programa de BECAS
DGTIC
de FORMACIÓN en TIC

Seguridad Informática

Programación con Perl

Cervantes Varela Juan Manuel
Vallejo Fernández Rafael Alejandro

Data::Dumper

```perl
use Data::Dumper;
my $var = 35;
my @array = qw(contenido del arreglo);
my %hash = ( 'clave' => 'valor', 'dos' => 'segunda');
my @listas = (['a','b','c'], [1,2,3]);

print Dumper $var;
print Dumper @array;
print Dumper \@array;

print Dumper %hash;
print Dumper \%hash;

print Dumper @listas;
print Dumper \@listas;
```

Data::Dumper

```
kali@kali:~/Perl/debug$ perl dumper.pl
$VAR1 = 35;
$VAR1 = 'contenido';
$VAR2 = 'del';
$VAR3 = 'arreglo';
$VAR1 = [
          'contenido',
          'del',
          'arreglo'
        ];
$VAR1 = 'dos';
$VAR2 = 'segunda';
$VAR3 = 'clave';
$VAR4 = 'valor';
$VAR1 = {
          'dos' => 'segunda',
          'clave' => 'valor'
        };
$VAR1 = [
          'a',
          'b',
          'c'
        ];
$VAR2 = [
          1,
          2,
          3
        ];
$VAR1 = [
          [
            'a',
            'b',
            'c'
          ],
          [
```

```
        ];
$VAR1 = [
          [
            'a',
            'b',
            'c'
          ],
          [
            1,
            2,
            3
          ]
        ];
kali@kali:~/Perl/debug$
```

Debug

- **Data::Printer**

Este módulo escribe en STDERR como el módulo warnings y por ello es posible encontrar más fácil la salida.

Ordena los hash por defecto y permite ver los métodos de un objeto.

Este módulo no viene por defecto y debe instalarse:

```
1   sudo cpan -i Data::Printer
```

Data::Printer

```
1   use Data::Printer;
2
3   use LWP::UserAgent;
4   my $var = 35;
5   my @array = qw(contenido del arreglo);
6   my %hash = ( 'clave' => 'valor', 'dos' => 'segunda');
7   my @listas = (['a','b','c'], [1,2,3]);
8   my $ua = LWP::UserAgent->new();
9
10  p $var;
11  p @array;
12  p %hash;
13  p @listas;
14
15  p $ua;
```

Data::Printer

```
ali@kali:~/Perl/debug$ perl printer.pl
5

    [0] "contenido",
    [1] "del",
    [2] "arreglo"


    clave    "valor",
    dos      "segunda"


    [0] [
        [0] "a",
        [1] "b",
        [2] "c"
    ],
    [1] [
        [0] 1,
        [1] 2,
        [2] 3
    ]
```

```
LWP::UserAgent  {
    Parents       LWP::MemberMixin
    public methods (47) : add_handler, agent, clone, co
t_basic_credentials, get_my_handler, handlers, head, is_
head, patch, post, prepare_request, progress, protocols_
le, run_handlers, send_request, send_te, set_my_handler
    private methods (6) : _agent, _has_raw_content, _ma
    internals: {
        def_headers             HTTP::Headers,
        handlers                {
            response_header   HTTP::Config
        },
        local_address           undef,
        max_redirect            7,
        max_size                undef,
        no_proxy                [],
        protocols_allowed       undef,
        protocols_forbidden     undef,
        proxy                   {},
        requests_redirectable   [
            [0] "GET",
            [1] "HEAD"
        ],
        send_te                 1,
        show_progress           undef,
        ssl_opts                {
            verify_hostname   1
        },
        timeout                 180,
        use_eval                1
    }
}
kali@kali:~/Perl/debug$
```

Debug

- **Data::Show**

Este módulo permite utilizar la función show que recibe como argumento una sola variable y muestra:

- El nombre de la variable
- El archivo desde el que se ejecuta la función show
- La línea del archivo desde donde se ejecuta show
- El contenido de la variable en un formato legible

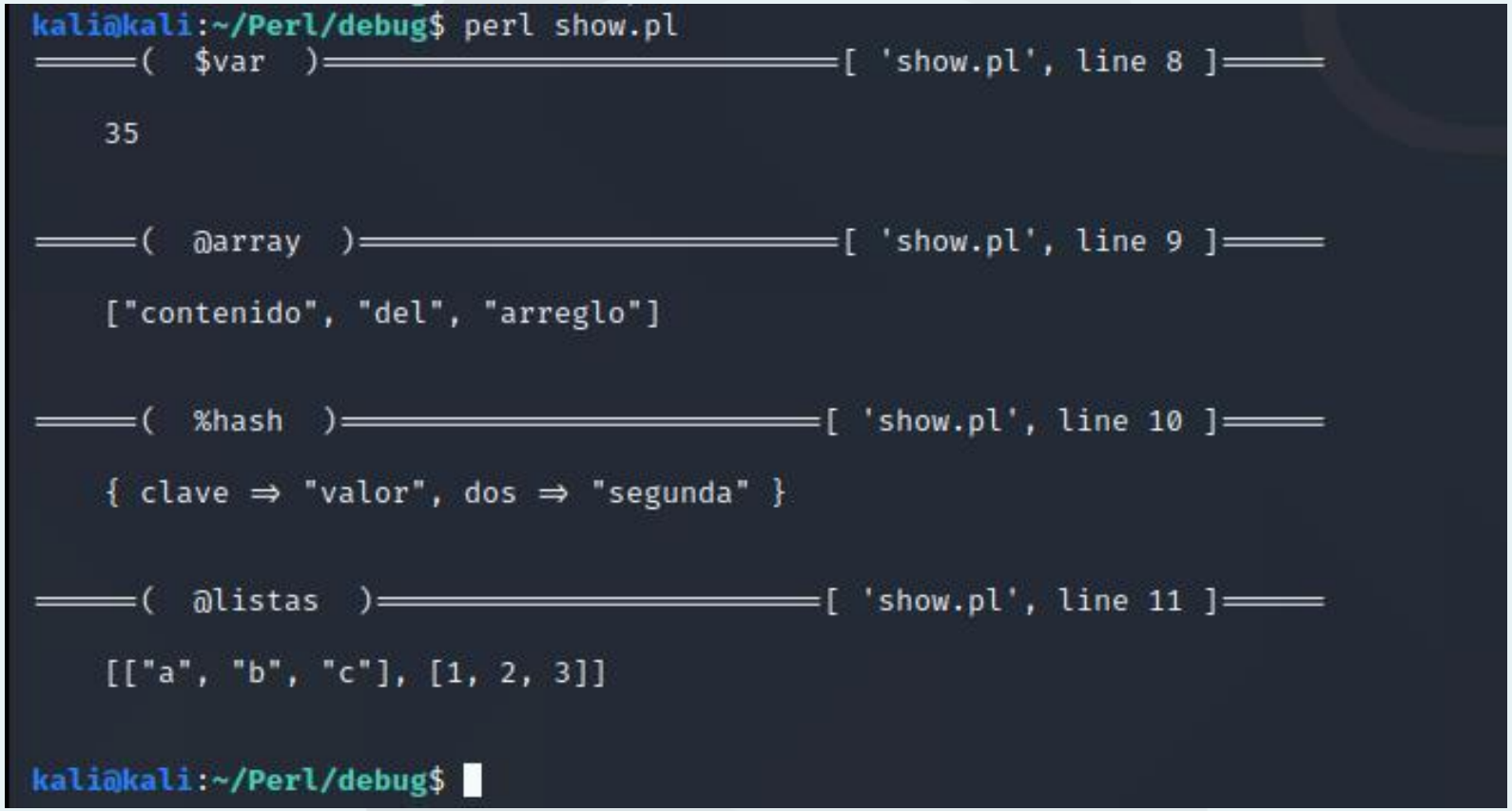
Este módulo debe instalarse mediante:

```
1   sudo cpan -i Data::Show
```

Data::Show

```
1   use Data::Show;
2
3   my $var = 35;
4   my @array = qw(contenido del arreglo);
5   my %hash = ( 'clave' => 'valor', 'dos' => 'segunda');
6   my @listas = (['a','b','c'], [1,2,3]);
7
8   show $var;
9   show @array;
10  show %hash;
11  show @listas;
```

Data::Show


kali@kali:~/Perl/debug$ perl show.pl
($var)['show.pl', line 8]
35
(@array)['show.pl', line 9]
["contenido", "del", "arreglo"]
(%hash)['show.pl', line 10]
{ clave ⇒ "valor", dos ⇒ "segunda" }
(@listas)['show.pl', line 11]
[["a", "b", "c"], [1, 2, 3]]
kali@kali:~/Perl/debug$

Ofuscación de código

Perl tiene la particularidad de que no requiere de indentación de código y por ello es posible escribir código que es ilegible o difícil de comprender, pero que Perl puede interpretar.

Existen competencias de código ofuscado de Perl.

Ofuscación de código

Ejemplo:

```
1  $_='loH,oe! lrdlwdo';$_.=$1,print$2while s/(..)(.)//;
```

¿Qué hace este código?

Ofuscación de código

Desofuscando con debug:

```
1  use Data::Printer;
2  $_='loH,oe! lrdlwdo';p$_,$_.=$1,p$2while s/(..)(.)//;
```

```
"H"
",oe! lrdlwdolo"
"e"
"! lrdlwdolo,o"
"l"
"rdlwdolo,o! "
"l"
"wdolo,o! rd"
"o"
"lo,o! rdwd"
","
"o! rdwdlo"
" "
"rdwdloo!"
"w"
"dloo!rd"
"o"
"o!rddl"
"r"
"ddlo!"
"l"
"o!dd"
"d"
"do!"
"!"
"do"
```

Ofuscación de código

Ofuscando nuestro código (ejemplo simple)

Existen módulos que nos permiten hacer la ofuscación de forma automática por diversión.

En este ejemplo utilizaremos Acme::EyeDrops que transforma un código fuente en una imagen y que al ejecutarse realiza lo que hace el programa original.

Es necesario instalarlo mediante:

```
1   sudo cpan -i Acme::EyeDrops
```

Ofuscación de código

Ejemplo:

- hola.pl (código fuente a ofuscar):

```
1   print "Ejemplo de código ofuscado";
```

- crea.pl (script que crea la figura):

```
1   use Acme::EyeDrops qw(sightly);
2
3   print sightly( { Shape      => 'camel',
4                    SourceFile => 'hola.pl' } );
```

Ofuscación de código

Ejecutar script crea.pl:

Ofuscación de código

Redirigir la salida al nuevo script ofuscado:

```
kali@kali:~/Perl/ofuscacion$ perl crea.pl > figura.pl
1 shapes completed.
kali@kali:~/Perl/ofuscacion$
```

Vemos el contenido de figura.pl:

```
kali@kali:~/Perl/ofuscacion$ cat figura.pl
eval eval '"'.


                                    ('['^"\+").(
          ('[')^                      ')').(``|')').
        (``|'.').                   ('['^'/').('{'^'['
 ).'\\'.'"'   .(`'^                '%').(``|'*').(``|
'%').(``|'-').('['^               '+').(``|',').((``)|
'/').('{'^'[').("\`"|           '$').(``|'%').('{'^'['
  ).(``|"\#").'\\'.(          '['^'#').(``|'#').("\^"^(
      ``|'-')).'\\'         .('['^'#').(``|'"').(('^')^(
     ``|'-')).(``       |'$').(``|')').(``|"'").("\`"|
    '/').('{'^'[').   (``|'/').(``|'&').('['^'.').('['^
   '(').(``|'#').   (``|'!').(``|'$').(``|'/').'\\'.'"'
   .';'.('!'^'+').  '"';$:='.'^'~';$~='@'|'(';$^=')'^'[';$≠
```

Ofuscación de código

Ejecutamos figura.pl:

```
kali@kali:~/Perl/ofuscacion$ perl figura.pl
Ejemplo de código ofuscadokali@kali:~/Perl/ofuscacion$
```

Vemos la salida del programa original.

Ejercicio

- **Ejercicio 17**

Utilizando el módulo Acme::EyeDrops (o buscar otro), realizar una figura diferente al camello mostrado (que imprima o realice cualquier cosa).

Modos de abrir archivos

Al momento de abrir los archivos se puede definir el modo en que se hace, ya sea para definir el modo de acceso o la codificación de caracteres.

La estructura es la siguiente:

```
1   # open(HANDLE, MODE, PATHNAME)
```

```
1  # Abrir archivo en modo lectura con codificacion de caracteres utf-8
2  open($handle, "<:encoding(UTF-8)", $filename);
3  # Abrir archivo en modo lectura de forma binaria
4  open($handle, "<:raw", $filename);
```

- https://perldoc.perl.org/PerlIO.html#Layers

Lectura/escritura en modo binario

Para leer bytes desde un archivo abierto en mod binario, se hace uso de la funcion read, el cual tiene la siguiente estructura:

```
1  read FILEHANDLE, VARIABLE, NO_BYTES
2  read FILEHANDLE, VARIABLE, NO_BYTES, OFFSET
```

Y la cual devuelve el numero de bytes leidos.

Para escribir se hace uso de la función print.

```
1  print FILEHANDLE, BYTES
```

Lectura/escritura en modo binario

```
1  open FILE, "<:raw", "archivo.file";
2  # open FILE, "<:", "archivo.file";
3  my $bytes_read = read FILE, my $bytes, 44;
```

```
1  open FILE, ">:raw", "archivo.file";
2  # open FILE, ">:", "archivo.file";
3  print FILE, $bytes;
```

Módulos - importación explícita

Cuando se crea un modulo se puede definir que funciones exportar.

De a misma forma, cuando un programa usa un modulo, puede definir explicitamente que funciones importar.

Esto proporciona algunos beneficios como:

- Identificar facilmente que función pertenece a qué módulo.
- Evitar conflictos al importar dos módulos con funciones con el mismo nombre.
- Importar unicamente las funciones necesarias para optimizar recursos.

Módulos - importación explícita

Dentro del módulo se debe incluir la funcion import del modulo estandar Export, además de definir en el arreglo @EXPORT la o las funciones que se exportarán

```
1  use Exporter qw(import);
2  our @EXPORT = qw(funcion1, funcion2);
```

El el programa que importa el módulo, deberán definirse explicitamente (despues del nombre del módulo) las funciones que desean importarse

```
1  use Funciones "funcion1";
```

Módulos - importación explícita

```perl
# Funciones.pm
package Funciones;

use Exporter qw(import);
our @EXPORT = qw(funcion1 funcion2);

sub funcion1 {
    $arg = shift;
    print "Funcion 1: $arg\n";
}

sub funcion2 {
    $arg = shift;
    print "Funcion 2: $arg\n";
}

sub funcion3 {
    $arg = shift;
    print "Funcion 3: $arg\n";
}

1;
```

```perl
# main.pl

use strict;
use warnings;

use lib "./";
use Funciones qw(funcion1);

funcion1("hola");
funcion2("adios");
```

Módulos - importación explícita

```
1  our @EXPORT = qw(funcion1 funcion2);
```

```
1  use Funciones qw(funcion1);
```

```
kali@kali:~/14g/mod$ perl main.pl
Funcion 1: hola
Undefined subroutine &main::funcion2 called at main.pl line 10.
kali@kali:~/14g/mod$
```

```
1  our @EXPORT = qw(funcion1);
```

```
1  use Funciones qw(funcion1 funcion2);
```

```
kali@kali:~/14g/mod$ perl main.pl
"funcion2" is not exported by the Funciones module
Can't continue after import errors at main.pl line 7.
BEGIN failed--compilation aborted at main.pl line 7.
kali@kali:~/14g/mod$
```

Módulos - importación explícita

```
1  use Funciones qw(funcion1 funcion2);
```

```
1  use Funciones qw(funcion1);
```

```
kali@kali:~/14g/mod$ perl main.pl
Funcion 1: hola
Undefined subroutine &main::funcion2 called at main.pl line 10.
kali@kali:~/14g/mod$
```

```
1  our @EXPORT = qw(funcion1 funcion2);
```

```
1  use Funciones qw(funcion1 funcion2);
```

```
kali@kali:~/14g/mod$ perl main.pl
Funcion 1: hola
Funcion 2: adios
kali@kali:~/14g/mod$
```

Cifrado Vigenere

Es un cifrado polialfabético que se realiza mediante la tabla de Vigenere.

Consiste en una matriz cuadrada (nxn) con las letras del alfabeto donde la primera fila es el alfabeto de la A-Z, la segunda es de la B-A, la tercera de la C-B, etc., ya que se va desplazando un espacio a la izquierda por cada iteración hasta la última fila, es decir, n-1.

Cifrado Vigenere

ENTRADA CLAVE	ENTRADA TEXTO PLANO																								
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Cifrado Vigenere

Ejemplo de cifrado:

Mensaje: “EJEMPLOSIMPLE”

Clave: “HOLA”

Mensaje cifrado:

“LXPMWZZSPAALL”

ENTRADA CLAVE \ ENTRADA TEXTO PLANO	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Cifrado Vigenere

Ejemplo de descifrado:

Mensaje cifrado: “LXPMWZZSPAALL”

Clave: “HOLA”

Mensaje:

“EJEMPLOSIMPLE”

ENTRADA CLAVE \ ENTRADA TEXTO PLANO	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Práctica

Práctica 7

Hacer un script que cifre y descifre un mensaje con el cifrado Vigenere.

Recibe como argumento el mensaje y la clave o si se ejecuta sin argumentos los pide por entrada estándar.

Nota: No utilizar módulos externos como Crypt::Vigenere.

Ejercicio

Ejercicio 18

Script que cifre y descifre archivos con AES.

Práctica

Práctica Extra

1. Implementar un programa que cifre y descifre archivos en perl usando el modulo Crypt::OpenSSL::AES.

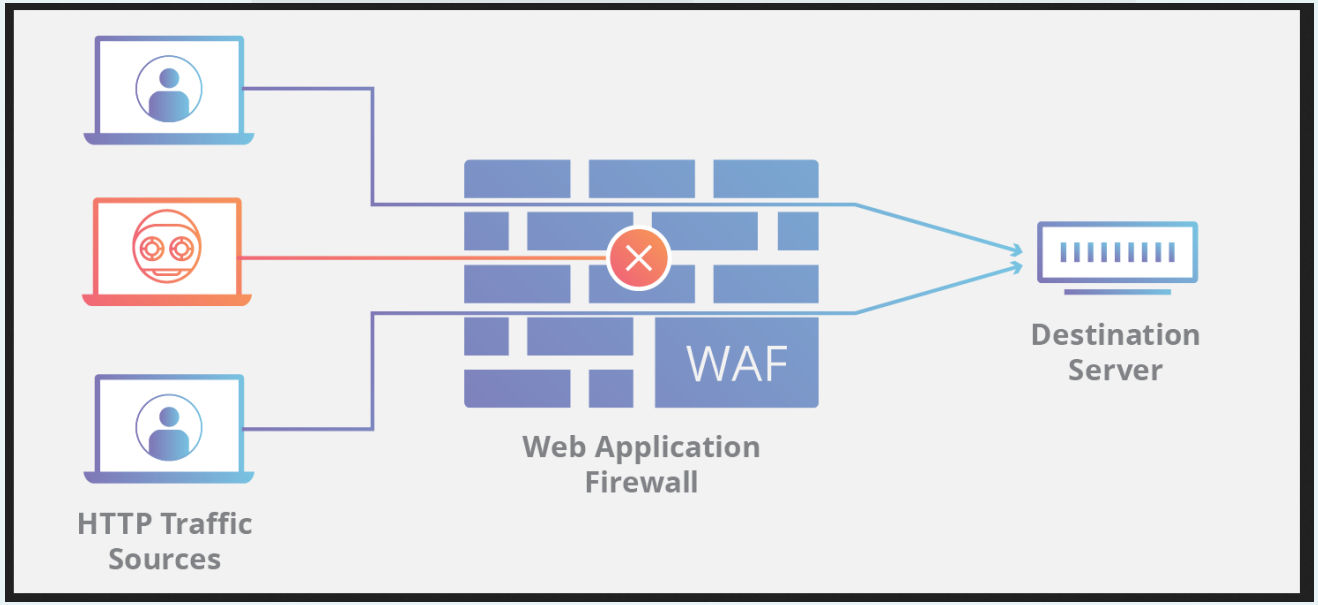
2.Se deberá crear un servidor el cual podrá enviar instrucciones al detectar una conexión, enviando comandos para realizar acciones en el sistema infectado.

3. Se deberá crear un cliente el cual se conecte a un servidor y esperará diversos comandos, con los cuales podrá cifrar/descifrar archivos y mostrar amenazas.

WAF (Web Application Firewall)

- Es un dispositivo o software que permite proteger aplicativos web mediante el filtrado y monitoreo de tráfico HTTP entre cliente y servidor web.

- Funciona en la capa de aplicación del modelo OSI.



WAF (Web Application Firewall)

- Funciona aplicando un conjunto de reglas para poder detectar los ataques y/o bloquearlos.

- Es posible debido a que inspecciona los headers y el cuerpo de las peticiones.

- Permite detectar y bloquear de ataques como:
  - XSS
  - SQLi
  - CSRF
  - File Inclusion
  - Etc.

WAF (Web Application Firewall)

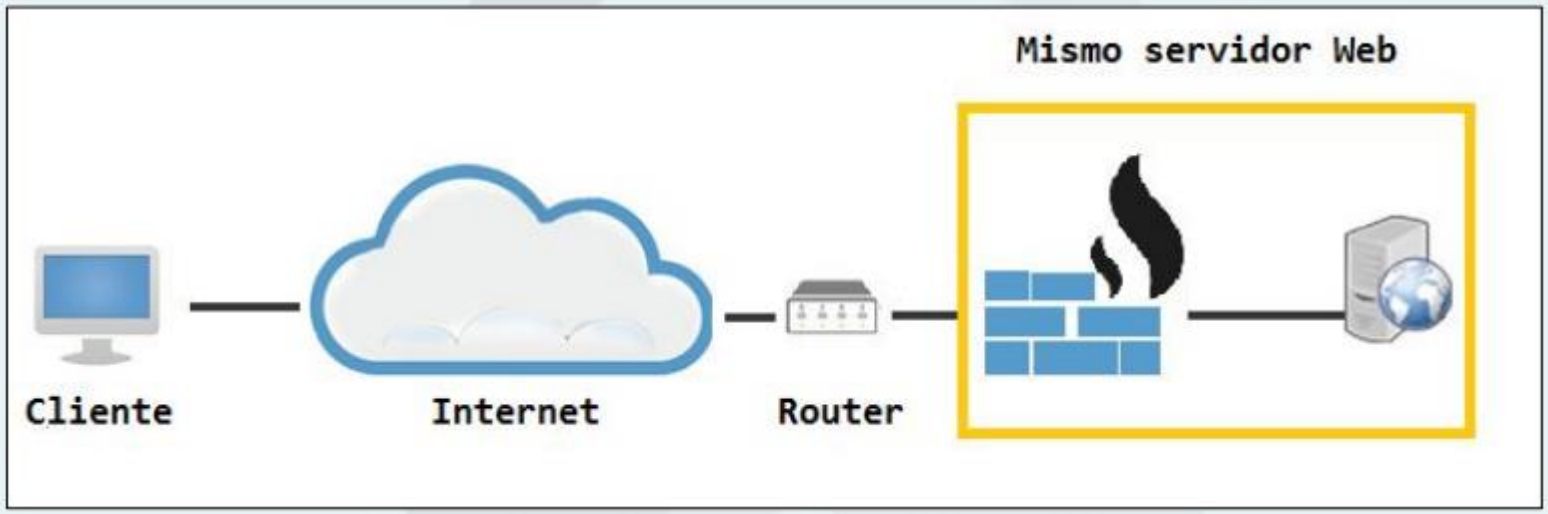
Existen tres tipos de WAF:

- Embebido o plugin
- Transparente (bridge)
- Proxy inverso

WAF – Modo embebido

- Este modo permite instalar el firewall de aplicaciones web como un complemento o plugin en el servidor que se quiere proteger.
- Su instalación depende totalmente del tipo de servidor web y el sistema operativo que posee.
- Para poder trabajar hace uso de recursos del hardware y software del servidor web.

WAF – Modo embebido

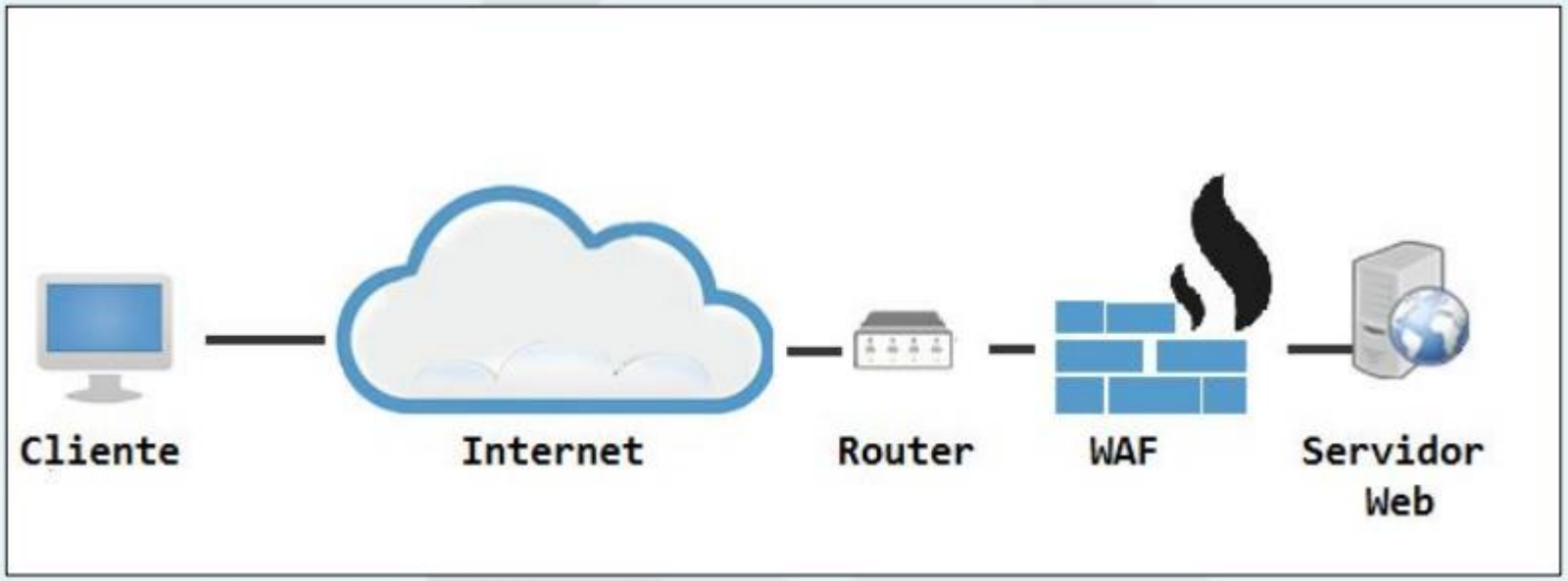


WAF – Modo transparente

- Funciona como un equipo que interconecta a dos segmentos de red diferentes, pero sus interfaces de red no tienen dirección IP.

- Es imperceptible para el usuario, de tal manera que no se requiera modificar la configuración de direcciones IP de los servidores web.

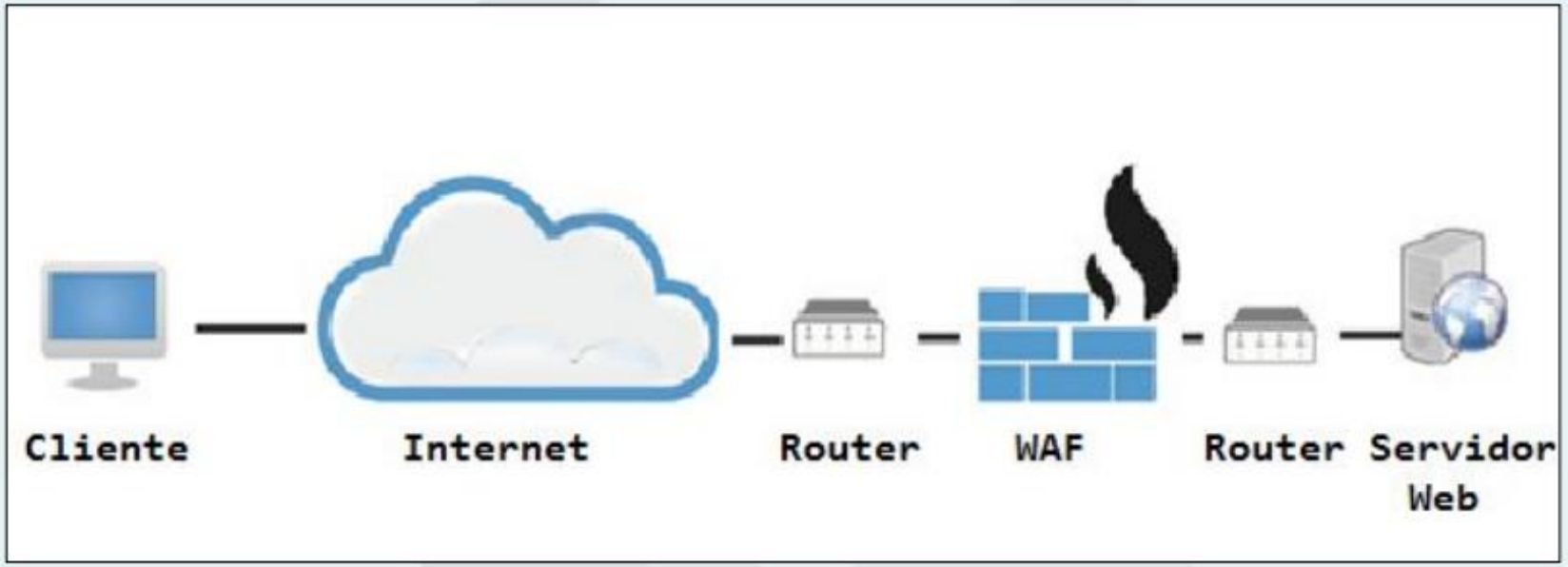
WAF – Modo transparente



WAF – Proxy inverso

- Funciona como un dispositivo que interconecta dos equipos o más segmentos de red, pero este si cuenta con dirección IP propia.
- Responde a las peticiones web como si este fuera el servidor web.
- Permite proteger múltiples servidores.

WAF – Proxy inverso



WAF (Web Application Firewall)

- **Reglas**

Son patrones escritos en forma de expresiones regulares.

Son las encargadas de filtrar lo que llega y no al servidor.

Si la regla coincide con el contenido de una petición se toman las acciones correspondientes

Práctica

Práctica 8

Hacer un script que realice ataques de fuerza bruta para conectarse por medio de SSH.

Recibe como argumentos el archivo de usuarios y el de contraseñas a probar.

Nota: Preparar el ambiente local para que prueben su script.

Nota: Pueden utilizar módulos externos para realizar las conexiones SSH.