

# Seguridad Y Criptación

Javier Fernández Rivera - [www.aurea.es](http://www.aurea.es)

---

## INDICE

Prologo



### ○ Teórico

- Seguridad
- Amenaza a la seguridad
- Ataque a la seguridad
- Tipos de ataques a la seguridad
  - Interrupción
  - Intercepción
  - Modificación
  - Fabricación
- Ataques pasivos
- Ataques Activos
- Virus, gusanos y caballos de troya
- Servicios de seguridad
  - Autenticación
  - identificación de usuarios
  - contraseñas
  - artefactos
  - identificación física
  - Confidencialidad
  - Integridad
  - Control de acceso
  - Disponibilidad

- Mecanismos de seguridad
  - Intercambio de autenticación
  - Cifrado
  - Integridad de los datos.
  - Firma digital
  - Control de acceso
  - Trafico de relleno
  - Control de encaminamiento
  - Unicidad
  
- Criptografía
  - Definición
  - Etimología
  - Encriptación
  - Desencriptación
  - Criptoanálisis
  - Criptografía
  
  - Partes de la Critpgrafía
    - Código
    - Clave
  
  - Métodos de encriptación
    - Sustitución
    - Transposición
  
    - Estenografías
  
- **Practico**
  - Introducción
  
  - Historia
  
  - Problemas reales de la encriptación
  - Solución (encriptación "dinámica")
  
  - Introducción al lenguaje scripting para mIRC
    - Variables
    - Identificadores
    - Sentencias condicionales
    - Bucles
    - Eventos
  
  - Encriptación en OrioNScripT
    - Fichero
    - Código
  
  - Explicación de código

Este manual tiene un nivel medio-avanzado en cuanto a los tratamientos de los contenidos que hay en el. El trabajo se divide en dos partes:

1. *Teórico*: En esta parte intentare explicar conceptos relacionados con el mundo de la seguridad a nivel de software. Servirá como una base teórica de aprendizaje.
2. *Practico*: Veremos un ejemplo practico del mundo de la encriptación, para ello he tomado como referencia un script de IRC: OriON ScripT en el cual llevo desarrollándolo desde hace bastante tiempo. Además podremos construir una pequeña utilidad bajo BAT de MSDOS para detectar posibles troyanos en nuestro equipo.

La palabra seguridad abarca muchos campos pero en este documento he tratado de centrarme exclusivamente en la seguridad referida a la comunicación a través de redes especialmente en internet y dentro de la red de redes pondremos ejemplos de seguridad en el campo de la criptografía basándonos en un entorno IRC (Internet Relay Chat).

Orientamos todo ello en un plano más bien lógico, dejando de un lado el aspecto de la seguridad física. Dentro del apartado de la seguridad nos introduciremos de lleno en el campo de la criptografía, poniendo en el un especial interés, tanto teórico como practico como ya se venia diciendo anteriormente.

La criptografía será el tema que tratare en profundidad y con el que también finalizare este trabajo.

## TEORICO

### ▪ Seguridad

Es la ciencia que trata de impedir, prevenir, detectar y corregir violaciones durante la transmisión de la información.

### ▪ Amenaza a la seguridad

Es el ente, persona, maquina o suceso que en un momento determinado podría dar lugar a la violación de la información, esto es que la información deje de ser confidencial, o romper su integridad lógica.

### ▪ Ataque a la seguridad

Es la amenaza de seguridad hecha realidad.

### ▪ Tipos de ataques a la seguridad

Existen cuatro tipos de ataques o amenazas a la seguridad de la información:

- **Interrupción**: Se trata de un ataque mediante el cual se deja inhabilitado (destruido o no disponible) un componente que entraba en juego en la comunicación de la información. Como por ejemplo, cortar la línea de comunicación. Este tipo de ataques atañen a la integridad física de los recursos del sistema.
- **Intercepción**: Una entidad que en principio no esta autorizada o identificada, es capaz de acceder a los recursos del sistema, así como a su información, con lo que atañe a la confidencialidad de esa información. Un ejemplo de este tipo de ataques es realizado por grandes gobiernos anglosajones (EEUU, Inglaterra, Australia, Canadá) en los que a la cabeza se coloca EEUU (como no), a través de su RED de satélites, antenas y un largo etc., captan e interceptan millones de transferencias de información a través de "todas" las redes de comunicación existentes: Internet, mail, Teléfono, Fax, etc. Dicho hecho fue denunciado por la UE como un espionaje industrial (descarado), no se supo mas del tema. En la

actualidad EPSILON ha crecido, y controla incluso las transacciones de bolsa. Con este tipo de actos, todavía existen personas que se pregunta ¿Por qué EEUU es la primera potencia mundial?, ¿Juega limpio?, etc.

- **Modificación:** La entidad ahora no solamente es capaz de acceder al recursos sino que también puede manipularlo a su antojo. Este tipo de ataques por ejemplo es muy comúnmente practicado a través de troyanos.
- **Fabricación:** La entidad no autorizada introduce objetos falsificados en el recurso del sistema de la información.

Todos estos ataques se pueden agrupar en dos subgrupos.

### ▪ Ataques pasivos

En este tipo de ataques se usa la interceptación, o sea que se monitoriza o se escucha únicamente a la víctima en ningún momento se emprenden acciones o violaciones contra su sistema o transferencia de información.

### ▪ Ataques Activos

En este tipo de ataques si se rompe esa confidencialidad o integridad de la información mantenida por los recursos que intervienen en la comunicación.

### ▪ Virus, gusanos y caballos de troya

**Programas caballos de trola (troyanos):** Estos programas ocultan su función habilidad. Por ejemplo un programa falso de login, con idéntica identificación que el original. El usuario teclea su password y el usuario teclea su password a partir de entonces el intruso tiene acceso a todo el sistema de ficheros de la victima, esto es a todo su sistema lógico, absolutamente todo.

Un caballo de trola o troiano consta de dos partes principales, por un lado tiene la consola este es el dispositivo desde el cual se controlan a los usuarios infectados. Y por otro lado tiene el fichero que cumple la misión de infectar.

Este fichero al ejecutarse deja una "línea" virtual abierta dejando la sesión en "login" y el dueño del troiano puede desde su consola actuar sobre todo, absolutamente todo el sistema de ficheros del usuario que ha infectado.

A continuación explicare un ejemplo, muy conocido y propio de lamers con ganas de sentirse hackers.

El lamer en cuestión es el que dispone de un troiano, se encuentra en el IRC. Y encuentra a una persona que es un poco novatillo. Entonces le dice al usuario novatillo que le va a pasar un programa de guerra o de defensa para no se que royo, bueno la típica chorada. Lo que en verdad le va a pasar es el fichero para infectarse, el que abre la sesión una vez clikeado, el que hace un auto-login. Bueno, el lamercillo se lo pasa por dcc al novato, y este lo cojeé con ilusión, lo abre y ve que aquello, no hace nada (en algunos troianos el fichero desaparece). Bueno, pues a partir de que el novato ejecuto ese fichero, es como si dejara una sesión abierta, una línea de comunicación directa entre, la consola del troiano, que la tiene el lamer y todo el sistema de ficheros del novato. A partir de entonces el lamer puede actuar a sus anchas dentro del ordenador del novato, modificar, borrar ficheros, etc. Actúa totalmente sobre el software del novato. Pudiendo pues joderle completamente el ordenador.

Algunos de estos troianos son:

NETBUS (NT) = puerto: 12345

BACKORIFICE (BO) = puerto: 31337  
SUBSEVEN (S7) = puerto: ----  
PARADISE (P) = puerto: ----

He creado una utilidad para OriON v4.1 en la cual es posible detectar ciertas infecciones por parte de estos trojanos, escaneando su puerto de entrada, y viendo si este esta abierto o a la escucha. Este scan podemos realizarlo a traves del Sistema operativo MSDOS por ello la utilidad esta programada bajo lenguaje script bat de procesamiento por lotes. Y se presenta a continuación:

```
@echo off
echo.
echo =====
echo [ OriON ScaN - Trojans ]
echo Escaneador de posibles infecciones por trojanos
echo Idea por ^Banzai^, Lista de ports por phsyko
echo Editado por MoAsT, Creado por Quasi
echo =====
echo.
echo Preparando lista interna de trojanos y puertos.....
echo.
echo Dependiendo del resultado de la conexion a un puerto
echo estaras o no infectado por el trojano.
echo.
echo -- La conexion al puerto puede dar 3 resultado.
echo -1-- A la escucha, es un sys de seguridad [no infectado].
echo -2-- Conexión establecida, posiblemente estés infectado [=infectado].
echo -3-- No da ningún dato, la conexion se encuentra cerrada [no infectado].
echo.
pause
echo.
echo Escaneando AimSpy (777)
netstat -an | find "777" > scantrojans.txt
echo Escaneando Ambush (10666)
netstat -an | find "10666" >> scantrojans.txt
echo Escaneando AOLTrojan1.1 (30029)
netstat -an | find "30029" >> scantrojans.txt
echo Escaneando Attack FTP (666)
netstat -an | find "666" >> scantrojans.txt
echo Escaneando BackConstruction1.2 (5400)
netstat -an | find "5400" >> scantrojans.txt
echo Escaneando Backdoor (1999)
netstat -an | find "1999" >> scantrojans.txt
echo Escaneando Back Orifice 2000 (31337)
netstat -an | find "31337" >> scantrojans.txt
echo Escaneando BigGluck, aka TN (34324)
netstat -an | find "34324" >> scantrojans.txt
echo Escaneando Bla (20331)
netstat -an | find "20331" >> scantrojans.txt
echo Escaneando Bla1.1 (1042)
netstat -an | find "1042" >> scantrojans.txt
echo Escaneando BladeRunner (5400)
netstat -an | find "5400" >> scantrojans.txt
echo Escaneando BO jammerkillahV (121)
netstat -an | find "121" >> scantrojans.txt
```

```
echo Escaneando Coma 10607
netstat -an | find "10607" >> scantrojans.txt
echo Escaneando Chupacabra (20203)
netstat -an | find "20203" >> scantrojans.txt
echo Escaneando Deep Throath 1,2,3.x (6670)
netstat -an | find "6670" >> scantrojans.txt
echo Escaneando DeltaSource ( 6883)
netstat -an | find "6883" >> scantrojans.txt
echo Escaneando Der Spaehher 3 (1000)
netstat -an | find "1000" >> scantrojans.txt
echo Escaneando Der Spaehher 3 (1001)
netstat -an | find "1001" >> scantrojans.txt
echo Escaneando Der Spaehher 3 (2000)
netstat -an | find "2000" >> scantrojans.txt
echo Escaneando Der Spaehher 3 (2001)
netstat -an | find "2001" >> scantrojans.txt
echo Escaneando Devil 1.03 (65000)
netstat -an | find "65000" >> scantrojans.txt
echo Escaneando Doly Trojan (1011)
netstat -an | find "1011" >> scantrojans.txt
echo Escaneando Doly Trojan v1.35 (1010)
netstat -an | find "1010" >> scantrojans.txt
echo Escaneando Doly Trojan v1.5 (1015)
netstat -an | find "1015" >> scantrojans.txt
echo Escaneando Eclipse 2000 (12701)
netstat -an | find "12701" >> scantrojans.txt
echo Escaneando FileNail (4567)
netstat -an | find "4567" >> scantrojans.txt
echo Escaneando Firshotcker (5321)
netstat -an | find "5321" >> scantrojans.txt
echo Escaneando Fore, Schwindler (50766)
netstat -an | find "50766" >> scantrojans.txt
echo Escaneando FTP99CMP (1492)
netstat -an | find "1492" >> scantrojans.txt
echo Escaneando Gatecrasher (6969)
netstat -an | find "6969" >> scantrojans.txt
echo Escaneando GirlFriend (21554)
netstat -an | find "21554" >> scantrojans.txt
echo Escaneando Gjamer (12076)
netstat -an | find "12076" >> scantrojans.txt
echo Escaneando Hack'99 KeyLogger (12223)
netstat -an | find "12223" >> scantrojans.txt
echo Escaneando Hack'a'tack (31787)
netstat -an | find "31787" >> scantrojans.txt
echo Escaneando Hackers Paradise (456)
netstat -an | find "456" >> scantrojans.txt
echo Escaneando HVL Rat 5 (2283)
netstat -an | find "2283" >> scantrojans.txt
echo Escaneando ICQKiller (7789)
netstat -an | find "7789" >> scantrojans.txt
echo Escaneando IcqTrojan (4950)
netstat -an | find "4950" >> scantrojans.txt
echo Escaneando IcqTrojen (4950)
netstat -an | find "4950" >> scantrojans.txt
echo Escaneando Illusion Mailer (5521)
```

```
netstat -an | find "5521" >> scantrojans.txt
echo Escaneando InCommand 1.9 (9400)
netstat -an | find "9400" >> scantrojans.txt
echo Escaneando Indoctrination (6939)
netstat -an | find "6939" >> scantrojans.txt
echo Escaneando Inkiller (9989)
netstat -an | find "9989" >> scantrojans.txt
echo Escaneando iNi-killer (9989)
netstat -an | find "9989" >> scantrojans.txt
echo Escaneando Kuang (30999)
netstat -an | find "30999" >> scantrojans.txt
echo Escaneando Kuang2 theVirus (17300)
netstat -an | find "17300" >> scantrojans.txt
echo Escaneando Logged! (20203)
netstat -an | find "20203" >> scantrojans.txt
echo Escaneando Master Paradise (31)
netstat -an | find "31" >> scantrojans.txt
echo Escaneando Master Paradise (40423)
netstat -an | find "40423" >> scantrojans.txt
echo Escaneando Maverick's Matrix (1269)
netstat -an | find "1269" >> scantrojans.txt
echo Escaneando Millenium (20000)
netstat -an | find "20000" >> scantrojans.txt
echo Escaneando NeTadmin (555)
netstat -an | find "555" >> scantrojans.txt
echo Escaneando NetBus 1.x (avoiding Netbuster) (12346)
netstat -an | find "12346" >> scantrojans.txt
echo Escaneando NetBus Pro (20034)
netstat -an | find "20034" >> scantrojans.txt
echo Escaneando NetMetro 1.0 (5031)
netstat -an | find "5031" >> scantrojans.txt
echo Escaneando NetMetropolitan 1.04 (5031)
netstat -an | find "5031" >> scantrojans.txt
echo Escaneando NetMetropolitan 1.04 (5032)
netstat -an | find "5032" >> scantrojans.txt
echo Escaneando NetMonitor (7306)
netstat -an | find "7306" >> scantrojans.txt
echo Escaneando NetSphere (30100)
netstat -an | find "30100" >> scantrojans.txt
echo Escaneando Netspy (1033)
netstat -an | find "1033" >> scantrojans.txt
echo Escaneando NetSpy DK (31339)
netstat -an | find "31339" >> scantrojans.txt
echo Escaneando OOTL+OOTLT Cart (5011)
netstat -an | find "5011" >> scantrojans.txt
echo Escaneando Pass Riper (2023)
netstat -an | find "2023" >> scantrojans.txt
echo Escaneando Phase0 (555)
netstat -an | find "555" >> scantrojans.txt
echo Escaneando Phineas (2801)
netstat -an | find "2801" >> scantrojans.txt
echo Escaneando Portal of Doom (9875)
netstat -an | find "9875" >> scantrojans.txt
echo Escaneando PortalOfDoom (9872)
netstat -an | find "9872" >> scantrojans.txt
```

echo Escaneando Priority (16969)  
netstat -an | find "16969" >> scantrojans.txt  
echo Escaneando Progenic Trojan (11223)  
netstat -an | find "11223" >> scantrojans.txt  
echo Escaneando Prosiak 0.47 (22222)  
netstat -an | find "22222" >> scantrojans.txt  
echo Escaneando Psyber Streaming Server (1509)  
netstat -an | find "1509" >> scantrojans.txt  
echo Escaneando Remote Windows Shutdown (53001)  
netstat -an | find "53001" >> scantrojans.txt  
echo Escaneando RoboHack (5569)  
netstat -an | find "5569" >> scantrojans.txt  
echo Escaneando Schoolbus 1.6 (54321)  
netstat -an | find "54321" >> scantrojans.txt  
echo Escaneando Schoolbus 2.0 (54321)  
netstat -an | find "54321" >> scantrojans.txt  
echo Escaneando Schwindler 1.82 (21554)  
netstat -an | find "21554" >> scantrojans.txt  
echo Escaneando Senna Spy Trojans (11000)  
netstat -an | find "11000" >> scantrojans.txt  
echo Escaneando Shitheap (69123)  
netstat -an | find "69123" >> scantrojans.txt  
echo Escaneando Shitheap (6912)  
netstat -an | find "6912" >> scantrojans.txt  
echo Escaneando Shiva Burka (1600)  
netstat -an | find "1600" >> scantrojans.txt  
echo Escaneando ShockRave (1981)  
netstat -an | find "1981" >> scantrojans.txt  
echo Escaneando Silencer (1001)  
netstat -an | find "1001" >> scantrojans.txt  
echo Escaneando Silencer (1001)  
netstat -an | find "1001" >> scantrojans.txt  
echo Escaneando Socket23 (30303)  
netstat -an | find "30303" >> scantrojans.txt  
echo Escaneando SoftWar (1207)  
netstat -an | find "1207" >> scantrojans.txt  
echo Escaneando SpySender (1807)  
netstat -an | find "1807" >> scantrojans.txt  
echo Escaneando Stealth Spy (555)  
netstat -an | find "555" >> scantrojans.txt  
echo Escaneando Streaming Audio Trojan (1170)  
netstat -an | find "1170" >> scantrojans.txt  
echo Escaneando Striker (2565)  
netstat -an | find "2565" >> scantrojans.txt  
echo Escaneando SubSeven (1243)  
netstat -an | find "1243" >> scantrojans.txt  
echo Escaneando Telecomando (61466)  
netstat -an | find "61466" >> scantrojans.txt  
echo Escaneando The Invasor (2140)  
netstat -an | find "2140" >> scantrojans.txt  
echo Escaneando The tHing (6400)  
netstat -an | find "6400" >> scantrojans.txt  
echo Escaneando The Unexplained (29891)  
netstat -an | find "29891" >> scantrojans.txt  
echo Escaneando TheSpy (40412)

```

netstat -an | find "40412" >> scantrojans.txt
echo Escaneando TheThing 1.6 (6000)
netstat -an | find "6000" >> scantrojans.txt
echo Escaneando Tiny Telnet Server (34324)
netstat -an | find "34324" >> scantrojans.txt
echo Escaneando Total Eclypse 1.0 (3791)
netstat -an | find "3791" >> scantrojans.txt
echo Escaneando Transcout 1.1 + 1.2 (1999)
netstat -an | find "1999" >> scantrojans.txt
echo Escaneando Trojan Spirit 2001 a (33911)
netstat -an | find "33911" >> scantrojans.txt
echo Escaneando TrojanCow (2001)
netstat -an | find "2001" >> scantrojans.txt
echo Escaneando UglyFtp (23456)
netstat -an | find "23456" >> scantrojans.txt
echo Escaneando Vampire (6669)
netstat -an | find "6669" >> scantrojans.txt
echo Escaneando Voodoo (1245)
netstat -an | find "1245" >> scantrojans.txt
echo Escaneando WebEx (1011)
netstat -an | find "1011" >> scantrojans.txt
echo Escaneando WhackJob (23456)
netstat -an | find "23456" >> scantrojans.txt
echo Escaneando Wincrash (5742)
netstat -an | find "5742" >> scantrojans.txt
echo Escaneando Wincrash2 (2583)
netstat -an | find "2583" >> scantrojans.txt
echo Escaneando Xtcp 5550
netstat -an | find "5550" >> scantrojans.txt
echo.
echo Si alguna conexion resulta infectada.
echo Será grabado el dato en el fichero scantrojans.txt
echo Ver los resultados en [ orion\scantrojans.txt ]
echo.
echo Fin del scan.

```

El programa es muy secuencial también esto es así al estar apoyado en un lenguaje BAT, de todas formas todas estas líneas podrían simplificarse a traves de un programa en C que use un bucle iterativo y haga estas llamadas al SO MSDOS con la función system(), pasando como parámetro el comando a realizar.

La explicación al programa es muy sencilla. Su funcionamiento se simplifica a dos líneas.

```

echo Escaneando Xtcp 5550
netstat -an | find "5550" >> scantrojans.txt

```

Como todos sabemos “echo mensaje” muestra texto por pantalla, eso es lo que vemos en la pantalla de MSDOS al ejecutar esta utilidad.

El comando “netstat” es una utilidad que implemento el sistema operativo MSDOS para hacer frente a las innovaciones marcadas por las redes. NETSTAT es un comando que Chekea estados de las conexiones del sistema.

El comando se emplea aquí para chequear en modo –an las conexiones abiertas este comando trabajo internamente mediante sockets (el tema de sockets no lo puedo tratar aquí sería extenderse demasiado), solo es necesario saber, para comprender el funcionamiento del comando que si netstat devuelve un puerto que es el que buscamos, con el comando find enlazado mediante un “|” al netstat, es que la conexión existe y se encuentra abierta o bien a la escucha, en tal caso se procede a almacenar ese dato en el fichero

scantrojans.txt mediante un redireccionamiento no destructivo ">>". Esto quiere decir que posiblemente se encuentre infectado por ese tipo de troyano en el ejemplo (Xtcp), puesto que este actúa en el puerto 5550.

### Gusanos y virus informáticos

- Los primeros son programas en si que usan desproporcionadamente los recursos del sistema bloqueando.
- Los segundos son trozos de código que se copian en otros programas o que son capaces de borrar ficheros, etc.

Normalmente el virus se suele distribuir a traves de medios que lleguen a muchos usuarios, cuantos más mejor. Por ello Internet es un buen medio para la transmisión de virus. También lo sería un juego, etc.

## ▪ Servicios de seguridad

Existen varias técnicas o servicios que pueden mantener la seguridad de la información.

- **Autenticación:** Es el proceso mediante el cual se determina si una entidad esta autorizada o no.
  - **Identificación de usuarios:** La seguridad de acceso de un sistema se basa en la combinación de tres tipos de identificación:
    - Por contraseña
    - Artefactos
    - Física
  - **Contraseñas:** Suelen estar almacenadas en ficheros. El problema es que muchos usuarios usan el nombre de ciudades, parientes, fechas de nacimiento, etc.; fáciles de obtener. Hay software especial, que tras un numero de intentos te deniegan el acceso, bloquean la cuenta. Y otros que tras un periodo de tiempo determinado piden y obligan a cambiar la password.
  - **Artefactos:** Suelen ser bandas magnéticas, tarjetas electrónicas, donde la password esta almacenada internamente en la tarjeta lo que hace difícil hacerse con ella.
  - **Identificación física:** consiste en usar características físicas del usuario:
  - **Fisiológicas:** huellas dactilares, características faciales, geometría de la mano, etc.
  - **De comportamiento:** análisis de firma, o patrones de voz.

- **Confidencialidad:** Vela porque solo la información sea vista por usuarios autorizados.

- **Integridad:** Vela porque solo la información sea tratada, procesada y manipulada por usuarios autorizados.

- **Control de acceso:** Debemos de controlar el acceso al sistema, tanto física como lógicamente.

Con física, me refiero a que nadie no deseable pueda entrar en el departamento donde se encuentra el servidor y actuar a sus anchas.

Respecto a lógicamente, quiero decir que debemos controlar el acceso, que no se produzca un acceso masivo que pueda ocasionar fallos de seguridad.

- **Disponibilidad:** Requiere que los recursos del sistema de información se encuentren siempre disponibles a las entidades autorizadas.

## ▪ **Mecanismos de seguridad**

Antes de todo, decir que no existe ningún mecanismo capaz de mantener la seguridad en todos los aspectos citados. Siempre existen agujeros y personas capaces de detectarlos, como también siempre se terminan programando parches para dichos agujeros también llamados bugs o errores.

Entendemos por objeto tanto a las unidades del ordenador (discos, impresoras, etc) como a las informaciones que se almacenan (archivos de datos, programas, etc.). A cada objeto podemos acceder a través de operaciones (leer, escribir, ejecutar). Un dominio es un conjunto de objetos y las operaciones permitidas para cada objeto. La capacidad para ejecutar una operación sobre un objeto es un derecho de acceso. Así pues un dominio es un conjunto de derechos de acceso cada uno de los cuales está formado por un par de la forma: <nombre del objeto, conjunto de sus derechos>

En UNIX a través del identificador de usuarios (uid) y del grupo (gid) se define el dominio. Es decir una lista de todos los objetos a los que puede tener acceso cada usuario y el tipo de permiso de acceso.

Los distintos mecanismos de seguridad podríamos clasificarlos en:

1. **Intercambio de autenticación:** Se intercambia la autenticación entre las entidades participantes en la transferencia de la información.
2. **Cifrado:** Consiste en que solo la información es legible y entendible por las entidades autenticadas, el resto de terceras entidades capta esa información como datos ilegibles. El cifrado se lleva a cabo a través de una clave.
3. Mas adelante entraremos de lleno en este apartado poniendo como ejemplo una aplicación real. Usada en el IRC. Así nos servirá de práctica.
4. **Integridad de los datos.**
5. **Firma digital.**
6. **Control de acceso:** Anteriormente ya explicado.
7. **Traffic de relleno:** Consiste en enviar información de relleno junto a los datos válidos para que el intruso no sepa ni pueda calcular la situación de la información válida.
8. **Control de encaminamiento:** De routers, encaminadores, consiste en enviar la información, o sea transmitirla por zonas denominadas "autorizadas", por las que se supone no existen ningún tipo de violación de la seguridad.
9. **Unicidad:** Empleo de algoritmos para un determinado cifrado con formato, añadiendo fecha y hora, etc.

## ▪ Criptografía

- **Definición:** La criptografía es la ciencia que estudia el arte de la escritura.
- **Etimología:** Su etimología es propia del griego.

Viene de “CRIPTO” que sería secreto y “GRAFIA” viene de grafo y esto es escritura. Con lo cual formaríamos “Escritura secreta”.

La criptografía se basa en la comunicación o transferencia de información entre un número limitado de entidades las cuales alteran la información haciendo de ello un resultado incomprensible para el resto de terceras entidades no autenticadas.

Las reglas de autenticación pueden ser varias, en internet por ejemplo podrían ser un rango de IPs, una palabra clave, etc.

Cuando tenemos la información y las entidades alteran esa información se está produciendo un fenómeno conocido dentro de la criptografía como cifrado, encriptación o codificación.

Si se pasa esa información encriptada a información entendible, el proceso a la inversa, se produce el fenómeno de descifrado, desencriptación, o decodificación.

De ahora en adelante usare los términos: codificación y decodificación por razones de gusto personal. ;) Abreviadas serán: cod/dec.

El criptoanálisis agrupa al conjunto de procedimientos o técnicas que se ofrecen para encontrar la clave de la cod/dec y así proceder a formar parte de ese grupo de autenticación capaz de codificar y decodificar las transferencias de la información.

La criptología: Engloba todos los conceptos.

criptología = criptografía + criptoanálisis + criptógrafo + criptoanalista.

En la criptografía, existen dos partes muy importantes y claramente diferenciadas.

Hablamos pues de:

- **Código**
- **Clave**

Donde el código es la traducción de toda la información. Por ejemplo transformamos una frase “Orion Script para IRC” en caracteres ilegibles “%\$.&%.\$%/\$”\$&”, esto es el código.

Mientras que la clave es el paso para proceder a la decodificación de esos caracteres ilegibles, la clave convierte una entidad tercera en una entidad autenticada capaz de ver legiblemente el código procesándolo por una descodificación. Si puede hacer esto, también será capaz de proceder a una codificación.

## ▪ Métodos de encriptación

- **Sustitución:** Este método es el más usado, consiste en sustituir cada carácter de la información por otro carácter alternativo. Por ejemplo, a la letra "O" le asignamos el carácter de codificación (que se llama) "&". Con lo que siempre que pongamos "O" veremos "&" si tenemos la clave entonces si veríamos "O", debido a que se produciría un proceso de decodificación.
- **Transposición:** Consiste en transformar el posicionamiento inicial de los caracteres que constituyen el conjunto de la información. Esto se hace a través de algoritmos matemáticos. Lo que se provoca es alterar la posición de las letras.
- **Estenografías:** Se trata de ocultar mensajes e información tras una imagen.

## PRACTICO

### ▪ Referencia Orion Script

Aplicación: Se trata de un programa "script" basado en mIRC para IRC.

::: ircOrion.net ::: Versión: 5 (C)1999-2002 by Quasi (Javier Fernández Rivera)

- Cliente IRC: mIRC 6.01 (c) bY Khaled Mardam-Bey
- Web oficial: <http://www.ircOrion.net>
- Portal patrocinador: <http://www.telodamos.com>
- Web patrocinador: <http://www.ayuda-internet.net>
- Mail script: [script@ircorion.net](mailto:script@ircorion.net)
- Mail autor: [quasi@ircorion.net](mailto:quasi@ircorion.net)

- **Artículos publicados:** Revista @rroba, especializada en temas informáticos: Redes, Linux, Hacking, Internet, Prehacking. Ha dedicado 2 números al Orion Script y Xscript como mejores scripts en España, documentando código para tutorial de aprendizaje en scripting.

### ▪ Historia

Hace unos años concretamente al final del 1999 se me ocurrió la idea de programar alguna utilidad para que los usuarios que chateaban en el IRC con el Orion y concretamente en canales (chats, salas, etc), pudieran escribir mensajes e información y mandarla públicamente al resto del personal de esa sala, y que solo pudiera ser interpretado por usuarios del Orion de forma exclusiva.

Imaginemos que en un canal del IRC existen como todos sabemos varios usuarios chateando cada uno con su cliente de IRC y script preferido: Orion, Xscript, IRCap, etc.

La idea era que en esa sala si por ejemplo hubiera 10 usuarios y 5 de ellos usaban el Orion se pudieran escribir de forma pública esto es viéndolo también los otros 5 usuarios con otros programas scripts. Con la diferencia de que estos otros 5 usuarios solo vieran textos codificados. Y los 5 usuarios del Orion pudieran ver sus mensajes perfectamente.

Vamos un sistema de codificación de y para Orion.

Con el tiempo, fui desarrollando la idea, poco a poco mejorándola, más adelante se dio a conocer, con mucho éxito entre los seguidores de Orion y de otros scripts.

## ▪ Problema en la encriptación

Cuando ya la fama de este sistema se extendió, el creador del IRCap, tuvo la fenomenal idea de coger el fichero de texto del Orión en el que se apoyaban todas las sustituciones de caracteres (A=\*, B=&, C=\$, etc). Y con ello se sacó una utilidad denominada "Descodificación del Orión". Una vez hecho esto claro la criptografía había sido vulnerada, ya dejó de ser criptografía.

### Soluciones

*El cambio de caracteres:* Más trabajo para mí y para él si tenía ganas de volver a repetir la jugada.

*El cambio de clave:* Lo mismo, él podría hacerse con la clave.

*Ocultar o codificar el fichero:* El fichero de texto es abierto y para interactuar con él desde el script es necesario que sea ini y abierto al público.

*No incluir el fichero con las sustituciones en la distribución del programa:* Tampoco era posible.

Después de darle muchas, muchas y muchas más vueltas, tuve una muyyyyy buena idea, y era basarme en la teoría de las webs dinámicas.

### Solución definitiva

El Orión cada vez que conecta al IRC internamente a través de sockets (conexiones) hace una llamada al servidor donde tengo alojado el programa para su descarga ([www.ircorion.net](http://www.ircorion.net)).

En esa llamada el socket o conexión busca un fichero llamado ldecode.ini y mira su fecha. Y se hace la pregunta ¿La fecha del ldecode.ini del server es superior a la fecha del ldecode.ini que tengo en mi directorio del Orión local?, en caso de que lo sea descarga ese fichero y lo carga, cargando así una serie de nuevos caracteres.

El fichero del servidor al que solo puedo acceder yo y cuando se necesite subo ldecode.ini actualizado y se que automáticamente todos los usuarios del Orión a partir de ese momento de forma interna para ellos se cargaran la nueva configuración.

Con esta idea IRCap ha quitado su empeño en decodificar mensajes ocultos del Orión.

A continuación se intentará explicar detalladamente el código programado de un sistema de encriptación que he desarrollado para Orión Script.

La idea de explicarlo, tiene como finalidad la de detallar aún más este trabajo, no limitándolo así a simples conceptos teóricos. Sino conjuntarlo con conceptos prácticos sobre el tema. Además dicha práctica es totalmente real, lo que permitirá adentrarnos más en este mundo en el que la teoría en ocasiones queda muy por delante de la práctica.

La idea de documentar un trozo de código que realiza una función en principio me pareció que sería complicado para el lector, pero al final como se ha programado bajo un lenguaje "script para mIRC" (lenguaje relativamente fácil) puede no resultar tan engorrosa.

Antes de presentar el código de la utilidad de encriptación. Haré una muyyy breve introducción al lenguaje "scripting para mIRC", viendo 5 campos importantes para la comprensión del código.

## ▪ Introducción al scripting

1. Variables
2. Identificadores
3. Sentencias condicionales
4. Bucles
5. Eventos

#### ▪ Variables

Es un tipo de dato que referencia a un valor. Dicho mas fácilmente viene a ser como una palabra que funciona como un contenedor para almacenar un dato (reverenciarlo), luego dicho dato puede variar en el transcurso del programa.

Todas las variables en scripting

Una variable por ejemplo sería: `%canal`

Como podéis ver una variable esta precedida por el signo `%` el cual hace que el mIRC identifique ese elemento como variable y seguidamente el nombre de la variable (en el ejemplo anterior es `canal`) este puede contener cualquier carácter alfanumérico (aunque no es recomendable). También es preferible que se le aplique un nombre con relación a la función o contenido que va a tener dicha variable. El mIRC no aprecia diferenciación de las variables en mayúsculas y minúsculas, así pues la variable `%canal` es igual que la variable `%CaNaL`.

Para otorgar un valor a una variable se hará: `%numero = 5`

La variable `%numero` contiene el dato 5.

Si pusiéramos: `echo -s El numero es: %numero`

Nos mostraría en pantalla de estado "El numero es 5".

#### ▪ Identificadores

Es otro tipo de datos los cuales devuelven un valor de retorno, en base a unas operaciones realizadas con unos parámetros dados.

Los identificadores en scripting vendrían a ser las llamadas funciones en otros lenguajes vease C, por ejemplo.

Existe una importante diferencia con respecto a otros lenguajes, en C y otros muchísimos mas lenguajes se entiende por identificador al nombre que se le asigna a una variable o constante. En scripting hay que olvidarse de este concepto porque solo nos confundirá.

Al igual que en lenguaje C las funciones pueden estar ya creadas por otros programadores o puedes crearlas tu, lo mismo pasara con los identificadores.

Su formato es.

```
suma {  
  %result = $1 + $2  
  return %result  
}
```

`echo -s La suma del numero 4+3 es: $suma(4,3)`

El identificador es "\$suma" vemos como en la definición del identificador no se pone el signo correspondiente a el "\$". El valor de retorno de dicho identificador es `%result` que a su vez es la suma del primer y segundo parámetro posicional pasado al identificador.

Los identificadores "\$1, \$2, \$3, etc" están ya programados y se cargan solos con los parámetros.

A la hora de llamar a un identificador si es necesario poner el signo "\$" y pasar los parámetros agrupados entre paréntesis vease el ejemplo: `$suma(4,3)`.

En scripting para poner varias sentencias en una misma línea se usa el carácter "|" (altgr+1). Ejemplo:

`%numero = 6 | %letra = a`

### ▪ Sentencias condicionales

Funcionan exactamente igual que en cualquier otro lenguaje.

Su formato es:

```
If (condicion) { sentencias }  
Else { sentencias }
```

La traducción de esto sería: Si (if) se cumple la condición expuesta entre paréntesis hacemos las sentencias agrupadas por llaves, en caso contrario (else) realizamos estas otras sentencias agrupadas por llaves. Recordemos que las sentencias en scripting son órdenes y estas no finalizan con “;”.

Ejemplo:

```
%numero1 = 5  
%numero2 = 6  
if (%numero1 > %numero2) { echo -s Numero1 mayor que Numero2 }  
else { echo -s Numero2 mayor que Numero1 }
```

Este ejemplo mostraría en pantalla: Numero2 mayor que Numero1

### ▪ Bucles

Veremos un caso de el bucle usado en el código, este es el while.

Totalmente igual que en lenguaje C. Es el típico bucle iterativo.

Formato: while (condicion) { sentencias }

Su traducción: Mientras (while) se cumpla condición, hacemos sentencias.

Ejemplo:

```
%numero = 0  
while (%numero < 11) {  
  %numero = %numero + 1  
  echo -s %numero  
}
```

Este ejemplo mostraría los números del 1 al 10.

### ▪ Eventos

En los lenguajes orientados a objetos u orientados a eventos, los eventos son sucesos que se producen o se dan a lo largo de una sesión de trabajo con el programa. Mediante la programación de eventos podemos capturar esos sucesos y actuar ante ellos de una forma especial, con lo que estamos predisponiendo el programa a actuar de formas en base a sucesos.

Un suceso por ejemplo puede ser al clicar el botón izq del ratón, esto sería capturado con el evento on click, todos los eventos en scripting se preceden de la palabra on como en otros lenguajes.

Ejemplo

```
on 1:input:*.:{  
  echo -s he pulsado el botón intro  
}
```

Mediante este evento capturamos el suceso correspondiente a la pulsación de la tecla “Enter” y cuando lo hacemos se muestra en pantalla “he pulsado el botón intro”.

Visto esta introducción abordaremos el código, dicho código será comentado paso a paso para que el lector no pierda la secuencia y el razonamiento.

## ▪ Código de encriptación Orion Script

El código de encriptación de Orion Script es del tipo encriptación por sustitución, su funcionamiento se basa en la sustitución de los caracteres por otros que hagan del mensaje un texto inteligible. A cada carácter legible decodificado (a, b, c) le corresponde un carácter de codificación inteligible (% , & , /).

El sistema se apoya en un código y un sistema de información basado en ficheros.

Esto es, que el código interactúa con un fichero que es el que tiene los caracteres de codificación y decodificación del mensaje.

El sistema de encriptación además es dinámico, lo que quiere decir que se actualiza cuando se desee de forma inapreciable para el usuario final, evitando con esto la copia de caracteres y creaciones de sistemas de decodificación.

**Como ya se dijo el sistema de encriptación del Orion consta de dos partes: código, y fichero.**

### Fichero

Nombre del Fichero: ldecode

Extensión: .ini

Tamaño: 309 bytes

Contenido:

[fecha]

fecha=16/10/2001

[chars]

clave='!&@€

0 = <

1 = f

2 = t

3 = r

4 = k

5 = v

6 = d

7 = o

8 = p

9 = w

q = y

w = H

e = \_

r = \

t = !

y = &

u = (

i = 0

o = 8

p = a

a = 7

s = )

d = '  
f = ]  
g = 4  
h = 1  
j = i  
k = 5  
l = >  
ñ = 3  
z = °  
x = 6  
c = #  
v = [  
b = ¬  
n = @  
m = x

NOTA: El sistema de codificación del Orión actual solo tiene cargados estos caracteres de sustitución, en caso de introducir uno que no se encuentra por un sustituto, este se escribe tal cual en el mensaje codificado resultante final.

De todas formas la lista se puede aumentar cuando quiera.

- **Manejo de ficheros ini en scripting**

Estos ficheros son procesados en scripting para el manejo de lo que digamos podríamos entender como variables pero apoyadas en un fichero como soporte de información.

El fichero ini se estructura en

- Grupos: Agrupan a una serie de variables. Ejp: [fecha]
- Variables: Son las variables pertenecientes a un grupo. Ejp: fecha=16/10/2001

- **Código**

En primer lugar expongo el tratamiento de la fecha, para otorgar capacidad dinámica al sistema de codificación, no es muy importante entender esta parte puesto, que no va referida directamente con el tema de encriptación. Es por ello que no explico detalladamente ninguna de sus líneas.

```
=====
on 1:connect:{ onlineldecode }

onlineldecode {
  sockclose ldecode
  .timerlcode 1 1 sockopen ldecode www.ircorion.net 80
}

on 1:sockopen:ldecode:{
  if ($sockerr > 0) { halt }
  else { sockwrite -tn ldecode GET /script/ldecode.ini }
}

on 1:sockread:ldecode:{
  sockread %ldecode
  if (fecha= isin %ldecode) && ($gettok(%ldecode,2,61) != $date) {
    sockclose ldecode | .timer 1 1 .remove sistema\ldecode\tldecode.ini halt
  }
  if (ldecodeEnd isin %ldecode) {
    sockclose ldecode
    .remove sistema\ldecode\ldecode.ini | .rename sistema\ldecode\tldecode.ini sistema\ldecode\ldecode.ini
    .timer 1 1 .remove sistema\ldecode\tldecode.ini
  }
  write -a sistema\ldecode\tldecode.ini %ldecode
}
=====
```

**Explicación general:** Mediante el evento on connect se captura cuando el usuario emprende la acción de conectar. Cuando realiza esto quiere decir que ya se encuentra en internet, por lo cual hago una llamada al alias o función/identificador sin retorno (onlineldecode), esta función cierra una posible conexión del tipo ldecode en caso de existir para no provocar errores, y en caso contrario, lanza un sock mediante sockopen al sitio [www.ircorion.net](http://www.ircorion.net), el servidor indicamos puerto 80 para poder leer el formato html aunque el file sea de texto así podremos leer su contenido y ver la fecha.

Mediante sockread vamos tomando el fichero y comprobamos en la condicional si la fecha del fichero ldecode.ini que tiene el usuario que conecto es inferior a la fecha del fichero ldecode.ini del servidor en caso de ser así procedemos a la descarga, y el borrado del viejo fichero. De no ser así conserva el mismo fichero y corta la conexión.

Todo este proceso se ha cronometrado, aproximadamente dura entre 2-4 seg. Todo ello es inapreciable por el usuario final. Si sabiendo cuando se esta actualizando y cuando no.

**El resto del código**

---

```

on 1:input:~*:{
  if (%vartypeletra == code) { %q = $typeletra(%vartypeletra,$1-) }
  .msg $active %q
  halt
}

on ^1:text:~*#:{
  if ($gettok($1-,2,32) == $readini(sistema\decode\decode.ini,chars,clave)) {
    %q = $typeletra(decode,$gettok($1-,3-,32))
  }
  haltdef
}

typeletra {
  if ($1 == code) {
    %a = 1
    %result = OrioNCode: $readini(sistema\decode\decode.ini,chars,clave) $chr(32)
    while ($mid($2,%a,1) != $null) {
      if ($mid($2,%a,1) == $chr(32)) {
        %result = %result $chr(32) | inc %a | continue
      }
      if ($read(sistema\decode\decode.ini,nw,$mid($2,%a,1) *) == *) {
        %result = %result $+ $remove($gettok($read(sistema\decode\decode.ini,nw,$mid($2,%a,1) *)*),2,61),$chr(32))
        inc %a
        continue
      }
      else { %result = %result $+ $mid($2,%a,1) }
      inc %a
    }
    return %result
  }
  if ($1 == decode) {
    %a = 1 | %result = 14OrioN D 14ecode: 1 $chr(32)
    while ($mid($2,%a,1) != $null) {
      if ($mid($2,%a,1) == $chr(32)) { %result = %result $chr(32) }
      if ($read(sistema\decode\decode.ini,nw,* = $mid($2,%a,1))) {
        %result = %result $+ $remove($gettok($read(sistema\decode\decode.ini,nw,* = $mid($2,%a,1)),1,61),$chr(32))
        inc %a
        continue
      }
      else { %result = %result $+ $mid($2,%a,1) | inc %a | continue }
      inc %a
    }
    return %result
  }
}

```

---

- **Explicacion del codigo**

Primero empezaremos explicando la codificación.

El usuario introduce un mensaje racional y perfectamente legible en el campo de edición de la ventana de canal del IRC. Para mandarlo es necesario que presione la tecla enter, como en todos los sistemas de IRC. Capturamos el suceso con `on 1:input:*;`, si la variable `%vartypeletra` tiene como valor "code" es que el usuario ha seleccionado en el Orión el tipo de letra codificada. Entonces se le da a la variable `%q` un valor de retorno que devuelve el identificador `$typeletra(%vartypeletra,$1-)`, al cual si nos fijamos le pasa dos parámetros que es `code` (la acción a realizar el tipo de letra codificada) y `$1-` que sería el texto que contendría el texto introducido por el usuario cuando pulso enter. Si por ejemplo escribí: `Hola soy Javi <enter>`

El identificador tendría estos valores en sus parámetros `$typeletra(code,Hola soy Javi)`

La llamada es como ya dijimos a la función de `typeletra` pasándole los dos parámetros mencionados. Dentro de esta función "typeletra" el primer paso es verificar a través de una condicional si el primer parámetro pasado a la función "`$1`" es igual a "code", como lo es pasa a proceder las siguientes sentencias.

Creamos una variable que funcionara como contador "`%a`", y la inicializamos a 1, creamos una variable `%result` que contendrá el resultado final de toda la codificación, le damos inicialmente el valor "`OriónCode: $readini(sistema\decode\decode.ini,chars,clave) $chr(32)`", esto es:

OriónCode simple texto llano.

`$readini(fichero,grupo,variable)`: lee fichero ini, concretamente las variables de este, en este caso lee pasamos como primer parámetro a `$readini` el fichero "sistema\decode\decode.ini" que es la ruta de localización del fichero mencionado anteriormente que contiene los caracteres de sustitución, la clave, y fecha; como segundo parámetro le pasamos el grupo donde se encuentra la variable que queremos obtener su valor de retorno en este caso será "chars", y como tercer parámetro pasamos el nombre de la variable "clave", `$readini` nos devolverá la variable "clave" del grupo "chars" del fichero "decode.ini" que es: `!&@€`. Después de esto pone "`$chr(32)`" este identificador devuelve el carácter ASCII correspondiente al número 32 (o sea un espacio).

Con lo que la variable `%result` queda inicializada a:

"OriónCode: !&@€ "

La clave ya está cargada en la variable, a continuación se sitúan los siguientes caracteres codificados. Para ir recopilando todos los caracteres expuestos en la frase y decodificarlos se procede con un `while` bucle iterativo "`while ($mid($2,%a,1) != $null)`", la condición es: `$mid` devuelve 1 carácter (parámetro 3) desde la posición "`%a`" de la cadena de caracteres "`$2`", que era toda la frase pasada a la función, en caso de que esto "`$mid`" devuelva un valor quiere decir que hay letra en la frase y que no es igual a nullo "`$null`" por lo tanto se entra en el bucle, la secuencia es: `$mid($2,1,1)`, `$mid($2,2,1)`, `$mid($2,3,1)`, `$mid($2,4,1)`. De esta forma se va extrayendo letra a letra todas las letras de la frase.

Al entrar en el bucle

Lo primero que se hace es comprobar si el carácter que está analizando `$mid` en la iteración es igual a un espacio. "`if ($mid($2,%a,1) == $chr(32))`" En caso de serlo se respeta ese espacio en la codificación y se pone textualmente: `%result = %result $chr(32) | inc %a | continúe`

En la variable `%result` metemos siempre el anterior contenido de la variable `%result` para no perderlo y a continuación metemos el espacio `$chr(32)`, luego incrementamos el valor de la variable `%a` y con `continúe` salimos a la siguiente iteración.

Si no es un espacio el carácter analizado pasa al siguiente `if` que cuestiona si el carácter analizado se encuentra dentro del fichero `decode.ini`

```

if ($read(sistema\decode\decode.ini,nw,$mid($2,%a,1) *=*)) {
%result = %result $+ $remove($gettok($read(sistema\decode\decode.ini,nw,$mid($2,%a,1) *=*),2,61),$chr(32))
inc %a
continue
}

```

Supongamos que la compleja estructura de identificadores `$read(sistema\decode\decode.ini,nw,$mid($2,%a,1) *=*)` en caso de retornar un valor retorna el valor que corresponde al carácter que estamos analizando ya sea la letra a, b o c y retorna su sustituto en caso de la letra a será "7". Y suponiendo que se cumpla y si devuelva un valor de retorno añadiríamos a la variable `%result`, el contenido de si misma (para no perderlo) mas el carácter de codificación. De esta forma iríamos acumulando todos los caracteres de codificación de correspondientes a cada letra de la frase.

A continuación se plantea el caso de que el carácter ni sea un espacio ni sea un carácter recogido en `ldecode.ini` ante esto entra en el "else": `else { %result = %result $+ $mid($2,%a,1) }` Y carga la variable `%result` con sus valores codificados anteriores y el mismo carácter que esta analizando y que no esta en el fichero `ldecode.ini` simplemente lo pone de forma textual dentro de toda la codificación.

En la línea posterior procedemos al incremento de la variable `%a` para acudir a analizar el siguiente carácter.

Cuando salimos del bucle quiere decir que no nos quedan caracteres de la frase por recorrer entonces es el turno de devolver la variable `%result` con todo el mensaje de codificación.

Si volvemos a recordar:

```

on 1:input:*. {
if (%vartypeletra == code) { %q = $typeletra(%vartypeletra,$1-) }
.msg $active %q
halt
}

```

`$typeletra` devuelve el `%result` que contenía toda la codificación, y lo mete en la variable `%q`. A continuación mediante el comando `msg` lanzamos al canal activo "\$active" la codificación de toda la frase "%q".

En cuanto a la descodificación (de forma ya resumida)

```

on ^1:text:*:#{
  if ($gettok($1-,2,32) == $readini(sistema\decode\decode.ini,chars,clave)) {
    %q = $typeletra(decode,$gettok($1-,3-,32))
  }
  haltdef
}

```

capturamos todos los textos que aparecen en el canal con el evento “on ^1:text:\*:#:”, luego en la condicional observamos si en el texto aparece la clave en caso de ser así, se procede al método de decodificación, que es exactamente igual pero sustituyendo a los caracteres a la inversa.

### Imágenes de la encriptación en acción

Codificacion
<pre> (Quasi) OrioNCode: !&amp;@€18&gt;7 1(48 (Quasi) OrioNCode: !&amp;@€ y(_ !7&gt; _)!7) (Quasi) OrioNCode: !&amp;@€ (@8)_0 )_!07 ~(_@7 0!_7 x!_) _@ _&gt; !7~7!8 &gt;7 '8#(x_@!7#08@'_ (@ #8'048, #8x_@!7\ &amp; _6^&gt;0#7\ #8x8 ](@#08@7 (@ #8'048 x_ \_)(&gt;!8 (@ !7@!8 #8x^&gt;0#7'8 </pre>

De-Codificacion
<pre> (Quasi) OrioNDecode: hola hugo (Quasi) OrioNDecode: que tal estas (Quasi) OrioNDecode: nose si seria buena idea meter en el trabajo la documentacion de un codigo, comentar 0 explicar como funciona un codigo me resulto un tanto complicado </pre>