

DragonJAR CTF 2016 @ 17. ago. 2016

by @NoxOner @javierprtd @alguien_tw && @MaranonD

• [DragonJAR Security Conference 2016 – CTF Writeup]•

<u>. Web 100 – Web hacking .</u>

. Descripción .



. Análisis .

Al ingresar a la URL del reto se nos muestra un mensaje de error con código "400 Bad Request". Examinando el código fuente de la página de error observamos un comentario HTML muy interesante:

```
<html>
<body>
<h1>BAD Request</h1>
Your browser sent a request that this server could not
understand
<!-- no provienes de http://www.cia.gov/cyberterrorism/1q2w.php
--><!-- no usas MacOSX --><!-- no usas Napster -->
If you think is a error server, please contact the
webmaster
<h2>Error 400</h2></body><html>
```

Se trata de un reto clásico en el cual debemos engañar a la aplicación web modificando las cabeceras Referer y User-Agent de nuestra solicitud HTTP.



by @NoxOner @javierprtd @alguien_tw && @MaranonD



. Solución .

Usamos el comando "curl" para enviar el request asignando como Referer la URL "http://www.cia.gov/cyberterrorism/1q2w.php" y como User-Agent el valor "Mozilla/5.0 (MacOSX Napster)".

```
# curl -A "Mozilla/5.0 (MacOSX Napster)" -H 'Referer:
http://www.cia.gov/cyberterrorism/1q2w.php' http://game-
ehacking.rhcloud.com/
```



El flag es: d0f5c7596f569f72377a47038d98565e76195d1d

. Web 150 – Tipografia .

. Descripción .

	Tipografia
	(150 points, resuelto por 6)
Descripción: http://wrg-	pleni.rhcloud.com/wtf1.html probemos tus habilidades.
Bandera correcta	

. Análisis .

La web nos muestra un cuadro de texto y nos pide ingresar la frase "Web Design is 95% Typography". Sin embargo, no es tan simple como aparenta. Al escribir en el cuadro de texto observamos que ciertos caracteres son reemplazados por otros.



Write in the text field the following sentence:

Web Design is 95% TypographyW*b D*sIgn Is **5 T*p*gruph*

Podemos suponer que las sustituciones se hacen con Javascript. Sin embargo, luego de examinar el código Javascript, concluimos que no es así. Además observamos que el formulario envía el texto al servidor tal cual ha sido ingresado, sin realizar ninguna modificación.

```
[...]
$.ajax({
    url: "wtf1",
    type: "POST",
    data: JSON.stringify({
        f: $('input[name="f"]').val()
    }),
    contentType: "application/json; charset=utf-8",
    dataType: "json",
    success: function(a) {
[...]
```

Al revisar el código CSS observamos que el cuadro de texto emplea una fuente personalizada la cual se encuentra en el archivo "wtf1.ttf".

```
[...]
@font-face {
    font-family: 'wtfl';
    src: url('wtfl.ttf') format('truetype');
    font-weight: normal;
    font-style: normal;
}
[...]
.center input[type='text'] {
    font-family: 'wtfl';
    font-size: 2.8 rem;
[...]
```

El fichero "wtf1.ttf" contiene una fuente llamada "Source Sans Pro" que ha sido generada, aparentemente, a partir de la fuente "Trebuchet MS" intercambiando,





 \leftarrow

reemplazando o moviendo los glifos de ciertos caracteres. Por ello, al escribir observamos glifos que no se corresponden con los caracteres pulsados en el teclado.

. Solución .

El reto consiste en encontrar los nuevos caracteres asociados al glifo correcto para que el texto se vea tal cual nos lo indican. Empleamos "charmap" (Windows) o "gucharmap" (Linux/Gnome) para analizar el fichero "wtf1.ttf" y buscar manualmente los caracteres.

Source Sans Pro	•	Negrita	Cursiva	22	- +					
Bloque Unicode Ciritico (supteme	Tabla	de caracte	eres	Detalles	del carácte	۲				
Armenio Hebreo	÷	•	•	:	:		•	·	÷	•
Árabe	÷	÷	``	V)	J	-	、
Sirio	•		-	У			<u>۲</u>			÷
Árabe (suplemen Thaana	¥	Ę	د	σ	ο	,	~~	7	7	<u>د</u>
N'Ko Samaritano Mandeo	ர்	5	2	72	`	Δ_	92	لد	٦	P
Árabe extendido-A Devanagari	Ľ	.a	Ē	т	۵	ک	ŕ	ċ	Ŷ	শ
Bengalí Gurmukhi		<i>'ı</i>	11	•	m	4	11		I	Ŧ
Texto para copiar:										Copiar
U+070D SYRIAC H	ARKLEAN	ASTERISC	US • mar	ks the beq	inning of a	phrase, wo	ord, or mor	pheme tha	t has a mar	rginal note

El caracter más difícil de encontrar fue la "y". Este se encontraba en el bloque unicode correspondiente al alfabeto Sirio. Luego de mucho tiempo y un gran esfuerzo grupal logramos reunir todos los caracteres faltantes.

Wêb DêsIgn Is)%‰ T**#**pôgrâph#

Write in the text field the following sentence:

Web Design is 95% Typography

Web Design is 95% Typography



amnesia

Luego de enviar el formulario obtenemos el flag.

Congratulations: bc9c21e45d19fd8a33581a130558a1f3

El flag es: bc9c21e45d19fd8a33581a130558a1f3

<u>. Web 200 – Crazy links .</u>

. Descripción .

		~~~	~ ~		inl	~		
			aZ	уı	IUK	S		
	(2	200 po	ints, i	resue	lto poi	r 14)		
escripción: http	://wrg-pleni.	rhclou	ud.co	m/wt	f3.htn	nl		
escripción: http	://wrg-pleni.	rhclou	ud.co	m/wt	f3.htn	nl		
e <b>scripción:</b> http Bandera corre	<b>:://wrg-pleni.</b> cta	rhclou	ud.co	m/wt	f3.htn	nl		
e <b>scripción:</b> http Bandera corre	<b>:://wrg-pleni.</b> tta	rhclou	ud.co Er	m/wt	f3.htn	nl		

#### . Análisis .

La web del reto nos muestra una página con 1024 enlaces cada uno de los cuales envía un número distinto al servidor. El número está comprendido entre 0 y 1023. Solo hay un enlace que envía el número correcto y tenemos 10 intentos para encontrarlo.

∰ <b>₩</b> ₩	TF~3 🏠		×								
$\leftrightarrow$ $\Rightarrow$ C	(i) wrg	-pleni.rhcl	oud.com/v	vtf3.html					🗟 🕁	۱ 🔶	:
****	****	****	*****	****	****	****	*****	****	****	****	1
****	****	****	****	*****	****	****	*****	*****	****	****	- 1
****	****	****	****	****	*****	****	****	*****	*****	****	- 1
****	****	****	****	****	****	****	****	****	****	****	- 1
****	****	****	****	****	****	*****	*****	****	****	****	
*****	****	****	****	****	****	****	*****	*****	*****	*****	
*****	*****	****	****	*****	*****	*****	****	*****	*****	*****	
*****	*****	*****	****	*****	*****	*****	****	****	*****	*****	
*****	****	*****	*****	*****	*****	*****	*****	*****	*****	*****	
*****	****	****	*****	*****	*****	*****	*****	*****	*****	****	
*****	*****	*****	*****	*****	*****	****	****	*****	*****	*****	
*****	*****	*****	****	*****	*****	*****	****	****	*****	*****	
****	****	*****	*****	****	****	*****	*****	*****	*****	*****	
*****	****	****	*****	*****	****	****	****	****	****	****	
*****	*****	****	****	*****	*****	****	*****	*****	*****	****	
****	****	****	****	****	*****	*****	****	****	*****	*****	
****	*****	*****	*****	****	****	****	****	****	****	*****	,





En el código Javascript observamos que existe un array "f" el cual contiene 10 caracteres diferentes, un contador "g" inicialmente en 0 y una variable "e" a la cual se le asigna un valor extraído de la cookie. Al hacer clic en los enlaces, también se envía al servidor un objeto JSON con el resultado de concatenar el caracter del array "f" que se encuentra en la posición "g" y el valor de la variable "e".

```
[...]
        e = "",
        \mathbf{f} = ["!", "{", "}", "[", "]", "-", "*", "(", ")", "^"],
        g = 0;
    $.get("wtf3", function(a, b, c)
        e = /^.*wtf3=([0-9]{1,10}).*$/.exec(document.cookie)[1]
    }):
    for (var h = Math.floor(Math.random() * a.length), i = 0; i < Math.pow(2, c);</pre>
i++) {
        var j = a[h],
             k = b(i, 10, 4, !1, j),
             l = [j[0], j[0], j[0], j[0], j[0]].join(""),
m = l.substring(0, c / 2 - k.length) + k,
             n = Math.floor(360 * Math.random()),
             o = $('<a href="/' + d + "?n=" + i + '" style="color:hsl(' + n +</pre>
',75%,25%)">' + m + "</a>");
        o.click(function(a) {
             a.preventDefault(), $(this).addClass("wrong"), $.ajax({
                 url: $(this).attr("href"),
                 type: "POST",
                 data: JSON.stringify({
                     m: f[g] + e
                 }),
                 contentType: "application/json; charset=utf-8",
                 dataType: "json",
                 success: function(a)
                     switch (a.msg) {
                          case "yep":
                              $(".wrapper").html(a.html);
                              break:
                          default:
g++, $(".msg").remove(), $(".wrapper").prepend('<div
class="msg">Server not convinced. You still have ' + (10 - g) + '
attempts<div class="close">X</div>'), $(".close").click(function() {
                                  $(".msg").remove()
                              }), g >= 10 && location.reload()
                      }
[...]
```

En efecto, al hacer clic sobre un enlace, observamos que se envía la siguiente solicitud HTTP.

```
POST /wtf3?n=8 HTTP/1.1
Host: wrg-pleni.rhcloud.com
User-Agent: Mozilla/5.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json; charset=utf-8
X-Requested-With: XMLHttpRequest
Referer: http://wrg-pleni.rhcloud.com/wtf3.html
Content-Length: 17
Connection: close
```



```
by @NoxOner @javierprtd @alguien_tw && @MaranonD
```

Aparentemente los caracteres del array "f" actúan como un prefijo que le indica al servidor el número de intentos. Luego, si enviamos siempre el mismo prefijo podemos burlar el límite de diez intentos e ir enviando uno a uno los 1024 números (o los que hagan falta) hasta encontrar el correcto.

#### . Solución .

Empleamos la siguiente instrucción bash:

```
for i in `seq 0 1024`; do curl -H 'Content-Type:
application/json; charset=utf-8' --data '{"m":"}633839779"}'
"http://wrg-pleni.rhcloud.com/wtf3?n=$i" 2>/dev/null; done
```



El flag es: f88008dc63a67983e5824dafa0935662



# <u>. Crypto 50 – Juego de posiciones .</u>

#### . Descripción .

Juego de posiciones (50 points, resuelto por 39)
<b>Descripción:</b> Durante la época del imperio romano, el emperador Julio Cesar desarrollo un sistema de cifrado mono alfabético que aplicaba en sus comunicaciones con sus generales para que si los mensajes o el mensajero fuesen capturados por el enemigo, este no pudiera entender el mensaje original. Para pasar esta prueba y encontrar tu bandera deberás estudiar sobre el avance de estos métodos de cifrado mono alfabéticos. Mensaje: " utaxrxspsth bq ccrmv wk xmabvteatvdbgzvhglnemhkxl ". Confiamos en ti.
Bandera correcta
Enviar

#### . Análisis .

Este reto por la descripción estaba claro que se trataba de un cifrado cesar. Por lo que ingresamos en la web <u>http://www.xarg.org/tools/caesar-cipher/</u>, colocamos en el recuadro *"utaxrxspsth bq ccrmv wk xmabvteatvdbgzvhglnemhkxl"* y dando la opción a guess nos escupe:

#### Output:

baheyezwzao ix jjytc dr ethicalhackingconsultores

El flag es: ethicalhackingconsultores





# <u>. Crypto 100 - Telegram 2 .</u>

#### . Descripción .



#### . Análisis .

Luego de contactar con el bot por Telegram, activamos la opción "Envío de imágenes". El bot comenzó a enviarnos imágenes en forma de fichas de un rompecabezas cada 5 minutos. Además nos muestra el mensaje: "Necesitamos la contraseña del usuario Dragonjar". Tras reunir todas las fichas, armamos el rompecabezas.



Observamos el contenido de un fichero ".htpasswd" que contiene usuarios y hashes de contraseñas. Con mucho cuidado transcribimos el contenido de la imagen a un fichero de texto. La letra "l" (ele minúscula) y la "I" (i mayúscula) son muy similares, por ello ante la duda, duplicamos el registro y usamos ambos caracteres para garantizar que el hash correcto se encuentra en el archivo.

```
dragonjar:$apr1$w43XR408$6Pb8MIcIE/f33xB8cviZ3.
dragonjar:$apr1$w43XR408$6Pb8MIcIE/f33xB8cviZ3.
dragonjar:$apr1$w43XR408$6Pb8MIcIE/f33xB8cviZ3.
dragonjar:$apr1$w43XR408$6Pb8MIcIE/f33xB8cviZ3.
root:$apr1$wf3XR408$6Pb8MIcIE/f33xB8cviZ3.
root:$apr1$RofJBd7W$WaVaXfkcKiCMh6kqynYL9/
ehacking:$apr1$eba7obgQ$TImnSFGR8tYw1Tr1W2rpY1
```





#### . Solución.

Crackeamos los hashes usando John the Ripper y el diccionario de claves de Hashkiller¹.



La contraseña del usuario "dragonjar" es "admin123". Se la enviamos al bot por Telegram y nos entrega el flag.

	Completa el Puzzle - imagen 8/12	
	Necesitamos la contraseña del usuario Dragonjar	
	Alguien	10:20:46 AM
-	admin123	
Strengthe State	DragonJAR Conference	10:20:48 AM
and the new West	Puzzle Completado	
	Hola Agente Alguien Felicidades. Aqui tienes el hash:	
	c3d9ada4bad8bc08f9993ac5f852de8017842904	
	Alguien	10:21:33 AM
	📃 Menu Principal	

El flag es: c3d9ada4bad8bc08f9993ac5f852de8017842904

¹ HashKiller Passwords - http://home.btconnect.com/md5decrypter/hashkiller-dict.rar





# <u>. Crypto 150 – Telegram .</u>

#### . Descripción .

Este reto consta de dos partes, por ello vamos a separar el análisis y la solución para cada parte del reto.

#### . Análisis (Parte I) .

Solicitamos al bot por Telegram el reto "Reto 1 – Frases Celebres" y nos entrega el siguiente mensaje cifrado.

.Ixp zlkqoxpbñxp cyx mywy vk byzk sxdobsyb. Tu vakjky jkpgx wak tgjok rg bkg, uvsvj trdszricr ivxlcridvekv q fg ugehsjlajds ugf wpljsñgk VW4gZ3JhbiBjb25zZWpvIMK/UXVpZW4gZXMgZWwgYXV0b3I/

Claramente se diferencian dos partes en el mensaje. La primera parte parece un cifrado de sustitución simple, posiblemente César. La segunda parte es base64.

#### . Solución (Parte I).

Usamos la web "<u>https://www.nayuki.io/page/automatic-caesar-cipher-breaker-javascript</u>" para descifrar la primera parte del mensaje. Se trata de César, pero usando rotaciones diferentes para varios segmentos del texto. Luego de decodificar cada segmento obtenemos:

Las contraseñas son como la ropa interior. No puedes dejar que nadie la vea, debes cambiarla regularmente y no compartirla con extraños





Por otra parte, el texto en base64 dice **"Un gran consejo ¿Quien es el** *autor?*". Luego de buscar en Google nos enteramos que el autor de la frase es "Chris Pirillo". Se lo enviamos al bot y nos entrega la segunda parte del reto.

	Alguien Chris Pirillo	6:42:58 PM
- Consequence and	DragonJAR Conference Reto 1 - Frases Celebres Solucionado Parte I Felicidades Agente Alguien Solucionaste la parte I, era un gran consejo Ahora vamos por la segunda y ultima parte. Decodifica este texto y resuelve el acertijo:	6:42:59 PM
	eusonxqodpaxqeeoroeeecoxocuovxlerdaxsdcilxuprcoxumoedx	
	Estoy seguro que necesitaras una clave para resolverlo esta fue escondida dentro del tag <title> de la pagina xxx</title>	
eusonxqodpaxqeeo	roeeecoxocuovxlerdaxsdcilxuprcoxumoedx	

#### . Análisis (Parte II) .

Este cifrado nos mantuvo entretenidos mucho tiempo, de hecho este reto fue el último que resolvimos. Antes de dar con el algoritmo de cifrado correcto probamos muchos otros, entre ellos la Escítala, Vigenère y Alberti (por la pista de "razonamiento circular" de la descripción).

El bot nos dice que necesitamos una contraseña y que está en el título de la página el evento. Así que probablemente sea alguna de esas palabras.

```
<title>DragonJAR Security Conference - Congreso de Seguridad
Informática</title>
```

Una observación importante fue que el criptograma contiene caracteres "x" a intervalos regulares. Esto finalmente nos permitió identificar el algoritmo de cifrado.

euson<mark>x</mark> qodpa<mark>x</mark> qeeoroeeeco<mark>x</mark> ocuov<mark>x</mark> lerdax sdcilx uprco<mark>x</mark>





Se trata del algoritmo de cifrado de transposición por columnas². Este cifrado además nos permite saber la longitud de su contraseña. Para ello dividimos la longitud total del criptograma (54) entre la longitud de los segmentos que terminan en "x" (6). Es decir, la contraseña debe tener nueve caracteres. Luego las posibles contraseñas que aparecen en el título son "dragonjar" y "seguridad".

#### . Solución (Parte II) .

Para resolverlo programamos el siguiente script en python. La contraseña resultó ser "dragonjar".

```
#/usr/bin/env python
def transpo(texto, clave):
    cifra = ''
    clave len = len(clave)
    tabla = []
    for i in xrange(0, len(texto), clave len):
        fila = ''
        if i+clave_len < len(texto):</pre>
            fila = texto[i:i+clave len]
        else:
            fila = texto[i:] + 'x' * (clave len - len(texto[i:]))
        tabla.append([x for x in fila])
    clave ord = [x for x in clave]
    clave ord.sort()
    for c in clave ord:
        pos = clave.index(c)
        cifra += ''.join([fila[pos] for fila in tabla])
    return cifra
def des transpo(cripto, clave):
    texto = ''
    clave len = len(clave)
    cosi = len(cripto) / clave len
    tabla = [cripto[i:i+cosi] for i in xrange(0, len(cripto),
cosi)]
    clave ord = [x for x in clave]
    clave ord.sort()
    for i in xrange(0, cosi):
        posi = [] # parche para posiciones repetidas xD
        for c in clave:
            pos = clave ord.index(c)
            if c in posi:
                pos += 1
            posi.append(c)
            texto += tabla[pos][i]
```

² Algoritmos de transposición – Transposición por columnas. <u>http://redyseguridad.fi-p.unam.mx/proyectos/criptografia/criptografia/index.php/2-tecnicas-clasicas-de-cifrado/22-opereraciones-utilizadas/223-algoritmos-de-transposicion?showall=&start=3</u>

by @NoxOner @javierprtd @alguien_tw && @MaranonD



return texto

```
cifra = 'eusonxqodpaxqeeoroeeecoxocuovxlerdaxsdcilxuprcoxumoedx'
print des_transpo(cifra, 'dragonjar')
```

La salida es:

¿Qué es lo que puede comerse crudo o cocido pero no lavado? ¡Agua!

8	Alguien agua	6:43:28 PM
Margula Br	DragonJAR Conference Reto 1 - Frases Celebres Solucionado Parte II Felicidades Agente Alguien Solucionaste la parte II Toma tu premio:	6:43:30 PM
	883dd07be8b2a4e08a32d3c38d6a1c7fcfc88915	
	Tuesday, September 20, 2016	

El flag es: 883dd07be8b2a4e08a32d3c38d6a1c7fcfc88915





# <u>. Crypto 250 – Enigma .</u>

#### . Descripción .

Enigma (250 points, resuelto por 4)
<b>Descripción:</b> Enigma era el nombre de una máquina que disponía de un mecanismo de cifrado rotatorio, que permitía usarla tanto para cifrar como para descifrar mensajes. Su fama se debe a haber sido adoptada por las fuerzas militares de Alemania desde 1930. Su facilidad de manejo y supuesta inviolabilidad fueron las principales razones para su amplio uso. http://wrg- pleni.rhcloud.com/wtf2.html
Bandera correcta
Enviar
Bandera correcta Enviar

#### . Análisis .

El reto es bastante explícito ¡Se trata de la máquina Enigma! Observamos el código cifrado en la parte superior izquierda junto a un contador que nos da tan solo 10 segundos antes de que el código cambie. En la parte central están la rueda fija (en azul) y las tres ruedas móviles (en negro y naranja) que nos muestran la configuración inicial de la máquina sombreada con gris. La configuración inicial también cambia cada 10 segundos. Finalmente en la parte inferior derecha hay una caja de texto para enviar el código descifrado.





Tenemos todo lo que necesitamos para descifrar el código. Nuestro principal problema es el tiempo. Antes de que terminemos de ingresar la configuración inicial y el código cifrado en un simulador de Enigma los 10 segundos ya habrán terminado y todo habrá cambiado. Por ello, decidimos automatizar la resolución del reto con Javascript ;)

#### . Solución .

El primer problema a resolver será extraer la configuración inicial de la máquina, es decir, cuantas posiciones se ha desplazado cada rueda móvil con respecto a la rueda fija. Las ruedas son imágenes SVG a las cuales se les aplica una transformación de rotación usando CSS. Podemos calcular la configuración inicial a partir del número de grados de dicha rotación.







La siguiente función Javascript recibe como parámetro el número de rueda y devuelve su configuración inicial.

```
function getrot(n) {
    rot = (/rotate\((.+)deg/.exec( $('.r' + n)[0].style.transform
)[1]) * 1;
    pos = Math.round(rot / (360 / 26)) % 26;
    pos = (pos < 0) ? 26 + pos : pos;
    return 26 - pos;
}</pre>
```

El siguiente problema a resolver es conseguir una implementación sencilla de la máquina Enigma que podamos ejecutar desde la consola Javascript del navegador. Decidimos programar nuestra propia implementación de Enigma y para ello fue fundamental la explicación³ que da el usuario constructor91 de Youtube sobre el mecanismo de cifrado y descifrado de Enigma.

El resultado a continuación...

```
Rueda = function(alfa, rota) {
    this.alfa = alfa;
    this.rota = rota;
    this.rotar = function() {
        this.rota = (this.rota + 1) % this.alfa.length;
        return this.rota == 0;
    }
    this.valor = function(pos) {
        return this.alfa[(this.rota + pos) % this.alfa.length];
    }
    this.buscar = function (letra, indice) {
        for (var pos=0; pos < this.alfa.length; pos ++) {</pre>
            if (letra == this.alfa[pos][indice]) {
                pos = pos - this.rota;
                return (pos < 0) ? this.alfa.length + pos : pos;</pre>
            }
        }
        return null;
    }
    this.estado = function() {
        s = ''
        for(var i=0; i<this.alfa.length; i++) {</pre>
            s += this.alfa[(this.rota + i) % this.alfa.length] +
'|';
        console.log(s);
    }
}
```

3 Maquina Enigma de LU1DSS II https://www.youtube.com/watch?v=Wsdx9CniR78

```
Enigma = function (rot1, rot2, rot3) {
    this.alfa = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
    this.ruedas = [
        new Rueda(['AF', 'BG', 'CJ', 'DQ', 'EH', 'FN', 'GE',
'HM', 'ID', 'JX', 'KU', 'LA', 'MW', 'NL', 'OT', 'PV', 'QB', 'RY',
'QY', 'RO',
'SR', 'TL', 'UX', 'VC', 'WG', 'XK', 'YP', 'ZW'], rot2),
    new Rueda(['AS', 'BF', 'CN', 'DR', 'EV', 'FU', 'GO',
'HI', 'IC', 'JJ', 'KH', 'LZ', 'MM', 'NB', 'OP', 'PY', 'QA', 'RT',
'SL', 'TX', 'UD', 'VG', 'WQ', 'XE', 'YK', 'ZW'], rot3)
    ];
    this.rotar = function () {
        return this.ruedas[2].rotar() && this.ruedas[1].rotar()
&& this.ruedas[0].rotar();
    }
    this.cifrar letra = function(letra) {
        pos = this.alfa.indexOf(letra);
         for(var i=this.ruedas.length - 1; i>=0; i--) {
             letra = this.ruedas[i].valor(pos)[1];
             pos = this.ruedas[i].buscar(letra, 0);
         }
        this.rotar();
        return this.alfa[pos];
    }
    this.decifrar letra = function(letra) {
        pos = this.alfa.indexOf(letra);
         for(var i=0; i<this.ruedas.length; i++) {</pre>
             letra = this.ruedas[i].valor(pos)[0];
             pos = this.ruedas[i].buscar(letra, 1);
         }
        this.rotar();
        return this.alfa[pos];
    }
    this.cifrar = function (texto) {
        cripto = '';
         for(var i=0; i < texto.length; i++) {</pre>
             cripto += this.cifrar letra(texto[i]);
         }
        return cripto;
    }
    this.decifrar = function(cripto) {
         texto = '';
         for(var i=0; i < cripto.length; i++) {</pre>
             texto += this.decifrar letra(cripto[i]);
         }
        return texto;
    }
```

amnesia



}

```
by @NoxOner @javierprtd @alguien_tw && @MaranonD
```

```
function getrot(n) {
    rot = (/rotate\((.+)deg/.exec($('.r' + n)[0].style.transform
)[1]) * 1;
    pos = Math.round(rot / (360 / 26)) % 26;
    pos = (pos < 0) ? 26 + pos : pos;
    return 26 - pos;
}
rot1 = getrot(1);
rot2 = getrot(2);
rot3 = getrot(3);
cifra = $('.text').text();
e = new Enigma(rot1, rot2, rot3);
texto = e.cifrar(cifra);
console.log('texto: ' + texto);</pre>
```



*Nota:* La máquina enigma cifra en una dirección y descifra en la dirección contraria. Por alguna extraña razón terminamos implementando nuestra máquina Enigma en sentido contrario al del reto. En consecuencia tuvimos que usar el método "cifrar" para descifrar y viceversa xD

El código anterior lo copiamos, pegamos y ejecutamos en la consola Javascript del navegador. Nos imprime el código descifrado el cual ingresamos en la caja de texto para obtener el flag.



El flag es: 8698db3583504fcc23324529d3243bf8



amnesia



## <u>. Estego 80 – Logo DragonJarCon .</u>

#### . Descripción .



#### . Análisis .

Descargaba un fichero PNG dónde se encontraba el flag escondido. Ninguno de los que estábamos jugando éramos experto en esta categoría, nuestro nivel de estenografía se limitaba a abrirlo con el editor hexadecimal.

 3D50h:
 22 2D D9 3A
 6C DB 79 E9
 25 DE AB B5
 98 8C E2 95
 "-Ù:lÛyé\$P≪µ~Cấ•

 3D60h:
 FF 0B 66 8A
 91 C8 0C 3C
 C0 85 00 00
 00 0E 74 45
 ÿ.fŠ`È.<À....tE</td>

 3D70h:
 58 74 43 6F 6D 6D 65 6E 74 00 44 72
 34 67 30 6E
 XtComment.Dr4g0n

 3D80h:
 D9 8C BF 45 00 00 00 00 49 45 4E 44 AE 42 60 82
 ÙŒ¿E....IEND®B`,

Nuestra mente ágil y discernidora determinó rápidamente que la cadena de caracteres "Dr4g0n" era una contraseña (en realidad, después de tanto intentar fue deducido).

El equipo comenzó a probar muchas técnicas, desde el LSB (Least Significat Bit), invertir los colores hasta jugar con las curvas de colores. El segundero sin descanso nos presionaba con su sonido y luego de muchas pruebas un software fue probado, OpenStego era su nombre.

OpenStego tiene la opción de extraer datos, donde se puede especificar una contraseña, al ingresar "Dr4g0n" como tal, se generó un archivo que al hacer cat del mismo soltaba (no imprime chars nulos y badchars).

00000000: 5777 675a 6d78 794f 4446 694d 4745 314e WwgZmxyODFiMGE1Nz 00000010: 7a4e 6b4d 5755 304d 5749 304e 5530 4d57 zNkMWU0MWI0NU0MW





Para obtener las cadenas codificadas en base64 sin los *badchars* bastaba con ejecutar los siguientes comandos.

```
$ cat dragonJar | awk '{print $10}' > output.txt &&
  tail -n 1 dragonJar | awk '{print $6}' >> output.txt &&
  cat output.txt | tr -d "\n" > base64.txt &&
  sed -i 's/\./\n/g' base64.txt
```

Finalmente, para decodificarlos en la consola.

for i in `cat base64.txt`; do echo \$i | base64 -d; done

Una cadena codificada daba el flag.

El flag es: a6281b0a573d1e41b46a8fa3c932b2455cfca4b1001

# <u>. Estego 80 – Musica ligera .</u>

. Descripción .





#### by @NoxOner @javierprtd @alguien_tw && @MaranonD



#### . Análisis .

Al escuchar el archivo descargado, se identifican rápidamente tonos largos y cortos indicando de que nuestros oídos escuchan a morse, su legado. El análisis con espectrómetro nos daría los caracteres para la posterior codificación.



La cadena obtenida esta codifica en hexadecimal, y al hacerlo se obtenía el flag.

```
>>> '476f446674486552'.decode('hex')
'GoDftHeR'
```

El flag es: GoDftHeR

<u>. Misc 20 – Directorios .</u>

. Descripción .

	Directorios (20 points, resuelto por 41)	
Descripción: Encuent Adjunto: <u>6b4e7857cc</u>	tra el directorio con el archivo que contiene el flag. <u>07409067ce79b5a7eb91684.gz</u>	
Bandera correcta		
	Enviar	

#### . Análisis .

El reto nos presenta un archivo comprimido con "gzip" y en su interior otro empaquetado con "tar". El empaquetado resulta ser un directorio "search" y en su interior una infinidad de subdirectorios anidados. En alguno de estos hay un archivo con el flag y debemos encontrarlo.





#### . Solución .

Empleamos el comando "find" con el filtro "-type f" para buscar unicamente ficheros dentro del directorio "search".

nd ./search/ -type f	
<b>log:</b> root-node [10:25:36]     alguien ~ \$ file 6b4e7857c07409067ce79b5a7eb91684.gz     6b4e7857c07409067ce79b5a7eb91684.gz: gzip compressed data, las     2016, from Unix	t modified: Thu Aug 4 13:2
root-node [10:25:40] alguien ~ \$ gzip -d 6b4e7857c07409067ce79b5a7eb91684.gz	
<b>lot:</b> <b>alguien</b> ~ \$ file 6b4e7857c07409067ce79b5a7eb91684 6b4e7857c07409067ce79b5a7eb91684: POSIX tar archive (GNU)	
<b>root-node [10:25:58]</b> alguien ~ \$ mv 6b4e7857c07409067ce79b5a7eb91684 6b4e7857c07409	067ce79b5a7eb91684.tar
<b>leg root-node [10:26:18]</b> alguien ~ \$ tar xf 6b4e7857c07409067ce79b5a7eb91684.tar	
root-node [10:26:32] alguien ~ \$ find ./search/ -type f ./search/here5/here4/here9/here6/file.txt	
root-node [10:26:49] alguien ~ \$ cat ./search/here5/here4/here9/here6/file.txt The key is: 3811712de1bc48a2c36e4eab53847aae10e7a7d9	
root-node [10:26:55] alguien ~ \$	

El flag es: 3811712de1bc48a2c36e4eab53847aae10e7a7d9

<u>. Misc 20 – Unico .</u>

. Descripción .







#### . Análisis .

El archivo en si eran un montón de hashes donde solo uno era diferente. Bastaba con ejecutar en consola:

😣 🖨 🗊 noname@ubuntu: ~/Escritorio	
noname@ubuntu:~/Escritorio\$ cat pass.txt   sort   uniq 8150e384faa238703f7782b4cd64de11be6c3411 noname@ubuntu:~/Escritorio\$ _	-u

El flag era: 8150e384faa238703f7782b4cd64de11be6c3411

<u>. Misc 30 – I</u>	<u>Misc</u>	
Descripción		
	Misc (30 points, resuelto por 14)	
	Descripción: analiza y encuentra Adjunto: 4006493df4b1ab16e4cef1206fb3127d	
	Bandera correcta	
	Enviar	

#### . Análisis .

Con un simple vistazo con un editor de texto vemos que el archivo que se nos proporciona es un "dump" o extracción de un archivo desconocido analizado con un editor hexadecimal.

Podemos observar que los primeros bytes de dicho archivo comienzan por 42 5A 68 ó "BZh", característica propia de los archivos bz2.





Así pues con un editor hexadecimal creamos un archivo utilizando todos los bytes:

File	Edi	t :	Searc	:h	View	/ F	orm	at	Scrip	ots	Tem	plat	es	Tool	s '	Wind	wot	Help
	- 6	3.	•		P (		-	9		6	þ (	Ē	Ø	ଜା	111	, O	ÅВ	🖗 🔶 🕹 🗚 🔌 Hex 🛒 ۹
a.bz2	2 💌																	
Ŧ	Edit	As: H	lex 🔻	,	Run S	Script		Ru	n Ter	nolati	e 🔻							E
		0	1	2	3	4	5	6	7	8	9	A	в	С	D	E	F	0123456789ABCDEF
0000	h:	42	5A	68	39	31	41	59	26	53	59	5A	1D	B1	FF	00	00	BZh91AY&SYZ.±ÿ
0010	h:	4C	FF	FF	FF	6E	86	65	08	Α5	FD	5F	86	ЗB	46	EF	СВ	Lÿÿÿnte.¥ý t;FïË
0020	h:	24	DB	E5	A1	B8	9E	10	<b>E</b> 2	28	46	BF	52	80	1A	2D	C1	\$Ûåi,ž.â(F¿R€.−Á
0030	h:	CF	8F	AF	во	00	F9	34	22	Α7	AA	01	Α7	A8	0C	9A	1E	Ï. ^{−°} .ù4″§².§¨.š.
0040	h:	A6	8D	34	7Å	83	4D	00	34	34	34	OD	19	03	43	4D	03	¦.4zfM.444CM.
0050	h:	35	1E	AO	ЗD	43	D4	F5	OD	34	С4	1E	69	4F	14	6F	6A	5. =CÔö.4Ä.iO.oj
0060	h:	A1	55	34	CO	03	49	80	04	60	4C	4D	18	00	04	CO	98	;U4À.I€.`LMÀ~
0070	h:	04	СО	46	43	00	00	04	СО	08	68	60	6A	65	05	OF	51	.ÀFCÀ.h`jeQ
0080	h:	ΕÀ	7Å	6A	01	AO	OC	91	00	06	23	21	AO	32	0C	41	AO	êzjš#! 2.A
0090	h:	00	00	03	46	13	23	20	34	34	2 B	04	E7	32	68	ΕA	04	F.# 44+.ç2hê.
OOAO	h:	С8	2 E	81	7Å	60	51	82	41	11	10	OB	34	07	4A	DO	DO	Èz`Q,A4.JĐĐ
00В0	h:	86	02	8C	A4	2 A	89	14	СО	91	7C	41	32	53	СС	04	DA	. †.Œ¤*‱.À` A2SÍ.Ú
ooco	h:	<b>A</b> 6	2D	88	2 E	A8	30	E5	53	5E	С7	85	20	Α7	D4	30	BC	¦−^."OåS^Ç… §ÖO¥⊄
OODO	h:	55	AB	B5	D8	FΕ	62	АЗ	21	7D	9B	53	В4	98	94	С8	59	U«µØþb£!}>S´~″ÉY
OOEO	h:	2 D	45	Β7	07	9B	6A	47	F6	56	4C	48	DB	C7	Α2	41	24	-E .>jGöVLHUÇ¢A\$
OOFO	h:	DE	67	53	59	5B	ЗD	91	С8	AO	F9	79	2 E	OF	2 D	44	12	ÞgSY[=šE úyD.
0100	h:	4A	09	60	50	51	ED	OA	7E	2 A	E9	05	4A	AD	6E	56	ЗB	J. PQ1.~*é.J-nV;
0110	h:	10	OD	FA	4A	3B	OA	DO	9C	E5	30	88	90	EB	20	12	23	úJ;.ĐœãO^œë .#
0120	h:	DC	OC	54	BD	EA	39	50	AF	AO	74	68	4F	96	AO	EO	43	U.Z%29P thOaC
0130	h:	3F	68	09	B4	AF	19	DB	A4	3A DD	DU	B2	14	44	D6	FE	85	?h.' .U¤:Đ≚.DOp
0140	h:	DC	25	03	20	02	84	48	1E	BE	94	7E	02	DU	99	36	50	U%/H.%(~.D**6P
0150	h:	14	C9	97	36	A'7	87	80	91	D4	C7	24	6E	F6	EU	A5	86	.E-6§‡€`OÇ\$noa¥†
0160	h:	FD	85	CB	83	01	F8	19	78	59	98	80	88	91	9A AD	44	12	yµEf.ø.zY°Œ,`SD.
0170	h:	36	83	42	B7	E8	71	43	17	8A QE	80	1E	98	47	4B	F2	14	5;5°e.C.5€.>GKO.
0180	h:	71	60	47	6D DC	A6 EE	32	02	09	35	04	40	30	80	33	B1 FF	42	q Gm; 25A&U.3±B
0190	n:	r A	57	08	D8	rr	ЗB	89	42	90	28	48	2 D	UE	<u>р</u> 8	rr	80	uw.wy< ~~œ(n−.wy€
OIAU	n:																	

#### y lo abrimos utilizando por ejemplo 7zip.

EZ C:\Us	er De	sktop\a.bz2\;	a/					_	$\times$
Archivo	Editar Ver	Favoritos	Herramientas	Ayuda					
	-	$\checkmark$	•		×	i			
Agregar	Extraer	Proba	r Copiar	Mover	Borrar	Información			
ø 📙	C:\Users\	Desktop\a	.bz2\a\						~
Nombre					Tamaño	Tamaño comp	Modificado	Creado	Acces
file					190	187	2016-09-17 03:47		
<									>
0 elemento	o(s) seleccion	ado(s)							



Nuevamente, mirando las cabeceras del nuevo (1F 8B 08), vemos que es un archivo comprimido .gz asique lo volvemos a abrir con 7zip.

Esta operación se repite un par de veces más con archivos tipo .tar hasta que obtenemos, por fin, un archivo en el que se puede leer:

"the key is: 13ba34cafa454fd9c110b8d131cec4b6ffd7ec93"



