

## REQUERIMIENTOS HARDWARE



Sistema base + 256 MB Ram + 10Gb Disco Duro.

## REQUERIMIENTOS SOFTWARE

VirtualBox3.2.X	<a href="http://www.virtualbox.org/">http://www.virtualbox.org/</a>
Linux Ubuntu 10.04Lts 32bits	<a href="http://www.ubuntu.com/desktop/get-ubuntu/download">http://www.ubuntu.com/desktop/get-ubuntu/download</a>
Dionaea	<a href="http://dionaea.carnivore.it/">http://dionaea.carnivore.it/</a>
Kippo	<a href="http://code.google.com/p/kippo/">http://code.google.com/p/kippo/</a>

## INSTALACION DEL VIRTUALBOX3

<http://www.virtualbox.org/manual/ch02.html>

## INTALACION DE UBUNTU

[http://www.guia-ubuntu.org/index.php?title=Instalaci%C3%B3n\\_est%C3%A1ndar](http://www.guia-ubuntu.org/index.php?title=Instalaci%C3%B3n_est%C3%A1ndar)

## CONSTRUCCION DE LA MAQUINA VIRTUAL

### AJUSTES DEL SISTEMA VIRTUAL

#### Desactivación del servicio dhcp

Al configurar la conexión con ip estática de desactiva.

#### Desactivación del servicio cups

```
~$ sudo update-rc.d -f cups remove
```

#### Deinstalación (No funciona desactivación) del servicio avahi-daemon

```
~$ sudo update-rc.d -f avahi-daemon remove (NO FUNCIONA)
~$ sudo apt-get remove avahi-daemon (SOLUCION)
```

#### Desinstalación de las herramientas multimedia, ofimática y juegos

Nos vamos a “Aplicaciones”->”Centro de software de Ubuntu” y desde ahí las quitamos una por una.

## INSTALACION DEL HONEYPOT DIONAEA (22/08/2011)

### #DEPENDENCIAS



```
~# sudo apt-get install libudns-dev libglib2.0-dev libssl-dev libcurl4-openssl-dev  
libreadline-dev libsqlite3-dev python-dev libtool automake autoconf build-essential  
subversion git-core flex bison pkg-config
```

### #POF (OPCIONAL)



```
~# sudo apt-get install p0f
```

### #OPENSSL (OPCIONAL)



```
~# sudo apt-get install cvs  
~# cvs -d anonymous@cvs.openssl.org:/openssl-cvs co openssl  
~# cd openssl  
~openssl# ./Configure shared --prefix=/opt/dionaea linux-x86_64 ERROR  
~openssl# ./config SOLUCION  
~openssl# make SHARED LDFLAGS=-Wl,-rpath,/opt/dionaea/lib  
~openssl# sudo make install  
~openssl# cd ..
```

### #LIBLCFG



```
~# git clone git://git.carnivore.it/liblcfg.git liblcfg  
~# cd liblcfg/code  
~liblcfg/code# autoreconf -vi  
~liblcfg/code# ./configure --prefix=/opt/dionaea  
~liblcfg/code# make install  
~liblcfg/code# cd ..  
~liblcfg# cd ..
```

### #LIBEMU



```
~# git clone git://git.carnivore.it/libemu.git libemu  
~# cd libemu  
~libemu# autoreconf -vi  
~libemu# ./configure --prefix=/opt/dionaea  
~libemu# make install  
~libemu# cd ..
```

## #LIBNL (OPCIONAL)



```
~# git clone git://git.kernel.org/pub/scm/libs/netlink/libnl.git
~# cd libnl
~libnl# autoreconf -vi
~libnl# export LDFLAGS=-Wl,-rpath,/opt/dionaea/lib
~libnl# ./configure --prefix=/opt/dionaea
~libnl# make
~libnl# make install
~libnl# cd ..
```

## #LIBEV



```
~# wget http://dist.schmorp.de/libev/libev-4.04.tar.gz
~# tar xzf libev-4.04.tar.gz
~# cd libev-4.04
~libev-4.04# ./configure --prefix=/opt/dionaea
~libev-4.04# make install
~libev-4.04# cd ..
```

## #PYTHON 3.2



```
~# wget http://python.org/ftp/python/3.2/Python-3.2.tgz
~# tar xzf Python-3.2.tgz
~# cd Python-3.2/
~Python-3.2# ./configure --enable-shared --prefix=/opt/dionaea --with-computed-gotos --
enable-ipv6 LDFLAGS="-Wl,-rpath=/opt/dionaea/lib/"
~Python-3.2# make
~Python-3.2# make install
~Python-3.2# cd ..
```

## #CYTHON



```
~# git clone https://github.com/cython/cython.git
~# cd cython
~cython# /opt/dionaea/bin/python3 setup.py install
~cython# python setup.py build
~cython# sudo python setup.py install
~cython# cd ..
```

## #LXML Y DEPENDENCIAS (RECOMENDADO)



```
~# apt-get install libxml2-dev libxslt1-dev
~# wget http://codespeak.net/lxml/lxml-2.3.tgz
~# tar xzf lxml-2.3.tgz
~# cd lxml-2.3
~lxml-2.3# /opt/dionaea/bin/python3 setup.py install
~lxml-2.3# cd ..
```

## #C-ARES



```
~# wget http://c-ares.haxx.se/c-ares-1.7.3.tar.gz
~# tar xzf c-ares-1.7.3.tar.gz
~# cd c-ares-1.7.3
~c-ares-1.7.3# ./configure --prefix=/opt/dionaea
~c-ares-1.7.3# make
~c-ares-1.7.3# make install
~c-ares-1.7.3# cd ..
```

## #CURL



```
~# wget http://curl.haxx.se/download/curl-7.20.0.tar.bz2
~# tar xjf curl-7.20.0.tar.bz2
~# cd curl-7.20.0
~curl-7.20.0# ./configure --prefix=/opt/dionaea --enable-ares=/opt/dionaea
~curl-7.20.0# make
~curl-7.20.0# make install
~curl-7.20.0# cd ..
```

## #LIBPCAP



```
~# wget http://www.tcpdump.org/release/libpcap-1.1.1.tar.gz
~# tar xzf libpcap-1.1.1.tar.gz
~# cd libpcap-1.1.1
~libpcap-1.1.1# ./configure --prefix=/opt/dionaea
~libpcap-1.1.1# make
~libpcap-1.1.1# make install
~libpcap-1.1.1# cd ..
```

## #UDNS



```
~# apt-get install libudns0 libudns-dev
```

## #DIONAEA



```
~# git clone git://git.carnivore.it/dionaea.git dionaea
~# autoreconf -vi
~# cd dionaea/
~dionaea# autoreconf -vi
~dionaea# ./configure --with-lcfg-include=/opt/dionaea/include/ \
--with-lcfg-lib=/opt/dionaea/lib/ \
--with-python=/opt/dionaea/bin/python3.2 \
--with-cython-dir=/opt/dionaea/bin \
--with-udns-include=/usr/include/ \
--with-udns-lib=/usr/lib/ \
--with-emu-include=/opt/dionaea/include/ \
--with-emu-lib=/opt/dionaea/lib/ \
--with-gc-include=/usr/include/gc \
--with-ev-include=/opt/dionaea/include \
--with-ev-lib=/opt/dionaea/lib \
--with-nl-include=/opt/dionaea/include \
--with-nl-lib=/opt/dionaea/lib/ \
--with-curl-config=/opt/dionaea/bin/ \
--with-pcap-include=/opt/dionaea/include \
--with-pcap-lib=/opt/dionaea/lib/ \
--with-glib=/opt/dionaea
~dionaea# make
~dionaea# make install
```

## SCRIPT DE INSTALACION AUTOMATICA DE DIONAEA

```
#!/bin/bash

if [ $? != 0 ]; then
    exit
fi

#DEPENDENCIAS
apt-get install libudns-dev \
libglib2.0-dev \
libssl-dev \
libcurl4-openssl-dev \
libreadline-dev \
libsqlite3-dev \
python-dev \
libtool \
automake \
autoconf \
build-essential \
subversion \
git-core \
flex \
bison \
pkg-config \
gettext

### sqlite3
apt-get install sqlite3

### p0f
apt-get install p0f

if [ ! -e "/opt/dionaea" ]; then
    mkdir /opt/dionaea
fi

if [ ! -e "dionaea" ]; then
    mkdir dionaea
fi
cd dionaea
MYPWD=`pwd`

### openssl (OPCIONAL)
if [ ! -e "openssl" ]; then
    apt-get install cvs
    cvs -d anonymous@cvs.openssl.org:/openssl-cvs co openssl
    cd openssl
    ./config
    make SHARED_LDFLAGS=-Wl,-rpath,/opt/dionaea/lib
    make install
    cd $MYPWD
fi

### liblcfg
if [ ! -e "liblcfg" ]; then
    git clone git://git.carnivore.it/liblcfg.git liblcfg
    cd liblcfg/code
else
    cd liblcfg/code
    git pull
fi
autoreconf -vi
./configure --prefix=/opt/dionaea
make install
cd $MYPWD

### libemu
if [ ! -e "libemu" ]; then
    git clone git://git.carnivore.it/libemu.git libemu
    cd libemu
else
    cd libemu
    git pull
fi
autoreconf -vi
./configure --prefix=/opt/dionaea
```

```

make install
cd $MYPWD

### libnl (OPCIONAL)
if [ ! -e "libnl" ]; then
    git clone git://git.kernel.org/pub/scm/libs/netlink/libnl.git
    cd libnl
else
    cd libnl
    git pull
fi
autoreconf -vi
export LDFLAGS=-Wl,-rpath,/opt/dionaea/lib
./configure --prefix=/opt/dionaea
make
make install
cd $MYPWD

### libev
if [ ! -e "libev-4.04" ]; then
    wget http://dist.schmorp.de/libev/libev-4.04.tar.gz
    tar xzf libev-4.04.tar.gz
    cd libev-4.04
    ./configure --prefix=/opt/dionaea
    make install
    cd $MYPWD
fi

### python3
if [ ! -e "Python-3.2" ]; then
    wget http://python.org/ftp/python/3.2/Python-3.2.tgz
    tar xzf Python-3.2.tgz
    cd Python-3.2/
    ./configure --enable-shared --prefix=/opt/dionaea --with-computed-gotos \
        --enable-ipv6 LDFLAGS="-Wl,-rpath=/opt/dionaea/lib/"
    make
    make install
    cd $MYPWD
fi

### cython
if [ ! -e "cython" ]; then
    git clone https://github.com/cython/cython.git
    cd cython
else
    cd cython
    git pull
fi
/opt/dionaea/bin/python3 setup.py install
cd $MYPWD

### lxml
if [ ! -e "lxml-2.3" ]; then
    ### DEPENDENCIAS libxml2 libxslt
    apt-get install libxml2-dev libxslt1-dev

    wget http://lxml.de/files/lxml-2.3.tgz
    tar xzf lxml-2.3.tgz
    cd lxml-2.3/
    /opt/dionaea/bin/python3 setup.py install
    cd $MYPWD
fi

### c-ares (OBSOLETO? NECESARIO PARA CURL)
if [ ! -e "c-ares-1.7.5" ]; then
    wget http://c-ares.haxx.se/download/c-ares-1.7.5.tar.gz
    tar xzf c-ares-1.7.5.tar.gz
    cd c-ares-1.7.5
    ./configure --prefix=/opt/dionaea
    make
    make install
    cd $MYPWD
fi

### curl
if [ ! -e "curl-7.20.0" ]; then
    wget http://curl.haxx.se/download/curl-7.20.0.tar.bz2
    tar xjf curl-7.20.0.tar.bz2

```

```

        cd curl-7.20.0
        ./configure --prefix=/opt/dionaea --enable-ares=/opt/dionaea
        make
        make install
        cd $MYPWD
    fi

### libpcap
if [ ! -e "libpcap-1.1.1" ]; then
    wget http://www.tcpdump.org/release/libpcap-1.1.1.tar.gz
    tar xzf libpcap-1.1.1.tar.gz
    cd libpcap-1.1.1
    ./configure --prefix=/opt/dionaea
    make
    make install
    cd $MYPWD
fi

### udns
###(OPCIONAL NO USAR EN DEBIAN - INSTALAR libudns0_0.0.9-3_i386.deb Y libudns-dev_0.0.9-3_i386.deb)
###(OPCIONAL NO USAR EN UBUNTU)
apt-get install libudns0 libudns-dev
#if [ ! -e "udns-0.0.9" ]; then
#    wget http://www.corpit.ru/mjt/udns/old/udns_0.0.9.tar.gz
#    tar xzf udns_0.0.9.tar.gz
#    cd udns-0.0.9/
#    ./configure
#    make shared
#    cd $MYPWD
#fi

### dionaea
###(CAMBIAR PARA DEBIAN Y UBUNTU)
###--with-udns-include=/opt/dionaea/include/ \
###--with-udns-lib=/opt/dionaea/lib/ \
### POR
###--with-udns-include=/usr/include/ \
###--with-udns-lib=/usr/lib/ \#
git clone git://git.carnivore.it/dionaea.git dionaea
cd dionaea
autoreconf -vi
./configure --with-lcfg-include=/opt/dionaea/include/ \
--with-lcfg-lib=/opt/dionaea/lib/ \
--with-python=/opt/dionaea/bin/python3.2 \
--with-cython-dir=/opt/dionaea/bin \
--with-udns-include=/usr/include/ \
--with-udns-lib=/usr/lib/ \
--with-emu-include=/opt/dionaea/include/ \
--with-emu-lib=/opt/dionaea/lib/ \
--with-gc-include=/usr/include/gc \
--with-ev-include=/opt/dionaea/include \
--with-ev-lib=/opt/dionaea/lib \
--with-nl-include=/opt/dionaea/include \
--with-nl-lib=/opt/dionaea/lib/ \
--with-curl-config=/opt/dionaea/bin/ \
--with-pcap-include=/opt/dionaea/include \
--with-pcap-lib=/opt/dionaea/lib/ \
--with-glib=/opt/dionaea
make
make install
cd $MYPWD


```




## CONFIGURACION DE DIONAEA

### Configuración de inicio como demonio del sistema durante el arranque:

#### Creamos el script de arranque.


```
 ~$ cd /etc/init.d  
/etc/init.d/$ sudo nano dionaea.sh
```

#### Escribimos el comando de inicio.

```
 #!/bin/sh  
#Carga del proceso p0f (detección del S.O que se nos conecta al honeypot)  
p0f -i any -u root -Q /tmp/p0f.sock -q -l -d -o /tmp/p0f.log  
  
#OJO CON -u nobody NO FUNCIONA EN RASPBERRY PI, PC POR CONFIRMAR  
#/opt/dionaea/bin/dionaea -D -u nobody -g nogroup -r /opt/dionaea -w /opt/dionaea -p  
#/opt/dionaea/var/dionaea.pid  
  
/opt/dionaea/bin/dionaea -D -u root -r /opt/dionaea -w /opt/dionaea -p  
/opt/dionaea/var/dionaea.pid
```

Ctrl+x para salir preguntándonos si queremos guardar.


#### Le damos permiso de ejecución.

```
 /etc/init.d/$ sudo chmod 744 dionaea.sh
```

#### Creamos los enlaces automáticamente a cada uno de los directorios para los runcommands (/etc/rcX.d) de los distintos modos de arranque.

Por defecto se nos crearán como activados S y prioridad 20 durante el arranque.

S20dionaea.sh → ../init.d/dionaea.sh

```
 /etc/init.d/$ sudo update-rc.d dionaea.sh defaults
```

## Creamos el archivo /opt/zetsuBD/zetsu.sqlite y damos de alta al administrador.



```
CREATE TABLE "login" ("username" VARCHAR PRIMARY KEY NOT NULL , "password" VARCHAR NOT NULL , "rol" VARCHAR NOT NULL )
```

```
INSERT INTO "main"."login" ("username","password","rol") VALUES (?1,?2,?3)
```

Parameters:

param 1 (text): zetsu

param 2 (text):

15e907c0e30958e3882bcd552b4216106f8c52f2cb767be788c446664aaf20d5de6019148cf2d70001f618cb17cd  
4399fbfb7c7257a9c05ea00ca55e25cbeb9

param 3 (text): administrador

## Modificación de permisos de dionaea:

/opt/dionaea/var/dionaea/logsql.sqlite (Base de datos de intentos de conexión)

/opt/dionaea/var/dionaea/zetsu.sqlite (Base de datos de Usuarios)

/opt/dionaea/etc/dionaea/dionaea.conf (Configuración de dionaea)



```
$ sudo chown root:www-data /opt/dionaea/var/dionaea
$ sudo chown root:www-data /opt/dionaea/var/dionaea/logsql.sqlite
/opt/dionaea/var/dionaea/sipaccounts.sqlite /opt/dionaea/var/dionaea/vtcache.sqlite
$ sudo chown root:www-data /opt/dionaea/etc/dionaea/dionaea.conf
$ sudo chmod 664 /opt/dionaea/etc/dionaea/dionaea.conf
$ sudo chown root:www-data /opt/zetsuBD/
$ sudo chmod 664 /opt/zetsuBD/zetsu.sqlite
```

## INSTALACION DEL HONEYPOT KIPPO

### #DEPENDENCIAS



```
~# sudo apt-get install python-twisted
```

### Instalación.



```
~$ sudo mkdir /opt/kippo
~$ sudo chown zetsu:zetsu /opt/kippo
~$ cd /opt/kippo
~/opt/kippo$ wget http://kippo.googlecode.com/files/kippo-0.5.tar.gz
~/opt/kippo$ tar -xvzf kippo*
~/opt/kippo$ mv kippo-0.5 /opt/kippo
```

## CONFIGURACION DE KIPPO

### Configuración de inicio en el arranque del sistema sistema:

Kippo no permite por razones de seguridad ser lanzado como usuario root. Por ello durante el arranque tendremos que cambiar de contexto del usuario root al usuario zetsu como se indica mas adelante.

### Creamos el script de arranque.



```
~$ cd /etc/init.d
/etc/init.d/$ sudo nano kippo.sh
```

Escribimos el comando de inicio. (Cambio de contexto a usuario zetsu)



```
#!/bin/sh
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 -j REDIRECT --to-port 2222
su - zetsu -c "rm /opt/kippo/kippo.pid"
su - zetsu -c "cd /opt/kippo;twistd -y /opt/kippo/kippo.tac -l /opt/kippo/log/kippo.log --
pidfile /opt/kippo/kippo.pid"
```

### Le damos permiso de ejecución.



```
/etc/init.d/$ sudo chmod 755 kippo.sh
```

## Modificación de permisos de kippo:

/opt/kippo/log/kippoBD.sqlite (Base de datos intentos de conexión)



```
~$ sudo chown -R zetsu:www-data /opt/kippo
```

**Creamos los enlaces automáticamente a cada uno de los directorios para los runcommands (/etc/rcX.d) de los distintos modos de arranque.**

Por defecto se nos crearán como activados S y prioridad 20 durante el arranque.

S20kippo.sh → ../init.d/kippo.sh



```
/etc/init.d/$ sudo update-rc.d kippo.sh defaults
```

# INSTALACION DE SERVIDOR APACHE Y MODULO PYTHON

## Instalación.



```
~$ sudo apt-get install apache2 libapache2-mod-python
```

## CONFIGURACION DEL SERVIDOR APACHE (HTTP)

### Cambiar el puerto de escucha de apache:



```
~$ sudo nano /etc/apache2/ports.conf
```

Modificamos las siguientes directivas dejándolas como sigue...



```
NameVirtualHost *:3650
Listen 3650
```

### Configuración del host virtual y activación del modulo python.



```
~$ sudo nano /etc/apache2/sites-available/zetsuHttp
```



Usamos la siguiente configuración.

```
<VirtualHost *:3650>

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www

    #Carga del modulo python
    LoadModule python module/modules/mod_python.so

    <Directory />
        #-FollowSymLinks No seguir enlaces simbolicos (seguridad)
        Options -FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/>
        #-Indexes No mostrar contenido del directorio (seguridad)
        Options -Indexes -FollowSymLinks MultiViews
        #No usamos .htaccess
        AllowOverride None
        #Manejador PSP de mod python
        AddHandler mod_python .psp
        PythonHandler mod_python.psp
        PythonDebug On
    </Directory>
    <Directory /var/www/imagenes>
        Options -Indexes -FollowSymLinks MultiViews
        AllowOverride None
    </Directory>
```

```
<Directory /var/www/js>
    Options -Indexes -FollowSymLinks MultiViews
    AllowOverride None
</Directory>

ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
<Directory "/usr/lib/cgi-bin">
    AllowOverride None
    Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    Order allow,deny
    Allow from all
</Directory>

ErrorLog /var/log/apache2/error.log

# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
LogLevel warn

CustomLog /var/log/apache2/access.log combined

Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    Allow from 127.0.0.0/255.0.0.0 ::1/128
</Directory>

</VirtualHost>
```

Activamos el sitio zetsuHttp.



```
~$ sudo a2ensite zetsuHttp
```


Finalmente reiniciamos el demonio apache2.



```
~$ sudo service apache2 restart
```

## CONFIGURACION DEL SERVIDOR APACHE (HTTP-S)

### Activamos el modulo ssl de apache:

```
 ~$ sudo a2enmod ssl
```

### Cambiar el puerto de escucha de apache:

```
 ~$ sudo nano /etc/apache2/ports.conf
```

Modificamos las siguientes directivas dejándolas como siguen.




```
#ZETSU HHTP
#NameVirtualHost *:3650
#Listen 3650

<IfModule mod_ssl.c>
    # If you add NameVirtualHost *:443 here, you will also have to change
    # the VirtualHost statement in /etc/apache2/sites-available/default-ssl
    # to <VirtualHost *:443>
    # Server Name Indication for SSL named virtual hosts is currently not
    # supported by MSIE on Windows XP.

    #ZETSU HTTPS
    NameVirtualHost *:4093
    Listen 4093
</IfModule>

<IfModule mod_gnutls.c>
    Listen 4093
</IfModule>
```

### Configuración del host virtual para ssl y activación del modulo python

```
 ~$ sudo nano /etc/apache2/sites-available/zetsuHttps
```

Usamos la siguiente configuración.



```
<IfModule mod_ssl.c>
<VirtualHost default :4093>

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www

    #Carga del modulo python
    LoadModule python_module modules/mod_python.so

    <Directory />
        #-FollowSymLinks No seguir enlaces simbolicos (seguridad)
        Options -FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/>
        #-Indexes No mostrar contenido del directorio (seguridad)
        Options -Indexes -FollowSymLinks MultiViews
```

```

        #No usamos .htaccess
        AllowOverride None
        #Manejador PSP de mod_python
        AddHandler mod_python .psp
        PythonHandler mod_python.psp
        PythonDebug On
    </Directory>
    <Directory /var/www/imagenes>
        Options -Indexes -FollowSymLinks MultiViews
        AllowOverride None
    </Directory>
    <Directory /var/www/js>
        Options -Indexes -FollowSymLinks MultiViews
        AllowOverride None
    </Directory>

    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>

    ErrorLog /var/log/apache2/error.log

    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog /var/log/apache2/access.log combined

    Alias /doc/ "/usr/share/doc/"
    <Directory "/usr/share/doc/">
        Options Indexes MultiViews FollowSymLinks
        AllowOverride None
        Order deny,allow
        Deny from all
        Allow from 127.0.0.0/255.0.0.0 ::1/128
    </Directory>

    # SSL Engine Switch:
    # Enable/Disable SSL for this virtual host.
    SSLEngine on

    # A self-signed (snakeoil) certificate can be created by installing
    # the ssl-cert package. See
    # /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
    # If both key and certificate are stored in the same file, only the
    # SSLCertificateFile directive is needed.
    SSLCertificateFile /etc/ssl/certs/zetsu.pem
    #SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

    # Server Certificate Chain:
    # Point SSLCertificateChainFile at a file containing the
    # concatenation of PEM encoded CA certificates which form the
    # certificate chain for the server certificate. Alternatively
    # the referenced file can be the same as SSLCertificateFile
    # when the CA certificates are directly appended to the server
    # certificate for convenience.
    #SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

    # Certificate Authority (CA):
    # Set the CA certificate verification path where to find CA
    # certificates for client authentication or alternatively one
    # huge file containing all of them (file must be PEM encoded)
    # Note: Inside SSLCACertificatePath you need hash symlinks
    #       to point to the certificate files. Use the provided
    #       Makefile to update the hash symlinks after changes.
    #SSLCACertificatePath /etc/ssl/certs/
    #SSLCACertificateFile /etc/apache2/ssl.crt/ca-bundle.crt

    # Certificate Revocation Lists (CRL):
    # Set the CA revocation path where to find CA CRLs for client
    # authentication or alternatively one huge file containing all
    # of them (file must be PEM encoded)
    # Note: Inside SSLCARevocationPath you need hash symlinks
    #       to point to the certificate files. Use the provided

```



```

#       Makefile to update the hash symlinks after changes.
#SSLCARevocationPath /etc/apache2/ssl.crl/
#SSLCARevocationFile /etc/apache2/ssl.crl/ca-bundle.crl

#   Client Authentication (Type):
#   Client certificate verification type and depth. Types are
#   none, optional, require and optional no ca. Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth 10

#   Access Control:
#   With SSLRequire you can do per-directory access control based
#   on arbitrary complex boolean expressions containing server
#   variable checks and other lookup directives. The syntax is a
#   mixture between C and Perl. See the mod ssl documentation
#   for more details.
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#               and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#               and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#               and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#               and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
#               or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>

#   SSL Engine Options:
#   Set various options for the SSL engine.
#   o FakeBasicAuth:
#       Translate the client X.509 into a Basic Authorisation. This means that
#       the standard Auth/DBMAuth methods can be used for access control. The
#       user name is the 'one line' version of the client's X.509 certificate.
#       Note that no password is obtained from the user. Every entry in the user
#       file needs this password: 'xxj31ZMTZzkVA'.
#   o ExportCertData:
#       This exports two additional environment variables: SSL_CLIENT_CERT and
#       SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#       server (always existing) and the client (only existing when client
#       authentication is used). This can be used to import the certificates
#       into CGI scripts.
#   o StdEnvVars:
#       This exports the standard SSL/TLS related 'SSL_*' environment variables.
#       Per default this exportation is switched off for performance reasons,
#       because the extraction step is an expensive operation and is usually
#       useless for serving static content. So one usually enables the
#       exportation for CGI and SSI requests only.
#   o StrictRequire:
#       This denies access when "SSLRequireSSL" or "SSLRequire" applied even
#       under a "Satisfy any" situation, i.e. when it applies access is denied
#       and no other module can change it.
#   o OptRenegotiate:
#       This enables optimized SSL connection renegotiation handling when SSL
#       directives are used in per-directory context.
SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

#   SSL Protocol Adjustments:
#   The safe and default but still SSL/TLS standard compliant shutdown
#   approach is that mod_ssl sends the close notify alert but doesn't wait for
#   the close notify alert from client. When you need a different shutdown
#   approach you can use one of the following variables:
#   o ssl-unclean-shutdown:
#       This forces an unclean shutdown when the connection is closed, i.e. no
#       SSL close notify alert is send or allowed to received. This violates
#       the SSL/TLS standard but is needed for some brain-dead browsers. Use
#       this when you receive I/O errors because of the standard approach where
#       mod_ssl sends the close notify alert.
#   o ssl-accurate-shutdown:
#       This forces an accurate shutdown when the connection is closed, i.e. a
#       SSL close notify alert is send and mod_ssl waits for the close notify
#       alert of the client. This is 100% SSL/TLS standard compliant, but in
#       practice often causes hanging connections with brain-dead browsers. Use

```

```
#      this only for browsers where you know that their SSL implementation
#      works correctly.
#      Notice: Most problems of broken clients are also related to the HTTP
#      keep-alive facility, so you usually additionally want to disable
#      keep-alive for those clients, too. Use variable "nokeepalive" for this.
#      Similarly, one has to force some clients to use HTTP/1.0 to workaround
#      their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
#      "force-response-1.0" for this.
BrowserMatch "MSIE [2-6]" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0
# MSIE 7 and newer should be able to use keepalive
BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>
</IfModule>
```

## Creamos el certificado para la sesión ssl.

Generamos una llave privada sin passphrase de 2048 bits (zetsu.key).

Generamos la solicitud de firma (Certificate Sign Request) (zetsu.csr). Rellenamos los datos solicitados.

Autofirmamos nuestro certificado con 10 años de tiempo de expiración.

Generamos el archivo pem uniendo clave privada y certificado.



```
~$ cd /etc/apache2
~$ sudo openssl genrsa -out zetsu.key 2048
~$ sudo openssl req -new -key zetsu.key -out zetsu.csr
~$ sudo openssl x509 -req -days 36500 -in zetsu.csr -signkey zetsu.key -out zetsu.crt
~$ sudo cat zetsu.key zetsu.crt > zetsu.pem
~$ sudo mv zetsu.* /etc/ssl/certs/
```

## Activamos el sitio zetsuHttps.



```
~$ sudo a2ensite zetsuHttp
```

## Finalmente reiniciamos el demonio apache2.



```
~$ sudo service apache2 restart
```

## CONFIGURACION DEL ENTORNO DE DESARROLLO Y USUARIOS

### Permisos en /var/www



```
~$ sudo chown root:www-data /var/www
```

**Agregamos el usuario zetsu al grupo www-data, para poder acceder a /var/www y desarrollar la aplicación.**



```
~$ sudo nano /etc/group
```

Modificamos la siguiente linea dejándola como sigue.



```
www-data:x:33:zetsu
```

### Permisos para el usuario www-data:

Al permitir la aplicación el cambio de la configuración de los servicios que se emulan, es necesario dar permisos a www-data para parar y relanzar el demonio de dionaea. Al ser este lanzado durante el arranque como root tenemos que darle permisos sudo y por seguridad limitarlo a tan solo a los scripts de parada e inicio.



```
~$ sudo nano /etc/sudoers
```

Modificamos la sección # User privilege specification como sigue.



```
# User privilege specification
root    ALL=(ALL) ALL
www-data ALL = NOPASSWD: /opt/scriptsCronJobs/offDionaea.sh,
/opt/scriptsCronJobs/onDionaea.sh
```

## INSTALACION EN EL ENTORNO DE PRODUCCION

Tras la instalación de VirtualBox en el servidor para explotación existen dos filosofías de arranque de las máquinas virtuales. Inicio manual por el administrador del sistema ó arranque de la misma como un servicio del sistema. En concreto durante el desarrollo el proyecto el honeypot estará instalado en humo.ugr.es (servidor Supermicro Intel Xeon ,8GB Ram, Windows Server 2003), que realiza la función de virtualización de servicios en Betatesting para el C.S.I.R.C, y es iniciado como un servicio del sistema durante el arranque (prelogin) junto con otras máquinas virtuales.

### Instalación y configuración (Windows Server 2003)

**Nota Importante:** Es altamente recomendable que las máquinas arrancadas como servicios del sistema no esten levantadas en el momento de arrancar VirtualBox (interface gráfica). Ya que se pueden dar errores de consistencia entre VirtualBox y VboxHeadless (manejador de máquinas virtuales sin la interface), si accedemos por error a la configuración de alguna de las máquinas que están corriendo en ese momento.

Copiamos el archivo zetsu.vdi en

C:\Documents and Settings\usuario\.VirtualBox\HardDisks

Arrancamos VirtualBox -> Pulsamos el boton “Nueva” ->



En el menú crear nueva máquina virtual

Nombre zetsu

Sistema operativo: Linux

Versión: Ubuntu

Pulsamos Next

En la siguiente pantalla nos saldrá seleccionado por defecto 512 MB pero con 256 MB podemos funcionar sin problemas y economizaremos recursos del sistema anfitrión

Pulsamos Next

En la siguiente pantalla seleccionamos usar disco duro existente

- ☒ Disco duro de arranque
- ☐ Crear disco virtual nuevo
- ☒ Usar un disco duro existente

Pulsamos



Pasamos al menú del administrador de medios virtuales y pulsamos



Navegamos por el menú hasta “C:\Documents and Settings\usuario\.VirtualBox\HardDisks”, seleccionamos zetsu.vdi y pulsamos abrir.

Volveremos al menú de medios virtuales y nos saldrá como un disco duro.

Discos duros			Imágenes de CD/DVD	Imágenes de disquete
Nombre	Tamaño virtual	Tamaño real		
zetsu.vdi	10,00 GB	3,71 GB		

Lo marcamos y pulsamos seleccionar.

Ahora nos saldrá seleccionado como disco duro de nuestra maquina virtual.

- ☒ Usar un disco duro existente
- zetsu.vdi (Normal, 10,00 GB)

Pulsamos Next y terminar.

Por último la única configuración pos instalación es asignarle una interface de red virtual conectada mediante bridge a la interface de red del equipo anfitrión (maquina física).

Para ello pulsamos configuración



Y en la sección de Red

Marcamos habilitar conector.

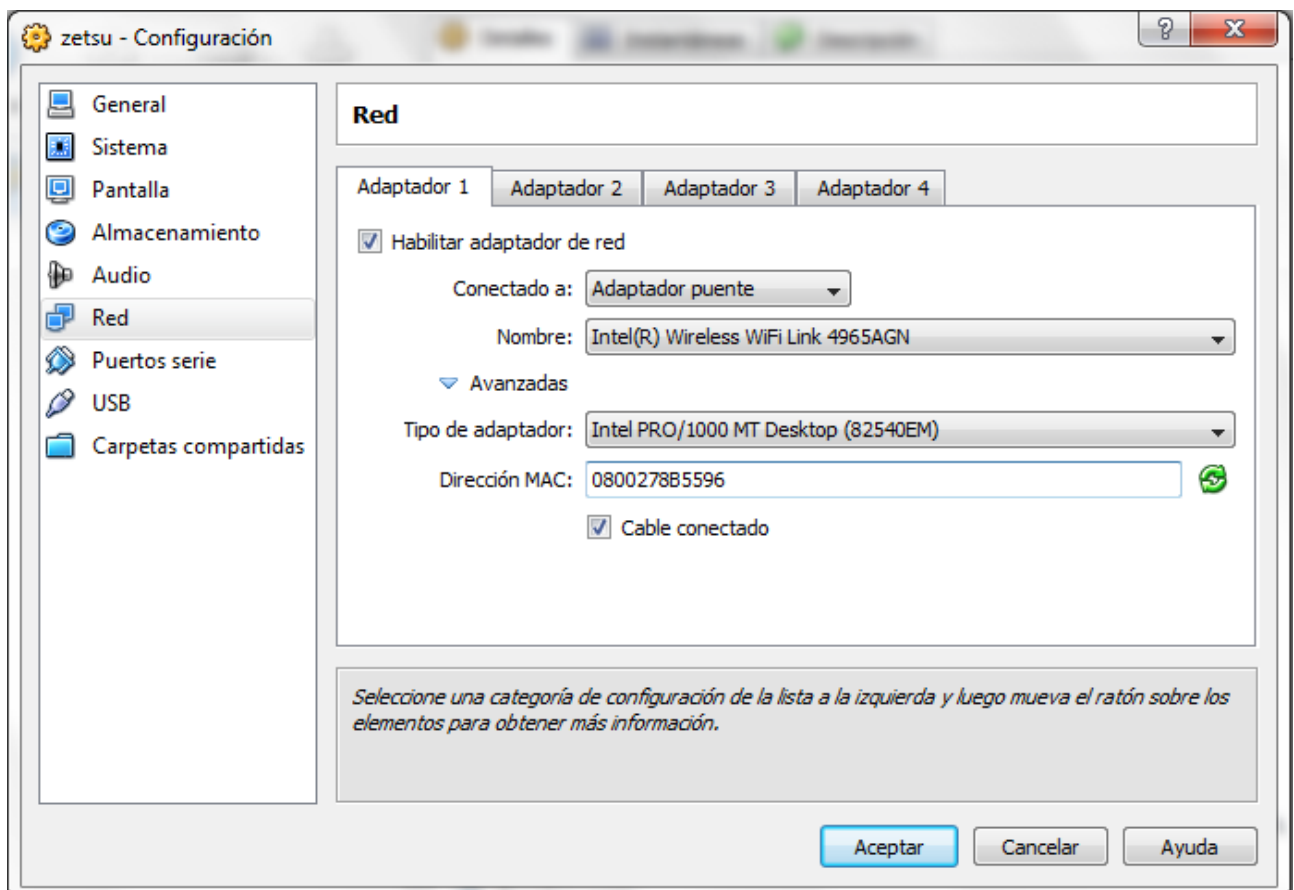
Seleccionamos Conectado a Adaptador puente

Seleccionamos a interface física

Desplegamos Avanzadas

Seleccionamos tipo de adaptador alguno de los existentes (menos red paravirtualizada)

Y marcamos cable conectado



En el caso de la red de la U.G.R es importante que el servicio de redes nos de una mac asociada a una ip para tener acceso a la red. Y será en este menú donde la asignemos.

## CONFIGURACION DEL HONEYPOT VIRTUAL EN EL HOST REAL

### Arranque como servicio del sistema del honeypot virtual:

#### Requisitos:

Instalar Windows Resource Kits en la carpeta srvany. Crear una carpeta srvany, en system32 se facilita no usar direcciones absolutas.

#### Creación del servicio del sistema:

Desde dentro de la carpeta ejecutamos.

"ruta del sistema"\INSTSRV.EXE MaquinaVirtualZetsu "ruta del sistema"\SRVANY.EXE"

Se nos creara el servicio de sistema MaquinaVirtualZetsu dentro del registro de windows.

#### Configuración del registro de Windows:

Buscamos MaquinavirtualZetsu.

Dentro del registro se crea una carpeta Parameters.

Dentro de Parameters creamos tres registros de strings values tales como:

AppDirectory "c:\Program Files\Oracle\VirtualBox\"


Application "c:\windows\system32\cmd.exe"


AppParameters /c "c:\Program Files\Oracle\VirtualBox\VBBoxHeadless.exe" -startvm zetsu -p 3380

### Apagado como servicio del sistema del honeypot virtual:

El apagado del servidor no contempla el apagado seguro de las máquinas virtuales lanzadas como servicios del sistema. Por eso definimos el siguiente script y lo añadimos a las políticas de apagado del sistema como se indica en:

[http://technet.microsoft.com/en-us/library/cc783802\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc783802(v=ws.10).aspx)

 C:\notepad shutdownVM.bat

  
@ECHO off  
ECHO ----- >> VMShutdown.log  
ECHO Registro de apagado de maquinas virtuales >> VMShutdown.log  
ECHO ----- >> VMShutdown.log  
sc stop MaquinaVirtualZetsu >> VMShutdown.log  
ECHO. >> VMShutdown.log  
ECHO Apagado del servicio MaquinaVirtualZetsu registrado el "%date%" a las "%time%" >>  
VMShutdown.log  
ECHO ----- >> VMShutdown.log