

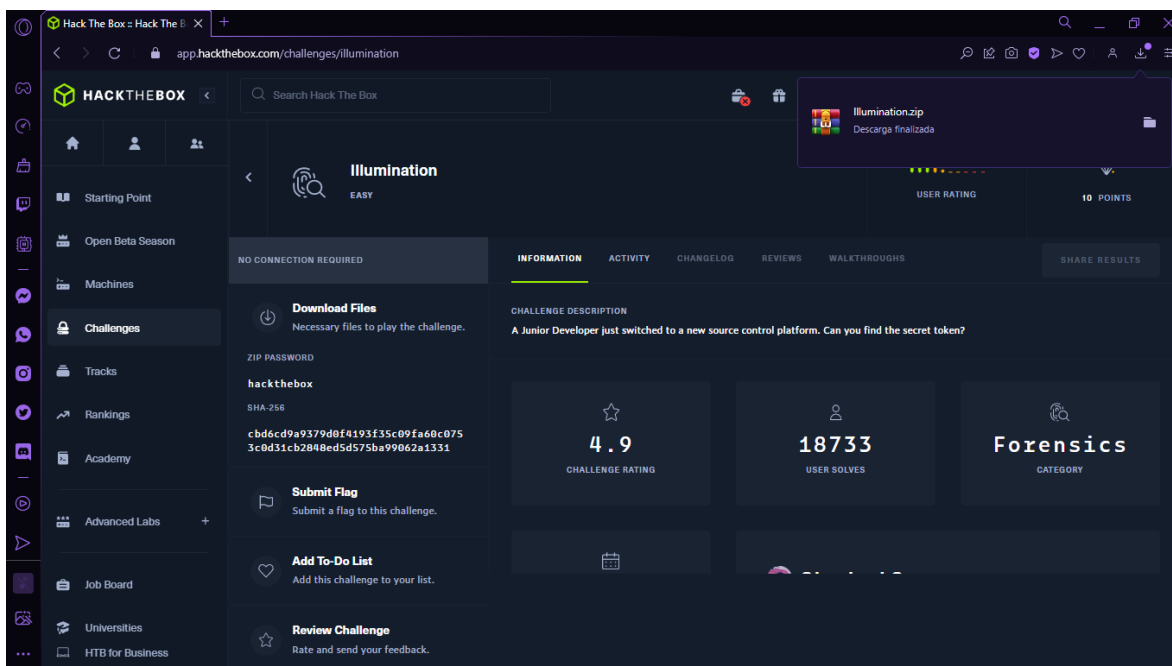
Tarea: Illumination HTB

Alumno; Quintana Escamilla Roberto Carlos

"Un desarrollador junior acaba de cambiar a una nueva plataforma de control de código fuente. ¿Puedes encontrar el token secreto?", para ello se nos proporciona un archivo zip y una contraseña para descomprimirlo y un hash de verificación sha 256.

Pass: hackthebox

Sha256: cbd6cd9a9379d0f4193f35c09fa60c0753c0d31cb2848ed5d575ba99062a1331



Para descomprimirlo utilizaremos una consola tipo Linux (como repaso para la clase). Primero confirmamos que se encuentra en formato zip para utilizar el comando adecuado de descompresión. Además de ello confirmaremos que se trata del archivo original verificando con shasum 256.

```
MINGW64:/c/Users/2im3q/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
$ ls
09_02/  GrabarClase/      Illumination_HTB.docx  cuaderno.docx  tareas/
21_02/  Illumination.zip  chat_the_box/          notepad/

2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
$ shasum -a 256 Illumination.zip
cbd6cd9a9379d0f4193f35c09fa60c0753c0d31cb2848ed5d575ba99062a1331 *Illumination.z
ip

2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
$
```

Hash proporcionado :

cbd6cd9a9379d0f4193f35c09fa60c0753c0d31cb2848ed5d575ba99062a1331

Has obtenido:

cbd6cd9a9379d0f4193f35c09fa60c0753c0d31cb2848ed5d575ba99062a1331

Una vez hecho esto con un ls podemos utilizar unzip, donde ingresaremos el password otorgado.

```
MINGW64:/c/Users/2im3q/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
$ unzip Illumination.zip
Archive:  Illumination.zip
  creating:  Illumination.JS/
  creating:  Illumination.JS/.git/
[Illumination.zip] Illumination.JS/.git/COMMIT_EDITMSG password:
```

```
MINGW64:/c/Users/2im3q/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
$ unzip Illumination.zip
Archive:  Illumination.zip
  creating: Illumination.JS/
  creating: Illumination.JS/.git/
[ Illumination.zip] Illumination.JS/.git/COMMIT_EDITMSG password:
  inflating: Illumination.JS/.git/COMMIT_EDITMSG
  inflating: Illumination.JS/.git/config
  inflating: Illumination.JS/.git/description
  extracting: Illumination.JS/.git/HEAD
  creating: Illumination.JS/.git/hooks/
  inflating: Illumination.JS/.git/hooks/applypatch-msg.sample
  inflating: Illumination.JS/.git/hooks/commit-msg.sample
  inflating: Illumination.JS/.git/hooks/fsmonitor-watchman.sample
  inflating: Illumination.JS/.git/hooks/post-update.sample
  inflating: Illumination.JS/.git/hooks/pre-applypatch.sample
  inflating: Illumination.JS/.git/hooks/pre-commit.sample
  inflating: Illumination.JS/.git/hooks/pre-push.sample
  inflating: Illumination.JS/.git/hooks/pre-rebase.sample
  inflating: Illumination.JS/.git/hooks/pre-receive.sample
  inflating: Illumination.JS/.git/hooks/prepare-commit-msg.sample
  inflating: Illumination.JS/.git/hooks/update.sample
  inflating: Illumination.JS/.git/index
  creating: Illumination.JS/.git/info/
  inflating: Illumination.JS/.git/info/exclude
  creating: Illumination.JS/.git/logs/
  inflating: Illumination.JS/.git/logs/HEAD
  creating: Illumination.JS/.git/logs/refs/
  creating: Illumination.JS/.git/logs/refs/heads/
  inflating: Illumination.JS/.git/logs/refs/heads/master
  creating: Illumination.JS/.git/objects/
  creating: Illumination.JS/.git/objects/11/
  extracting: Illumination.JS/.git/objects/11/ce945904a5061f42d1d81276106b51dcec4
b39
```

Notaremos que se nos creo la carpeta "Illumination.js" (no es un archivo .js lo podemos saber por que se crearon mas direcciones utilizando este nombre como carpeta raíz y por el echo de estar señalado con mayúsculas) y además dentro de ella se creo la carpeta ".git" por lo que podemos sospechar que "Illumination.js" es un repositorio de git hub. Actualmente estamos trabajando con una consola proporcionada por github por lo que podemos confirmarlo simplemente entrando al repositorio con cd y revisando que en la dirección actual se buestre un Branch con letras azules. En caso de estar en una consola que no cuente con esta etiqueta visual podemos intentar ejecutar un git log y ver si se han generados commits. Esto ultimo también lo haremos debido a que el ejercicio nos indicaba que el desarrollador había cambiado de plataforma de control, por lo que al ser inexperto podría haber provocado una falla en la seguridad.

```
MINGW64:/c/Users/2im3q/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS
2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad
$ cd Illumination.JS

2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS
(master)
$ git log
commit edc5aabf933f6bb161ceca6cf7d0d2160ce333ec (HEAD -> master)
Author: SherlockSec <dan@lights.htb>
Date: Fri May 31 14:16:43 2019 +0100

    Added some whitespace for readability!

commit 47241a47f62ada864ec74bd6dedc4d33f4374699
Author: SherlockSec <dan@lights.htb>
Date: Fri May 31 12:00:54 2019 +0100

    Thanks to contributors, I removed the unique token as it was a security risk. Thanks for r
eporting responsibly!

commit ddc606f8fa05c363ea4de20f31834e97dd527381
Author: SherlockSec <dan@lights.htb>
Date: Fri May 31 09:14:04 2019 +0100

    Added some more comments for the lovely contributors! Thanks for helping out!
```

Si somos atentos notaremos un commit comentado con “Thanks to contributors, I removed the unique token as it was a security risk. Thanks for reporting responsibly!” por lo que podemos intuir que puede existir alguna vulnerabilidad en la versión anterior a ese commit. Para verificarlo primero observaremos que fue lo que se modifico en este commit con el comando git show con el hash correspondiente al commit (47241a47f62ada864ec74bd6dedc4d33f4374699).

Nota: si se tiene problemas para salir del listado de git log basta con escribir “q”.

```
MINGW64:/c/Users/2im3q/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS
(master)
$ git show 47241a47f62ada864ec74bd6dedc4d33f4374699
commit 47241a47f62ada864ec74bd6dedc4d33f4374699
Author: SherlockSec <dan@lights.htb>
Date: Fri May 31 12:00:54 2019 +0100

    Thanks to contributors, I removed the unique token as it was a security risk. Thanks for r
eporting responsibly!

diff --git a/config.json b/config.json
index 316dc21..6735aa6 100644
--- a/config.json
+++ b/config.json
@@ -1,6 +1,6 @@
 {
-    "token": "SFRce3YzcnNpMG5fYzBudHIwbF9hbV9JX3JpZ2h0P30=",
+    "token": "Replace me with token when in use! Security Risk!",
     "prefix": "~",
     "lightNum": "1337",
     "username": "UmVkieh1cnJpbmcsIHJlYWQgdGhIEpTIGNhcmVmdWxseQ==",

2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS
(master)
$
```

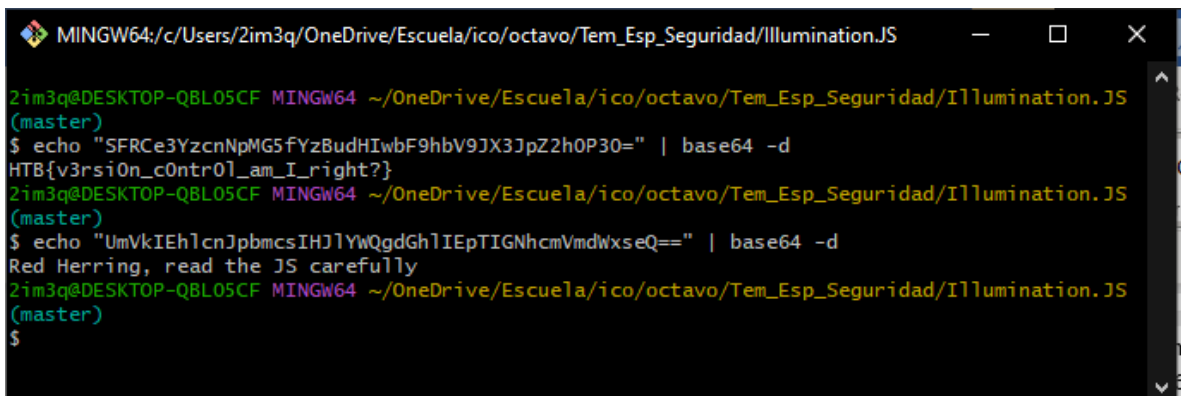
Notamos que se remplazó una línea en el archivo “config.json” , la cual antes mostraba un token codificado en base 64 (esto lo podemos sospechar debido al uso de caracteres alfanuméricos en mayúsculas y minúsculas además del signo igual al final de la cadena), esta fue remplazada por la leyenda “Replace me with token when in use! Security Risk!”, que parece ser mas que nada una instrucción para las desarrolladoras. También encontramos otro hash en base 64 donde se corresponde a la variable username.

Token: SFRCE3YzcnNpMG5fYzBudHIwbF9hbV9JX3JpZ2h0P30=

Username: UmVklEhlcnJpbmcsIHJlYWQgdGhIEpTIGNhcmVmdWxseQ==

Para decodificarlas utilizaremos el comando echo para utilizar los respectivos tokens como entrada del comando base64 con el modificador -d, esto con ayuda del carácter | como se muestra en el siguiente ejemplo.

echo “cadena_ejemplo” | base64 -d



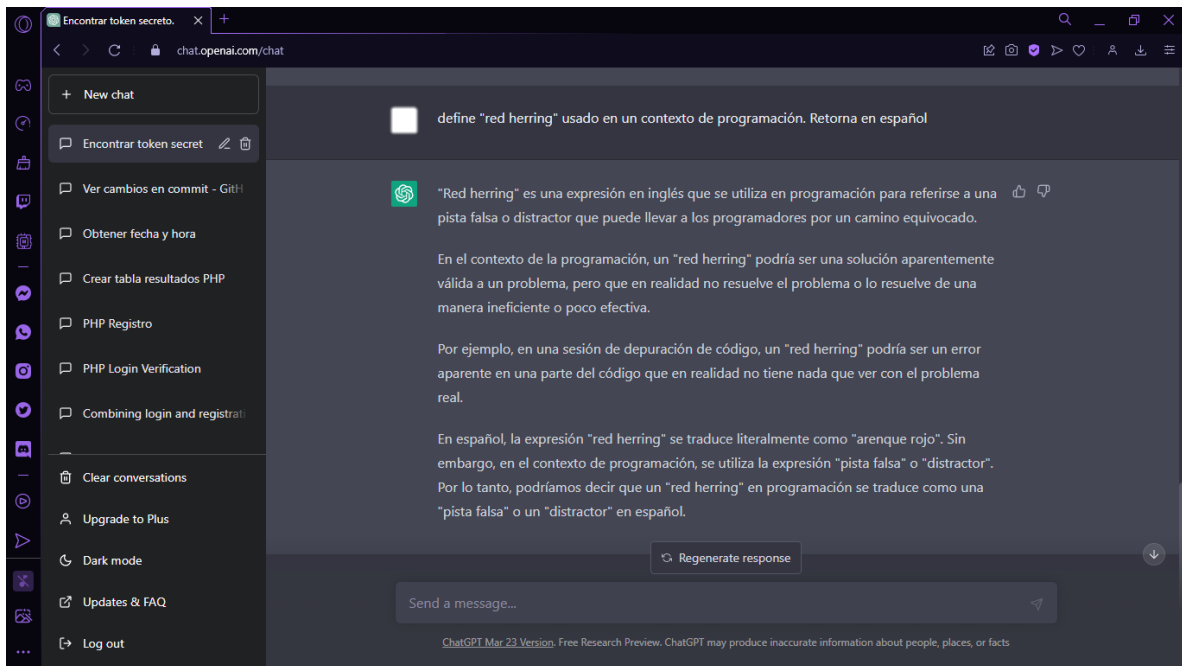
```
MINGW64: c:/Users/Zim3q/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS
Zim3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS
(master)
$ echo "SFRCE3YzcnNpMG5fYzBudHIwbF9hbV9JX3JpZ2h0P30=" | base64 -d
HTB{v3rsi0n_c0ntr0l_am_I_right?}
Zim3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS
(master)
$ echo "UmVklEhlcnJpbmcsIHJlYWQgdGhIEpTIGNhcmVmdWxseQ==" | base64 -d
Red Herring, read the JS carefully
Zim3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS
(master)
$
```

Token: HTB{v3rsi0n_c0ntr0l_am_I_right?}

flag : HTB{v3rsi0n_c0ntr0l_am_I_right?}

Username: Red Herring, read the JS carefully

Observamos que la cadena resultado de decodificar el Token antiguo es una probable flag de HTB, mientras que el de username indica un “Red Herring” . Como curiosidad si investigamos que significa utilizando chat GTP con el prompt{define "red herring" usado en un contexto de programación}. Retorna en español” nos indica que suele usarse como expresión de “pista falsa” por lo que podemos intuir que se colocó en el ejercicio pensando en que primero revisaríamos el código actual y nos encontraríamos con esta pista falsa.



Si revisamos el archivo bot.js que se encuentra en el repositorio podemos encontrar la función de login la cual indica que hay que loguearse con el token secreto.

```

MINGW64~/Users/Zim3q/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/IlluminationJS
$ cat bot.js
//Discord.js
const Discord = require("discord.js");
const client = new Discord.Client();
const fs = require("fs");
var config = JSON.parse(fs.readFileSync("./config.json"));

//Node-Hue-API
var hue = require("node-hue-api"),
    HueApi = hue.HueApi,
    lightState = hue.LightState;

//Display command results in console
var displayResult = function(result) {
    console.log(JSON.stringify(result, null, 2));
};

//Function taken from campushippo.com
var rgbToHex = function (rgb) {
    var hex = Number(rgb).toString(16);
    if (hex.length < 2) {
        hex = "0" + hex;
    }
    return hex;
};

//Function taken from campushippo.com
var fullColorHex = function(r,g,b) {
    var header = "0x"
    var red = rgbToHex(r);
    var green = rgbToHex(g);
    var blue = rgbToHex(b);
    return header+red+green+blue;
};

//Declarations
var host = config.host,
    username = config.username,
    api = new HueApi(host, username),
    state = lightState.create(),
    prefix = config.prefix,
    lightNum = config.lightNum;

//Bot code
client.on("ready", () => {
    console.log("Logged in as " + client.user.tag + "!");
});

client.on("message", message => {
    if (message.author.bot) return; //Ignore bot messages
    if (message.content.indexOf(prefix) !== 0) return; //Ensure prefix is at the beginning

```

```
state = lightState.create(),
prefix = config.prefix,
lightNum = config.lightNum;

//Bot code
client.on("ready", () => {
  console.log('Logged in as ${client.user.tag}!');
});

client.on("message", message => {
  if (message.author.bot) return; //Ignore bot messages
  if (message.content.indexOf(prefix) !== 0) return; //Ensure prefix is at the beginning

  const args = message.content.slice(prefix.length).trim().split(/ +/g); //Split command
  into arguments
  const command = args.shift().toLowerCase();

  switch (command) {

    case "light.off" : //Turn light off
      api.setLightState(lightNum, state.off())
        .then(displayResult)
        .done();
      message.channel.send("Light Off!");
      break;

    case "light.on" : //Turn light on
      api.setLightState(lightNum, state.on())
        .then(displayResult)
        .done();
      message.channel.send("Light On!");
      break;

    case "light.rgb" : //change light colour
      let r = args[0];
      let g = args[1];
      let b = args[2];
      api.setLightState(lightNum, state.on().rgb(r, g, b))
        .then(displayResult)
        .done();
      const embed = new Discord.RichEmbed()
        .setTitle('Light Colour Change')
        .setColor('RANDOM')
        .setDescription('Red Value: ${r}. Green Value: ${g}. Blue Value: ${b}');
      message.channel.send(embed);
      break;

    case "light.switch" : //Switch to different light
      let num = args[0];
      lightNum = num;
      //fs.writeFile('./config.json', JSON.stringify(config))
      message.channel.send(`Light number switched to ${lightNum}`);
      break;
  }
});

client.login(Buffer.from(config.token, 'base64').toString('ascii')) //Login with secret token
```

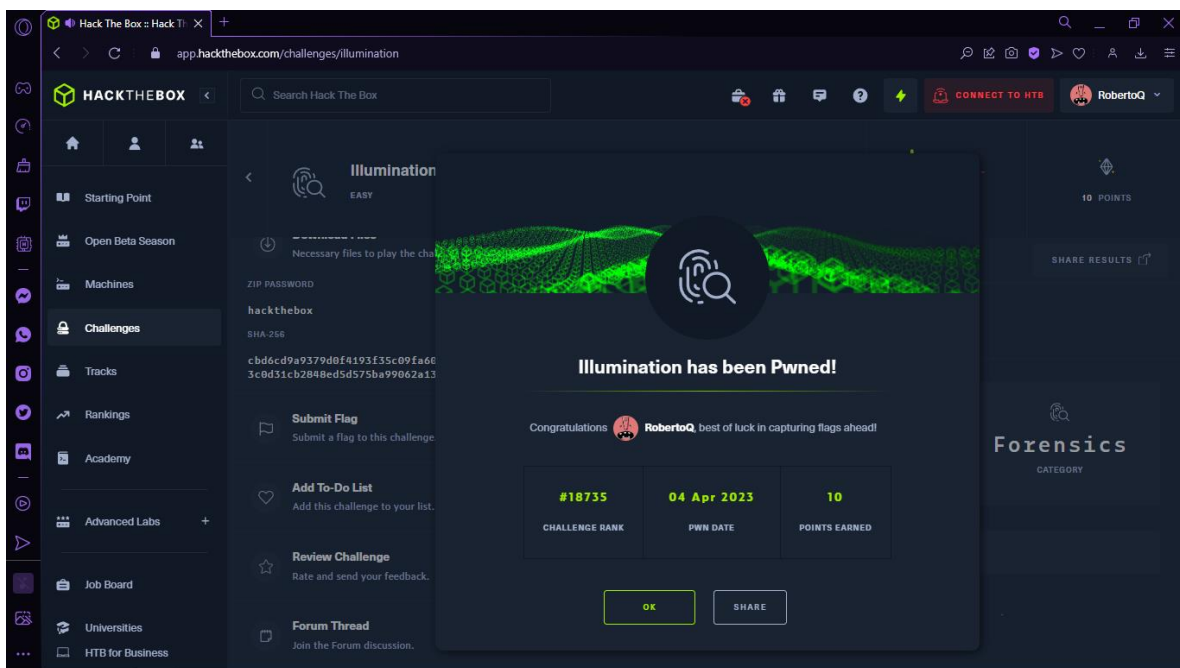
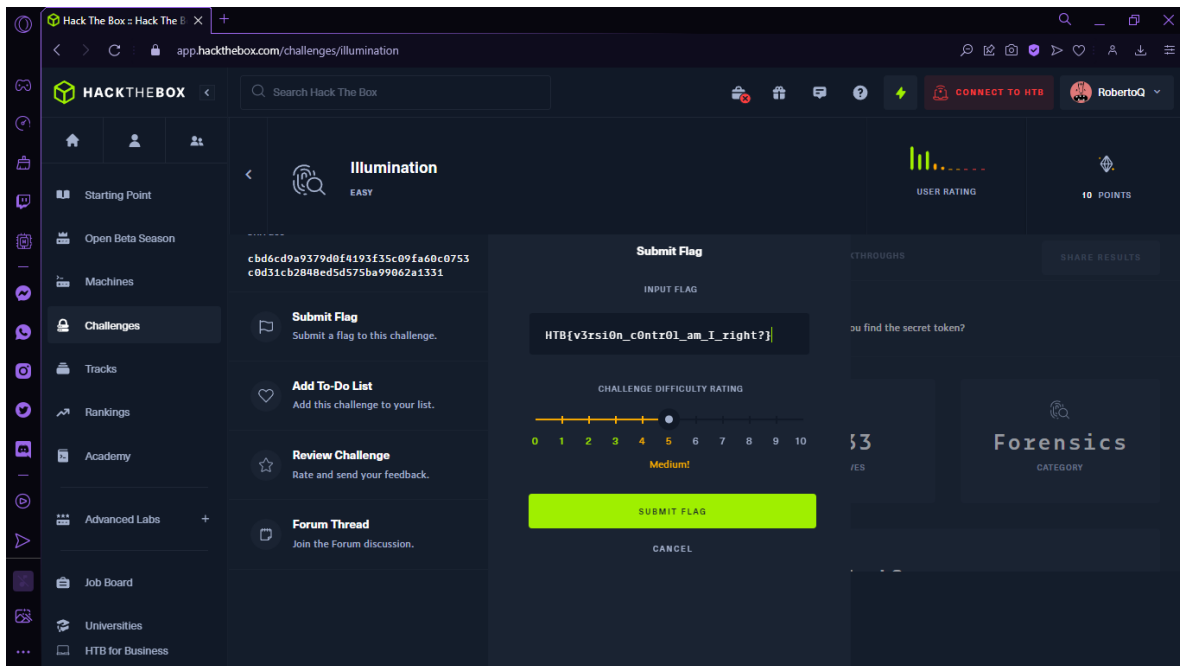
El cual si revisamos config.json es el mismo por el que se cambio en el commit analizado anteriormente

```
MINGW64:/c:/Users/2im3q/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS

2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS (master)
$ cat config.json
{
  "token": "Replace me with token when in use! Security Risk!",
  "prefix": "~",
  "lightNum": "1337",
  "username": "UmVkJElcnJpbmcsIHJlYWQgdGh1IEpTIGNhcmVmdWxseQ==",
  "host": "127.0.0.1"
}
2im3q@DESKTOP-QBL05CF MINGW64 ~/OneDrive/Escuela/ico/octavo/Tem_Esp_Seguridad/Illumination.JS (master)
$
```

Lo cual nos llevaría a sospechar que para resolver el ejercicio habría que encontrar este token secreto, por lo que nos llevaría a realizar el proceso de investigación en el repositorio y por ende a investigar los commits como hicimos anteriormente.

Por ultimo comprobaremos que la flag es correcta.



Vulnerabilidades y mitigaciones

1. La primera razón por la que pudimos atacar de manera sencilla el repositorio es precisamente porque teníamos acceso a él. Esto debido a que se nos otorgó un zip con el repositorio completo lo cual dio lugar a que pudiéramos acceder a versiones pasadas. En este caso era el objetivo que se revisara el sistema de control de versiones. Pero para mitigar riesgos es recomendable únicamente compartir los archivos correspondientes al proyecto final o bien establecer permisos.

2. En este repositorio se dejaron todos los commits incluyendo uno en el cual se especifica que se hizo un cambio en el cual había un riesgo para la integridad del proyecto. Esto es un riesgo por si mismo ya que un atacante se fijara en la declaración de esta vulnerabilidad, por lo cual no es recomendable dejar este tipo de commits. Para mitigarlos se recomienda:
 1. Etiquetar de forma más cuidadosa los commits.
 2. Realizar las etiquetas mediante una cadena cifrada
 3. En caso de que el commit y/o el código modificado sea un riesgo importante, se puede eliminar el commit del historial. Esto debe realizarse con mucha precaución ya que en general atenta contra el propósito del sistema de control de versiones por lo que es recomendable discutirlo primero con el equipo de trabajo. Para ello se utiliza el comando "git rebase -i hash_del_commit_a_eliminar" para que se nos habrá un archivo que contiene la información para administrar los commits, una vez eliminado salvamos y salimos para usar "git push --force"
3. El token se realizo usando una codificación de la flag en lugar de un cifrado. Esto reduce la seguridad debido a que para obtener la cadena original el atacante no necesita ninguna key para revertir la codificación por lo cual si la reconoce puede revertirla fácilmente. Para realizar esta encriptación y mitigar la vulnerabilidad se pueden utilizar diferentes algoritmos matemáticos o incluso aplicaciones las cuales nos facilitan el proceso.

Referencias

- WeLiveSecurity. (2016, diciembre 7). Codificación o cifrado: ¿cuál es la diferencia? [Entrada de blog]. Recuperado el 4 de abril de 2023, de <https://www.welivesecurity.com/la-es/2016/12/07/codificacion-o-cifrado-diferencia/>
- ChatGPT. (2023, 4 de abril). Prompt: define "red herring" usado en un contexto de programación. Recuperado el 4 de abril de 2023, de <https://chat.openai.com/chat/5b099fed-2a6d-4301-a7b4-9215f5a0c553>
- ChatGPT. (2023, 4 de abril). Prompt: describe como ocultar la información de un commit en github por terminal. Recuperado el 4 de abril de 2023, de <https://chat.openai.com/chat/5b099fed-2a6d-4301-a7b4-9215f5a0c553>