

# The 6 Hidden Flags

#challenge

#osint

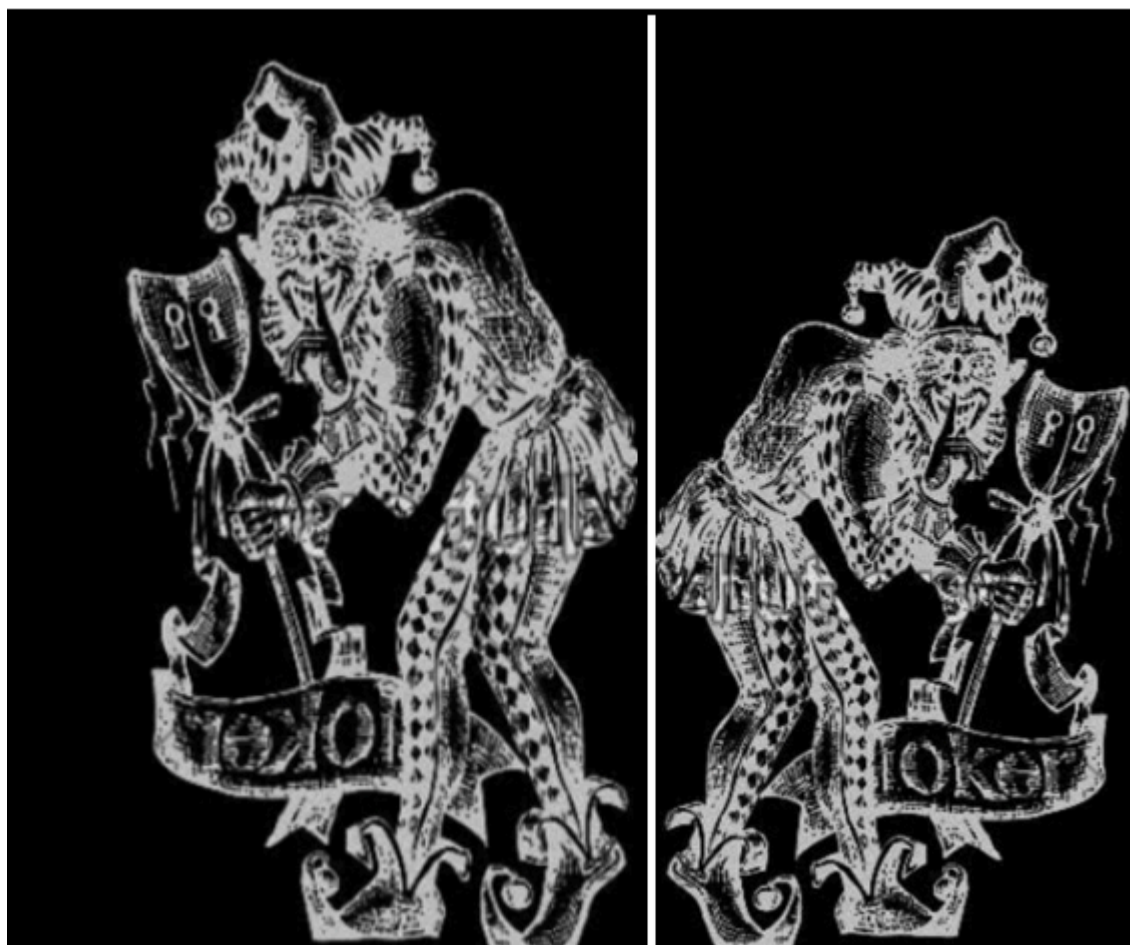
#infoaga

#code-analysis

Primero de todo comentar que hay varias vías para llegar a acceder a los retos ocultos pero yo voy a comentar dos:

## Vía 1

Si nos fijamos en la apariencia de la web vemos todo en casi todo momento una imagen en la parte inferior izquierda de la pantalla que esta al revés. Esto lo podemos saber porque hay unas letras en ella que forman "**Joker**".



Aquí visualizamos la misma imagen mostrada sin invertir. La imagen se llama bg.png pero quizás podríamos relacionar que joker podría ser un recurso o usuario. Así pues si vamos haciendo directory listing añadiendo la palabra joker encontramos el recurso **joker.php**.

## Vía 2

Como curiosos que somos podemos ir a un archivo algo conocido que nos suele mostrar más información sobre algunas webs. Visualizando a ver si existen los recursos comunes para visualizar información del sitio: sitemap.xml , .htaccess , robots.txt. No hay resultados en ningún recurso más que el de robots.txt. El archivo [robots.txt](#) es un archivo destinado a

los Crawlers de internet para gestionar el tipo de archivos , los directorios y algunas paginas que se van a indexar en los navegadores.

Así pues, accedemos al archivo y encontramos toda esta información:

 <https://tramunthackctf.com/robots.txt>

User-agent:

*Disallow: /wp-admin*

*Disallow: /wp-includes*

*Disallow: /wp-login.php*

*Disallow: /wp-content/plugins*

*Disallow: /wp-content/cache*

*Disallow: /wp-content/themes*

*Disallow: /trackback*

*Disallow: /trackback*

*Disallow: //trackback*

*Disallow: //feed//*

*Disallow: /feed*

*Disallow: /?*

*Disallow: /cgi-bin*

*Disallow: /.php\$*

*Disallow: /.inc*  
*Disallow : /\*.gz*

*Allow: /uploads*

*Allow: /.js*

*Allow: /.css*

*Allow: /.png*

*Allow: /.jpg*

*Allow: /.jpeg*

*Allow: /.gif*

*Allow: /.svg*

*Allow: /.webp*

*Allow: /.pdf*

*Disallow: /instructions.php*

*Disallow: /hackerboard.php*

*Disallow: /quests.php*

*Disallow: /machine.php*

*Disallow: /login.php*

*Disallow: /logout.php*

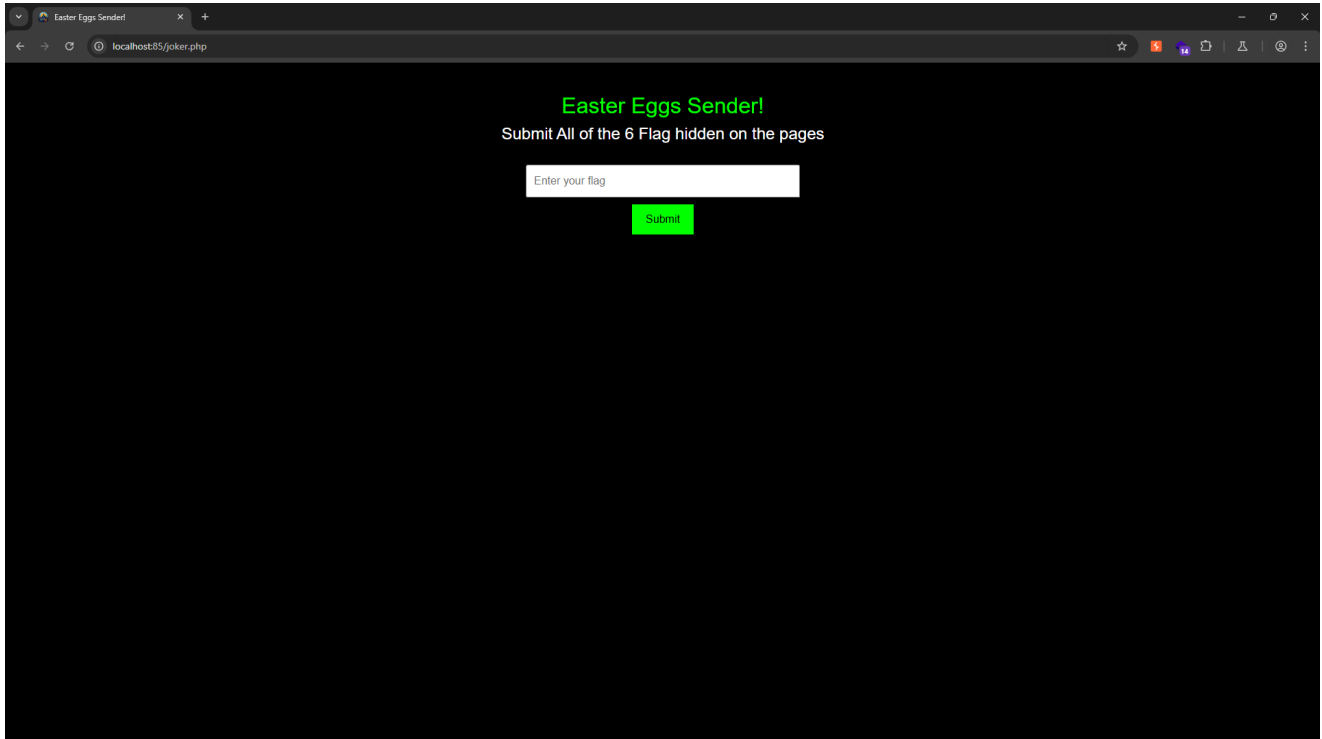
*Disallow: /joker.php*

CTF{697c8a99f6ca116a4f0bd76c3ac04fb7}

Bien pues como podemos observar he resaltado dos líneas que son algo importantes.

- Primero encontramos que se quita el permiso de acceso a la indexación de varias paginas que hemos visto previamente. Seguidamente observamos una nueva pagina que aun no habíamos accedido antes, joker.php.
- También vemos que obtenemos la primera Flag.

Llegamos a la página de Joker.php:



Hemos encontrado lo que parece ser el portal de envío de las 6 Hidden Easter egg flags. Introducimos la flag que hemos encontrado en el archivo robots.txt y ... Flag submitted successfully! Points: 100 , hemos obtenido **+100 puntos** 🙌🙌.

### Easter Eggs

- 1/6 robots.txt
- **Flags Restantes: 5**

Veamos que hay detras de la página de joker.php:

Presionamos `Ctrl + U` para acceder a `view-`

`source:https://tramunthackctf.com/joker.php` y nos muestra el codigo fuente HTML:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Easter Eggs Sender!</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<title>Submit Your Flag</title>
<link rel="stylesheet" href="css/bootstrap4-neon-glow.min.css">
<link rel="stylesheet" href="css/joker.css">
<link href="https://fonts.googleapis.com/css?family=Roboto"
rel="stylesheet">
<ctf>2613d5335779564f3dd7c44e5b6bebdb</ctf>
<link rel='stylesheet' href='//cdn.jsdelivr.net/font-
hack/2.020/css/hack.min.css'>

</head>

<body>
  <h1>Easter Eggs Sender!</h1>
  <h4>Submit All of the 6 Flag hidden on the pages</h4>
  <input type="hidden" name="ee2"
value="CTF{26cff0571c90ee821fe50a106e2f4d73}">
  <form method="POST">
    <input type="text" name="flag" placeholder="Enter your flag"
required>
    <br>
    <button type="submit">Submit</button>
  </form>

  <script src="js/joker.js"></script>
</body>

</html>

```

Podemos observar varias cosas:

Una etiqueta (tag) `<ctf>` que contiene un hash que parece ser cifrado **md5**, muy parecido al que hemos encontrado en robots.txt.

Construimos la flag con `CTF{HASH}` y obtenemos la flag: `CTF{2613d5335779564f3dd}`

La mandamos al formulario y... Flag submitted successfully! Points: 100, hemos obtenido **+100 puntos** 🙌🙌.

### 📄 Easter Eggs

- 1/6 robots.txt
- 2/6 joker.php <ctf>hash</ctf>
- **Flags Restantes: 4**

Una etiqueta `<input>` oculta con el valor `CTF{26cff0571c90ee821fe50a106e2f4d73}`.

Enviamos esta flag al formulario y recibimos el mensaje: Flag submitted successfully!

Points: 100. Otros **+100 puntos** 🙌🙌.

## Easter Eggs

- 1/6 `robots.txt`
- 2/6 `joker.php <ctf>hash<ctf>`
- 3/6 `joker.php <input type="hidden" value="ctf{hash}" />`
- **Flags Restantes: 3**

## Exploración de `css/joker.css`

Aunque es inusual encontrar flags en archivos CSS, decidimos revisar el contenido del archivo `joker.css`, encontrando lo siguiente:

```
body {
  font-family: Arial, sans-serif;
  background-color: #000;
  color: #fff;
  text-align: center;
  margin: 0;
  padding: 0;
}

h1 {
  margin-top: 20px;
  font-size: 2rem;
  color: #0f0;
}

form {
  margin-top: 30px;
}

input[type="text"] {
  padding: 10px;
  font-size: 1rem;
  width: 80%;
  max-width: 400px;
  margin-bottom: 10px;
}

CTF{
  8fcc4f47a8f0261fb3d93fc62af26038
}

button {
  padding: 10px 20px;
  font-size: 1rem;
}
```

```

background-color: #0f0;
color: #000;
border: none;
cursor: pointer;
}

CTF{
  visibility: hidden;
}

button:hover {
  background-color: #0c0;
}

p.message {
  margin-top: 20px;
  font-size: 1.2rem;
  color: #ff0;
}

```

La primera estructura contiene una flag: CTF{8fcc4f47a8f0261fb3d93fc62af26038} . La enviamos al formulario y recibimos el mensaje: Flag submitted successfully! Points: 100 . Otros **+100 puntos** 🙌🙌 .

### 📄 Easter Eggs

- 1/6 robots.txt
- 2/6 joker.php <ctf>hash<ctf>
- 3/6 joker.php <input type="hidden" value="ctf{hash}" />
- 4/6 css/joker.css CTF{hash}
- **Flags Restantes: 2**

## Exploración de js/joker.js

### js/joker.js flag1

Una vez accedido al archivo podremos visualizar un código parecido a este:

```

// Create the js xmlhttprequest to send the flag to the server using js and
get the response
document.querySelector("form").addEventListener("submit", function (event) {
  event.preventDefault();
  var flag = document.querySelector('input[name="flag"]').value;
  var xhr = new XMLHttpRequest();
  xhr.open("POST", "joker.php", true);
  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

```

```

xhr.onreadystatechange = function () {
  if (xhr.readyState === 4 && xhr.status === 200) {
    var response = JSON.parse(xhr.responseText);
    if (response.success) {
      alert("Flag submitted successfully! Points: " + response.points);
    } else {
      alert("Error: " + response.message);
    }
  }
};
xhr.send("flag=" + encodeURIComponent(flag));
});
function huybpil_upalinkaasdwb(gogor) {

  let intermediate = gogor;
  let tempStorage = null;

  if (typeof gogor === 'string' || typeof gogor === 'number' ||
Array.isArray(gogor)) {
    tempStorage = JSON.parse(JSON.stringify(intermediate));
  }

  if (typeof tempStorage === 'string') {
    tempStorage = tempStorage.split('').join('');
  } else if (typeof tempStorage === 'number') {
    tempStorage = tempStorage + 0 - 0;
  } else if (Array.isArray(tempStorage)) {
    tempStorage = tempStorage.map(item => item);
  }

  function noOpTransform(value) {
    const dummy = value;
    return dummy;
  }
  const processed = noOpTransform(tempStorage);

  let finalOutput;
  if (Array.isArray(processed)) {
    finalOutput = [ ... processed];
  } else {
    finalOutput = processed;
  }

  return typeof finalOutput === 'string'
    ? finalOutput.split('').reverse().reverse().join('')

```

```

        : finalOutput;
    }

    // credential managing
    function _0x14f3(_0x262332, _0x27cdd5) {
        var _0x14f307 = _0x27cd();
        _0x14f3 = function (_0x315e01, _0x248184) {
            _0x315e01 = _0x315e01 - 0x1a2;
            var _0x566a71 = _0x14f307[_0x315e01];
            return _0x566a71;
        };
        return _0x14f3(_0x262332, _0x27cdd5);
    }
    function _0x27cd() {
        var _0x41a9f1 = ["flag{", "6e1ab", "43adc8", "a9faab", "8491a", "307b7"];
        _0x27cd = function () {
            return _0x41a9f1;
        };
        return _0x27cd();
    }
    var _0x55468e = _0x14f3;
    var flag =
        _0x55468e(0x1a2) +
        "22734" +
        _0x55468e(0x1a3) +
        _0x55468e(0x1a4) +
        _0x55468e(0x1a5) +
        _0x55468e(0x1a6) +
        _0x55468e(0x1a7) +
        "}";

    // last flag
    // çœ<ä¼¼æœ%ç””çš„ăšÿèf½
    function calculateMetrics(data) {
        const processed = data.map((item) => item * Math.random());
        return processed.reduce((sum, value) => sum + value, 0);
    }

    function dedede(bobore) {
        const temp1 = bobore.map((num) => num + 0);
        const temp2 = temp1.filter((num) => num % 2 === 0 || num % 2 !== 0);

        let processed = [];
        for (let i = 0; i < temp2.length; i++) {
            processed.push(temp2[i] - 0);
        }

        const step1 = [];
    }

```





```

const bumburi = [
  104, 116, 116, 112, 115, 58, 47, 47, 97, 112, 112, 46, 97, 110, 121, 46,
  114, 117, 110, 47, 116, 97, 115, 107, 115, 47, 100, 55, 52, 49, 100, 50,
  52,
  49, 45, 100, 51, 98, 101, 45, 52, 55, 48, 53, 45, 57, 98, 102, 99, 45,
  101,
  50, 55, 101, 53, 56, 52, 57, 50, 55, 100, 50,
];
return dedede(bumburi);
}

// Pli utilaj aspektoj
function processAnalytics(events) {
  const result = events
    .filter((event) => event.type === "click")
    .map((event) => ({
      id: event.id,
      timestamp: event.timestamp,
    }));
  console.log("Processed analytics:", result);
  return result;
}

function updateCache(key, value) {
  const cache = {};
  cache[key] = value;
  console.log("Cache updated:", key);
  return cache;
}

```

La primera parte del codigo:

```

// Create the js xmlhttprequest to send the flag to the server using js and
get the response
document.querySelector("form").addEventListener("submit", function (event) {
  event.preventDefault();
  var flag = document.querySelector('input[name="flag"]').value;
  var xhr = new XMLHttpRequest();
  xhr.open("POST", "joker.php", true);
  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
  xhr.onreadystatechange = function () {
    if (xhr.readyState === 4 && xhr.status === 200) {
      var response = JSON.parse(xhr.responseText);
      if (response.success) {
        alert("Flag submitted successfully! Points: " + response.points);
      } else {
        alert("Error: " + response.message);
      }
    }
  }
}

```

```
};  
xhr.send("flag=" + encodeURIComponent(flag));  
});
```

Sirve para mandar a validar la flag enviada desde el formulario.

Posteriormente encontramos este fragmento de código donde encontramos que parece estar diseñado para perder tiempo entendiéndolo y otra parte ofuscada. Sin embargo podemos tratar de entender por encima lo que podría hacer para descartar funciones y/o código de relleno.

```
function huybpil_upalinkaasdwb(gogor) {  
  
    let intermediate = gogor;  
    let tempStorage = null;  
  
    if (typeof gogor === 'string' || typeof gogor === 'number' ||  
    Array.isArray(gogor)) {  
        tempStorage = JSON.parse(JSON.stringify(intermediate));  
    }  
  
    if (typeof tempStorage === 'string') {  
        tempStorage = tempStorage.split('').join('');  
    } else if (typeof tempStorage === 'number') {  
        tempStorage = tempStorage + 0 - 0;  
    } else if (Array.isArray(tempStorage)) {  
        tempStorage = tempStorage.map(item => item);  
    }  
  
    function noOpTransform(value) {  
        const dummy = value;  
        return dummy;  
    }  
    const processed = noOpTransform(tempStorage);  
  
    let finalOutput;  
    if (Array.isArray(processed)) {  
        finalOutput = [ ... processed];  
    } else {  
        finalOutput = processed;  
    }  
  
    return typeof finalOutput === 'string'
```

```

        ? finalOutput.split('').reverse().reverse().join('')
        : finalOutput;
    }

    // credential managing
    function _0x14f3(_0x262332, _0x27cdd5) {
        var _0x14f307 = _0x27cd();
        _0x14f3 = function (_0x315e01, _0x248184) {
            _0x315e01 = _0x315e01 - 0x1a2;
            var _0x566a71 = _0x14f307[_0x315e01];
            return _0x566a71;
        };
        return _0x14f3(_0x262332, _0x27cdd5);
    }
    function _0x27cd() {
        var _0x41a9f1 = ["flag{", "6e1ab", "43adc8", "a9faab", "8491a", "307b7"];
        _0x27cd = function () {
            return _0x41a9f1;
        };
        return _0x27cd();
    }
    var _0x55468e = _0x14f3;
    var flag =
        _0x55468e(0x1a2) +
        "22734" +
        _0x55468e(0x1a3) +
        _0x55468e(0x1a4) +
        _0x55468e(0x1a5) +
        _0x55468e(0x1a6) +
        _0x55468e(0x1a7) +
        "}";

    // last flag
    // çœ<ä¼¼æœ%ç””çšŸ,ăšŸëf½
    function calculateMetrics(data) {
        const processed = data.map((item) => item * Math.random());
        return processed.reduce((sum, value) => sum + value, 0);
    }

    function dedede(bobore) {
        const temp1 = bobore.map((num) => num + 0);
        const temp2 = temp1.filter((num) => num % 2 === 0 || num % 2 !== 0);

        let processed = [];
        for (let i = 0; i < temp2.length; i++) {
            processed.push(temp2[i] - 0);
        }
    }

```

```

const step1 = [];
for (let i = processed.length - 1; i >= 0; i--) {
  step1.push(processed[i]);
}
const step2 = [];
for (let i = step1.length - 1; i >= 0; i--) {
  step2.push(step1[i]);
}

const mapped = step2.map((value, index) => {
  const adjustment = index % 2 === 0 ? value + 1 : value - 1;
  return adjustment - (index % 2 === 0 ? 1 : -1);
});

const buffer = [];
mapped.forEach((charCode, idx) => {
  if (idx % 2 === 0) {
    buffer.push(charCode);
  } else {
    buffer.push(charCode + 0);
  }
});

const finalString = buffer.reduce((acc, curr) => {
  const char = String.fromCharCode(curr);
  return acc.concat(char);
}, "");

const result = finalString.split("").reverse().reverse().join("");
return huybpil_upalinkaasdwb(result);
}

```

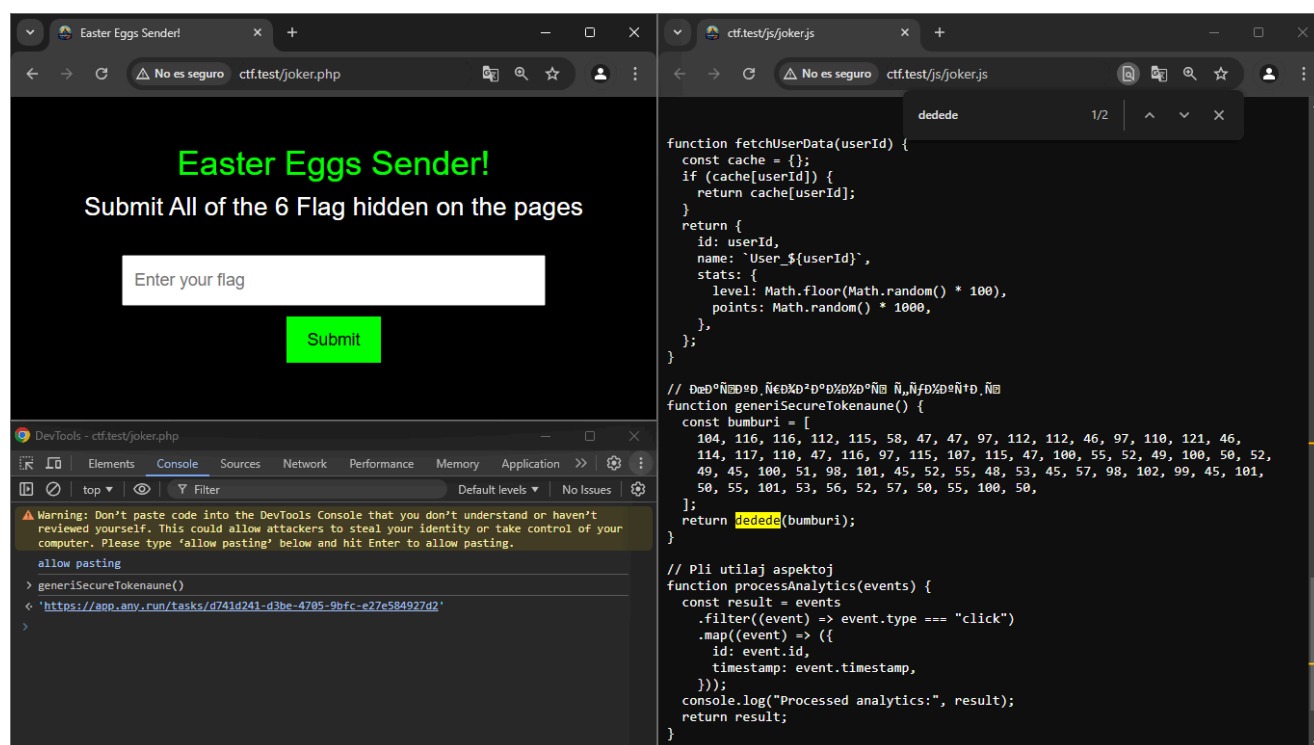
```

function fetchUserData(userId) {
  const cache = {};
  if (cache[userId]) {
    return cache[userId];
  }
  return {
    id: userId,
    name: `User_${userId}`,
    stats: {
      level: Math.floor(Math.random() * 100),
      points: Math.random() * 1000,
    },
  };
}

```

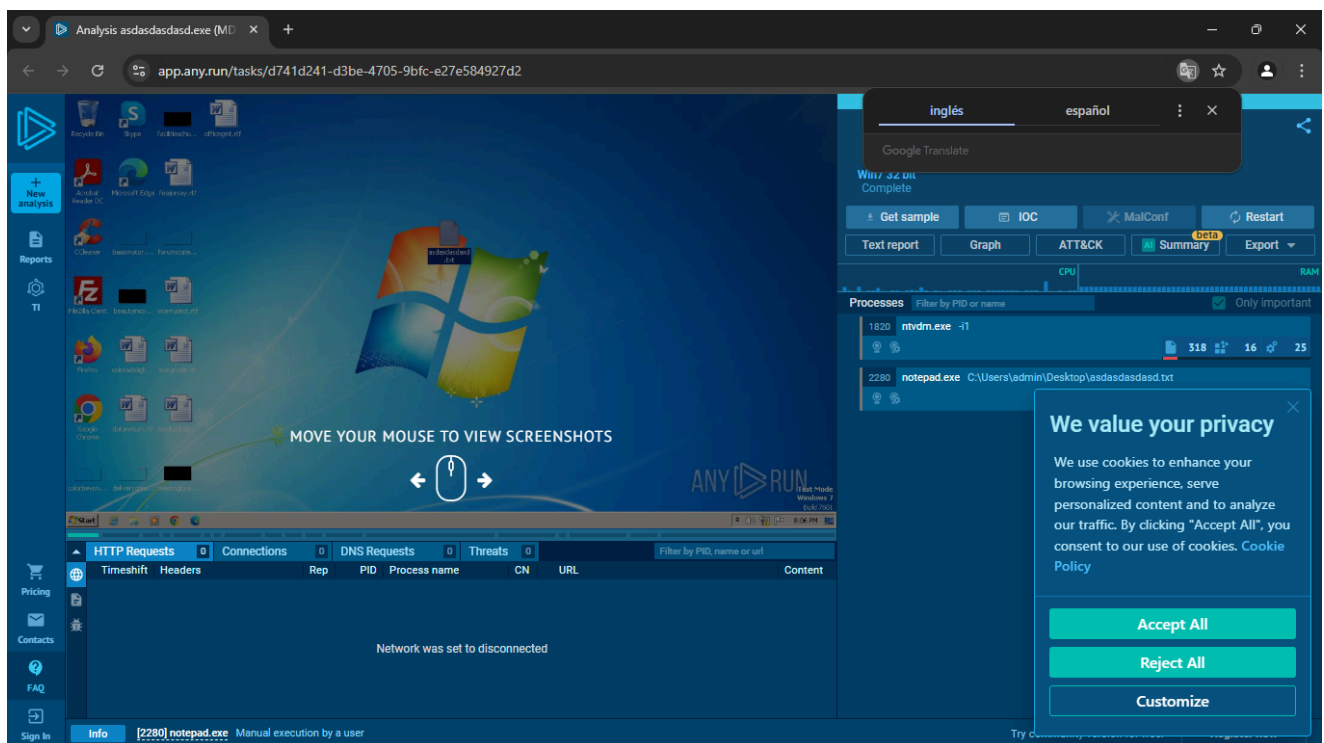
La función `huybpil_upalinkaasduwb(gogor)` vemos que tambien se llama más abajo en la función `dedede(bobore)` y que la función `dedede(bobore)` se llama en la función `generiSecureTokenane()`, así pues las funciones `huybpil_upalinkaasduwb(gogor)` y `dedede(bobore)` les entran un parametro y devuelven otro, mientras que `genericSecureTokenane()` contiene los datos que manda. así pues podriamos probar de llamar a la función ya que esta no se llama en ningun momento para así intentar visualizar lo que realiza.

Vamos a acceder a la consola del navegador presionando `Ctrl + Shift + I` o bien haciendo clic derecho en cualquier parte de la pagina y haciendo clic en "Inspeccionar Elemento", posteriormente presionaremos en la pestaña llamada "Console" y escribiremos `allow pasting` (esta parte es apra que nos deje realizar copy paste en la propia consola, ya que algunos navegadores como chrome han añadido esta pequeña medida de protección), ahora que ya tenemos activado el copy & paste pegaremos el nombre de la funcion `generiSecureTokenane()` y presionaremos enter.



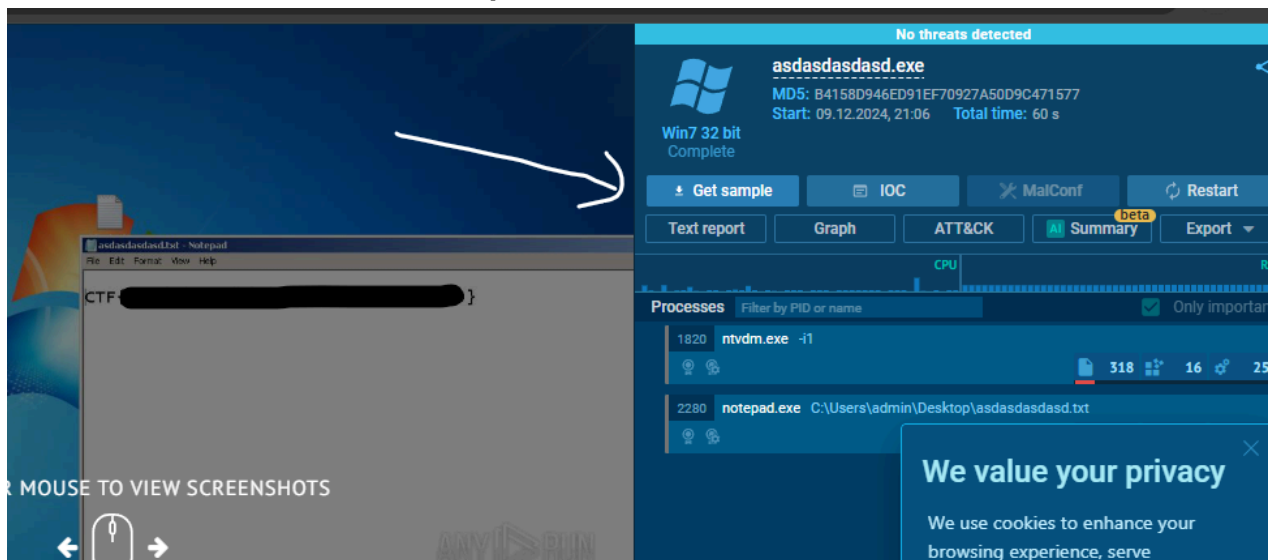
Devuelve un enlace a la web [any.run](https://app.any.run). Buscamos por internet lo que es [any.run](https://app.any.run) y encontramos que "ANY. RUN es una **herramienta de detección, seguimiento e investigación de cibermenazas en tiempo real**. El sandbox interactivo en línea es una solución perfecta para agilizar su análisis." Esta web solo permite el registro de cuentas corporativas, no admite mails genericos.

Accedemos a <https://app.any.run/tasks/d741d241-d3be-4705-9bfc-e27e584927d2> y encontramos lo siguiente:



Nos dice que movamos el ratón para ver las capturas de pantalla. Al mover el ratón visualizamos como en un Windows 7 se ejecuta un .exe que da error, despues lo que realiza el usuario es renombrar el nombre del archivo cambiando la extension .exe a extension .txt posteriormente procede a abrirlo y encuentra una linea muy larga de 0 hasta llegar al final donde se encuentra la estructura de flag CTF{hash} que hemos podido visualizar. Para extraer la flag podemos realizar lo siguiente:

- Si se tiene cuenta empresarial: registrarse en la plataforma y en el enlace del sample, hacer clic en el boton de **Get sample**



Se nos descargara el archivo ejecutable y podremos realizar los mismos pasos que sanen en el análisis en sandbox de any.run.

- Si no tenemos cuenta empresarial: podemos copiar manualmente la flag a mano o usando un conversor de imagen a texto (OCR) , aunque estos no son muy recomendables ya que muchas veces pueden fallar en algún carácter.

Así pues hemos obtenido la flag CTF{1aa6d80a7ff6df5c57dad90ed5c37a09}

La mandamos al formulario y... Flag submitted successfully! Points: 150 , hemos obtenido otros **+150 puntos** 🙌🙌.

### 📄 Easter Eggs

- 1/6 robots.txt
- 2/6 joker.php <ctf>hash<ctf>
- 3/6 joker.php <input type="hidden" value="ctf{hash}" />
- 4/6 css/joker.css CTF{\n\thash\n}
- 5/6 `js/joker.js generiSecureTokenane(); any.run
- **Flags Restantes: 1**

## js/joker.js flag2

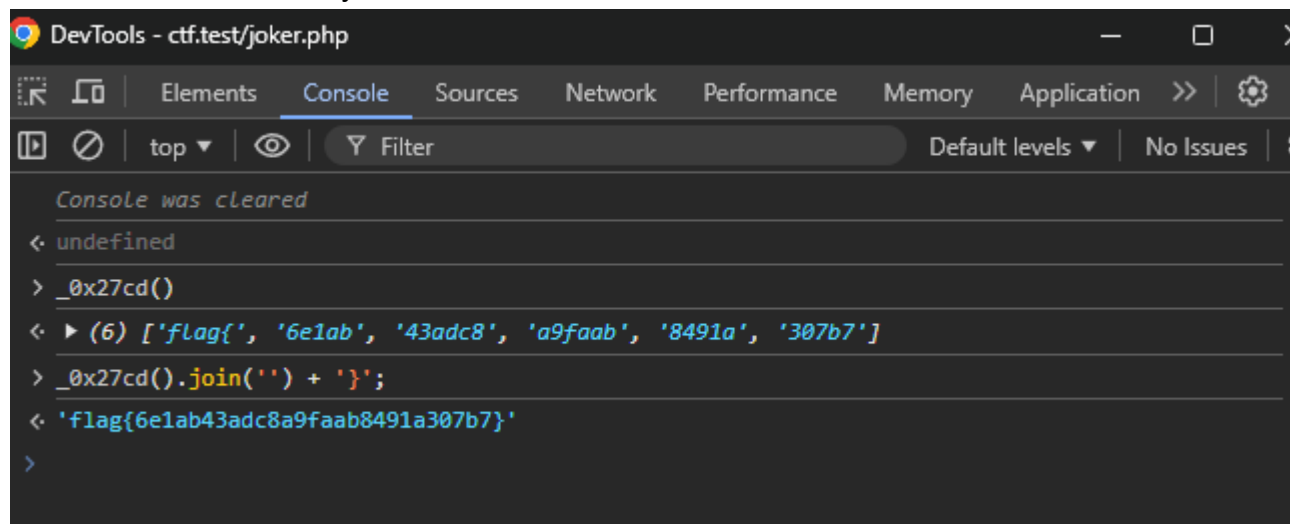
Continuamos analizando el código y nos centramos en la parte donde pone *credential managing*:

```
// credential managing
function _0x14f3(_0x262332, _0x27cdd5) {
  var _0x14f307 = _0x27cd();
  _0x14f3 = function (_0x315e01, _0x248184) {
    _0x315e01 = _0x315e01 - 0x1a2;
    var _0x566a71 = _0x14f307[_0x315e01];
    return _0x566a71;
  };
  return _0x14f3(_0x262332, _0x27cdd5);
}
function _0x27cd() {
  var _0x41a9f1 = ["flag{", "6e1ab", "43adc8", "a9faab", "8491a", "307b7"];
  _0x27cd = function () {
    return _0x41a9f1;
  };
  return _0x27cd();
}
var _0x55468e = _0x14f3;
var flag =
  _0x55468e(0x1a2) +
  "22734" +
  _0x55468e(0x1a3) +
  _0x55468e(0x1a4) +
  _0x55468e(0x1a5) +
  _0x55468e(0x1a6) +
  _0x55468e(0x1a7) +
  "}";
```



Vemos que hay dos sitios donde coincide una búsqueda de "flag", donde encontramos un array que se llama en la función pero se puede visualizar una estructura de la flag y otra que parece verse ofuscada.

en la función del primer resultado: `_0x27cd` realizamos una llamada por terminal y nos devuelve el mismo array:



```
DevTools - ctf.test/joker.php
Elements Console Sources Network Performance Memory Application >> | ⚙️
top | Filter
Console was cleared
< undefined
> _0x27cd()
< ▶ (6) ['flag{', '6e1ab', '43adc8', 'a9faab', '8491a', '307b7']
> _0x27cd().join('') + '{}';
< 'flag{6e1ab43adc8a9faab8491a307b7}'
>
```

Ha sido demasiado facil? Podria tratarse de un rabbit hole?

### 🔍 ¿Qué es un rabbit hole?

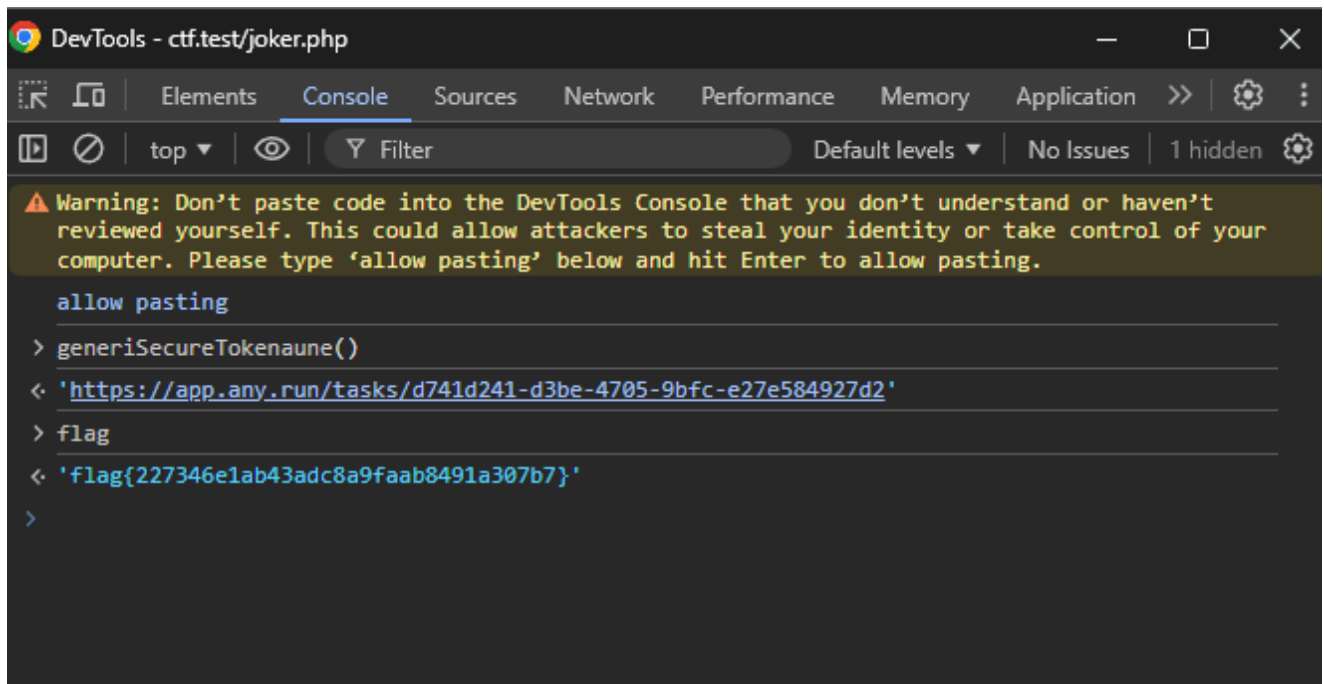
Serie de pistas, distracciones o rutas aparentemente relevantes en un entorno de ciberseguridad (como CTFs o investigaciones forenses) que pueden desviar al analista o participante de su objetivo principal. *En ocasiones, estas distracciones pueden incluir código ofuscado, cadenas falsas o caminos sin salida.*

Mandamos la Flag y... Error: Invalid flag . ¿Entonces podriamos decir que si se trata de un rabbit hole?

### 🔗 Easter Eggs

- 1/6 `robots.txt`
- 2/6 `joker.php <ctf>hash<ctf>`
- 3/6 `joker.php <input type="hidden" value="ctf{hash}" />`
- 4/6 `css/joker.css CTF{\n\thash\n}`
- 5/6 `js/joker.js generiSecureTokenane(); any.run`
- 6/6 `js/joker.js ofuscated code ['flag{','hash']`
- **Flags Restantes: 1**

Sin embargo visualizamos que en el código se declara una variable flag, la llamamos desde consola y nos devuelve el resultado de la flag:



DevTools - ctf.test/joker.php

Warning: Don't paste code into the DevTools Console that you don't understand or haven't reviewed yourself. This could allow attackers to steal your identity or take control of your computer. Please type 'allow pasting' below and hit Enter to allow pasting.

```
allow pasting
> generiSecureTokenane()
< 'https://app.any.run/tasks/d741d241-d3be-4705-9bfc-e27e584927d2'
> flag
< 'flag{227346e1ab43adc8a9faab8491a307b7}'
>
```

La mandamos al formulario y... Flag submitted successfully! Points: 100 , hemos obtenido otros **+100 puntos** 🙌🙌.

### ✓ Easter Eggs

- 1/6 robots.txt
- 2/6 joker.php <ctf>hash<ctf>
- 3/6 joker.php <input type="hidden" value="ctf{hash}" />
- 4/6 css/joker.css CTF{\n\thash\n}
- 5/6 js/joker.js generiSecureTokenane(); any.run
- 6/6 js/joker.js var flag
- **Flags Restantes: 0 - Completed!**

## Califica el Reto

Choose your browser from the list: