4tAruAzh6iFHMkTyVwqRKL7TelQ5hn71+pj9eOkr7fJ1Gi4l2M+2GPslmV6K7UNT/oK/yh21lrZ1zn13F3OsbdCoXM+TIPHnsqLurnY4Gzjmjt46VWaV5pI19s
fKbiRxrULFqmj74838ZPZAEGLqR2Ja5A0mkpZg8rVg/QV/BN/95mZJtnzqn7nBiR8RDQDVJX3T0VnEH7DRbCZHU1aut82WG7PoQ7wLkhyBYjzqP1zK+mOlcKLRzRBYMR4q/hfVIf0
M0NWSNu5FmZhaF1+yXp7prwfYiybBP1vCD7xlLjIphU3k49AgEFAoIBAAzdIHrTIGR9mLcmbqgkF38xZvzuf5TrTiTs7J4Z34zhaXabpR07HVgQ6EExUl1Tjz2X7/dbLyfbeXy3Ic
N9CVrl8Cf4PCiXRK7SH/2d/6nJIoq9iUpiWCWF4K+0QJR/Uc5P2R3RfePwWcUn10r9KiI9vKQdIyRy3X0HHEhq1d10oeVaGHWhiJ4UfagFNWsvx4RSfFdLrBx3plCpbIe8FqZDT6C
w2x3HaFgRoTTDzh5ME3Of8ulaWygQJrfcI+29BEEw2O83cgdXeBZswKosgRx1u7jIxQA0VslzEhfkKdPEKErNW3z4BKcDzIEFww5SAk12ZDSM/MtBtyI52S56+p0CgYEAvXy/twZs
YI0/bcBz3vbCfiDV9DyOILlFY7MLlYU9OA14cpg6SwE/oVUzNze0encCaF9whgwif45dqwxGLwk7EixjpKal+KD2xaEvvFXH+ys/xJndlDSTLcCG3Wo9VuuLeh0/nHEDxVhk7Pn0c
fWfJwhD2FZ7FZALrX1zbSseCSkCgYEArcqbGrCoSkwiTiV9uRoyH91XP4XTNfpmDXRWC5POgNu1Gxx5PmbnXAopzxCCtW+hFxddE5i6ZhZrvUH5y2ihR4N0myL3qTltLXRdbKq1UM
JUAv9QPNtEHkJ3v8ap09wkFmWfKkbRrZdwO+XAimuqSqOIUZG/K3spfg9sr1jyevUCgYBxsT/UNw3Th/Lbc3i4+nSyE7OSirutO/ZvBQbzT/GICBURjolgM/L6ZlHt7jkWR2fYOUO
DoOF/u9HNB11Pa/A+GqIvlzBh+i2pxxykM3f9GfMPj4TyhfHoQFDrP75nWiB8d/L3dwJDNQla/F93k1990cJOmknZvNPOfnh0s3hr5QKBgQCLCHwVWiA7cBtxt5fHSCgZfd9mBKj3
+4TXkETWD9hnFip8FsdlHx981O4/QGiRJhp4332pRvuE3ryXZ/sJIIEGAsOvTyyHYSQkXReKIiqnAdzPMqaXFdAYNSyZa7sP41AR6uW7a9rxRfNjHjOh77tu6Tna2v9V/CExpfCMR
yhiXQKBgBmx+yQXDaY1RwXxNq92/THMxwRLMQ7V90/k85anDwsoEQW7ARFZWNkIqc7zeruFDeiO3VT9CYHKJaoTJlpx9SqTkF9Q0HLF5Wh+0ZzEXN8RHWZ0zsqYBYV4Iw/U81vrXN
UIHL4nksHyBb+9CWFRSanZW+VRZacREZ3OSYJ2XaTN
```

Obtén los resultados y pega los resultados en las cajas.

```
Modulus: 80A344CC3F43ECE7F72780529168EAF7EE05E150FBD1310D71413E2D02BB80CE1EA21473244F2570A9128BED37A5439867EF5FA98FD78E92BEDF2751A2E25D8C
FB618FB25995E8AED4353FE82BFCA1DB596B675CE7D771773AC6DD0A85CCF9320F1E7B2A2EEAE76381B38E68EDE3A556695E69235F6C7CA6E2471AD42C5AA68FBE3CDFC64
F6401062EA47625AE40D26929660F2B560FD057F04DFFDE66649B67CEA9FB9C1891F110D00D5257DD3D159C41FB0D16C26475356AEB7CD961BB3E843BC0B921C81623CEA3
F5CCAFA63A570A2D1CD1058311E2AFE17D521FD0CD0D59236EE4599985A175FB25E9EE9AF07D88B26C13F5BC20FBC652E3229854DE4E3D

Starting Java Signature Process ...


--------------------New Signature--------------------
JEzMDsSUuyeZX5XnGejmZeYZyu2cAjfUT63mMU5oEhVGQnJ/eC02bhQDoS5XaaNGiLPzX0zsTjaPWUp+jC4rXMRi6Dwrg7V2Pdv3ED4DkaCwq568ittmLqTjLG5fZuP8yvUH6/Ols
ySvih/ag9igYrTzlmPPlP1t0B/Se8R5ly/cdH9DtE/6hMJQvIOeclC13FctW78Orcpnbmka3raajTr3JRTktHKCmq5/X9gkdqaCSBRpT0b+HKq3nsdTdz4jUiRWBmAUAeZYCc3nGL
WtZBkdu0r6/DZkSyqjJYh5CdCpGxvVN59EbQtlZxrVyOpV7Y0ZphZtq0SAxau1MP4OHg==
--------------------End Signature--------------------
[root@centos-s-1vcpu-1gb-35gb-intel-sfo3-01 ~]#
```

Assignment

Here is a simple assignment. A private RSA key is sent to you. Determine the modulus of the RSA key as a hex string, and calculate a signature for that hex string using the key. The exercise requires some experience with OpenSSL. You can search on the Internet for useful commands and/or use the HINTS button to get some tips.

Now suppose you have the following private key:

```
5anDwsoEQW7ARFZWNKIqc7zeruFDeiO3VT9CYHKJaoTJlpx9SqTkF9Q0HLF5Wh+0ZzEXN8RHWZ0zsqYBYV4Iw/U81vrXNUIHL4nksHyBb+9CWFRSanZW+VRZacREZ3OSYJ2XaTN
```

Then what was the modulus of the public key 80A344CC3F43ECE7F727 and now provide a signature for us based on that modulus

JEzMDsSUuyeZX5XnGejm post the answer

1.4 Find Secrets in Docker Container

En este ejercicio usaremos el software de contenedores docker.

Assignment

In this exercise you need to retrieve a secret that has accidentally been left inside a docker container image. With this secret, you can decrypt the following message: **U2FsdGVkX199jgh5oANElFdtCxIEvdEvciLi+v+5loE+VCuy6Ii0b+5byb5DXp32RPmT02Ek1pf55ctQN+DHbwCPiVRfFQamDmbHBUpD7as=**. You can decrypt the message by logging in to the running container (docker exec …) and getting access to the password file located in /root. Then use the openssl command inside the container (for portability issues in openssl on Windows/Mac/Linux) You can find the secret in the following docker image, which you can start as:

```
docker run -d webgoat/assignments:findthesecret
```

```
echo "U2FsdGVkX199jgh5oANElFdtCxIEvdEvciLi+v+5loE+VCuy6Ii0b+5byb5DXp32RPmT02Ek1pf55ctQN+DHbwCPiVRfFQamDmbHBUpD7as=" | openssl enc -aes
-256-cbc -d -a -kfile ....
```

What is the unencrypted message

and what is the name of the file that stored the password

post the answer

Ejecuta el contenedor docker indicado.

```
[admin@server1 ~]$ docker pull webgoat/assignments:findthesecret
findthesecret: Pulling from webgoat/assignments
5e6ec7f28fb7: Pull complete
1cf4e4a3f534: Pull complete
5d9d21aca480: Pull complete
0a126fb8ec28: Pull complete
1904df324545: Pull complete
e6d9d96381c8: Pull complete
d6419a981ec6: Pull complete
4cf180de4a1f: Pull complete
ff2e10214d79: Pull complete
Digest: sha256:3fba41f35dbfac1daf7465ce0869c076d3cdef017e710dbec6d273cc9334d4a6
Status: Downloaded newer image for webgoat/assignments:findthesecret
docker.io/webgoat/assignments:findthesecret
[admin@server1 ~]$ docker run -d webgoat/assignments:findthesecret
6706417a7a9cc067c1897d0d8ab6806788c6b088bf97116da271607b59c22e7e
[admin@server1 ~]$
```

Ahora debes superar el desafío de ubicar el archivo con la clave secreta y obtener el mensaje claro que ha sido codificado dentro del contenedor.

```
[admin@server1 ~]$ docker exec -it 6706417a7a9cc067c1897d0d8ab6806788c6b088bf97116da271607b59c22e7e /bin/bash
webgoat@6706417a7a9c:/$ cd root
bash: cd: root: Permission denied
webgoat@6706417a7a9c:/$ echo "El acceso no esta permitido"
El acceso no esta permitido
webgoat@6706417a7a9c:/$
```

Vamos a obtener el archivo de configuración de niveles de acceso por id de usuario en Linux, lo vamos a modificar para que el usuario Webgoat tenga privilegios de root dentro del contenedor.

```
[admin@server1 ~]$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED         STATUS         PORTS     NAMES
6706417a7a9c   webgoat/assignments:findthesecret   "/bin/bash /home/web…"   5 minutes ago   Up 5 minutes             serene_jennings
[admin@server1 ~]$ docker cp 6706417a7a9c:/etc/passwd ./passwd
Successfully copied 2.56kB to /home/admin/passwd
[admin@server1 ~]$ vim passwd
```

Observa que el usuario webgoat tiene id:100 y groupid:100, vamos a cambiar estos valor por 0 para que el Linux del contenedor asuma que webgoat es super usuario.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
webgoat:x:1000:1000::/home/webgoat:
~
```

Actualicemos los valores y copiamos el archivo al contendor.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
webgoat:x:0:0::/home/webgoat:
~
```

Copia el archivo passwd al contendor.

```
[admin@server1 ~]$ docker cp ./passwd 6706417a7a9c:/etc/passwd
Successfully copied 2.56kB to 6706417a7a9c:/etc/passwd
[admin@server1 ~]$
```

Iniciamos una sesión ssh en el contenedor y verifica que ahora podemos ver el contenido del directorio /root

```
[admin@server1 ~]$ docker exec -it 6706417a7a9c /bin/bash
root@6706417a7a9c:/# cd /root
root@6706417a7a9c:~# ls
default_secret
root@6706417a7a9c:~# cat default_secret
ThisIsMySecretPassw0rdF0rY0u
root@6706417a7a9c:~#
```

¡El archivo que estábamos buscando se llama **default_secret**, Eureka!, lo hemos encontrado.

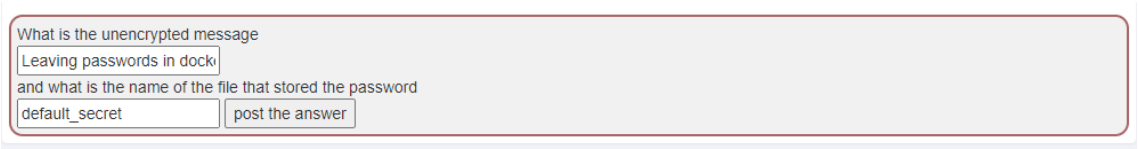
Ahora podemos obtener el mensaje codificado usando el archivo que contiene la clave para decodificar el mensaje codificado. Ejecuta el siguiente comando en el contenedor.

```
echo "U2FsdGVkX199jgh5oANElFdtCxIEvdEvciLi+v+5loE+VCuy6Ii0b+5byb5DXp32RPmT02E
k1pf55ctQN+DHbwCPiVRfFQamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d -a -kfil
e default_secret
```

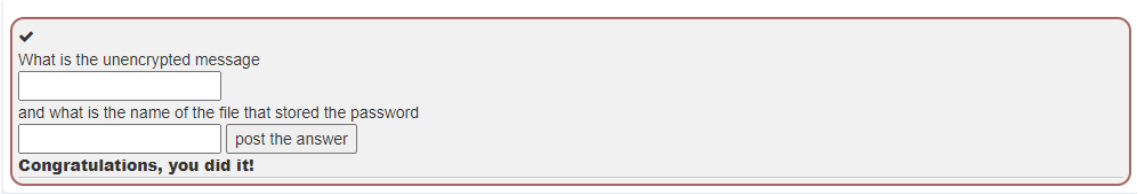
```
root@6706417a7a9c:~# echo "U2FsdGVkX199jgh5oANElFdtCxIEvdEvciLi+v+5loE+VCuy6Ii0b+5byb5DXp32RPmT02Ek1pf55ctQN+DHbwCPiVRfFQamDmbHBUpD7as=
" | openssl enc -aes-256-cbc -d -a -kfile ./default_secret
Leaving passwords in docker images is not so secureroot@6706417a7a9c:~#
```

El mensaje codificado es **“Leaving passwords in docker images is not so secure**”

Escribe los resultados en la caja de la aplicación Webgoat para superar el ejercicio.



Haz clic en “post the answer”.



Felicidades , haz superado el ejercicio.