DEPLOY A DNS SERVER IN A SECURE WAY

BIND (Berkeley Internet Name Domain) is one of the more widely used DNS servers. This article guides readers on how to deploy a BIND DNS server in a secure way by implementing three features of BIND-Transaction Signature, Zone signing and Views.

Before starting with the security aspect, let's deploy a master and a slave server. I am using two RHEL 6 machines running on VMware. The master DNS server is named server.example.com, and its IP address is 192.168.0.1; the slave DNS server is client.example.com, 192.168.0.2.

Configuring the master DNS server

To deploy a minimal master DNS server, first install the required bind packages (run yum y install bind bind-utils). Next, edit /etc/named.conf, and under the Options field, set listen-on port 53 { 192.168.0.1; }; and allow-query {any; };. At the end of the file, append the following lines:

```
zone example.com IN {
type master;
file □forward□;
allow-update { none; };
};
```

Next, run:

```
cd var/named
cp named.localhost forward
chgrp named forward
```

Now edit the forward file, add the following lines, and save it:

```
$TTL 1D
@ IN SOA server.example.com. root.server.example.com. (
0 ; serial
1D ; refresh
1H ; retry
1W ; expire
3H ) ; minimum
@ IN NS server.example.com.
server IN A 192.168.0.1
client IN A 192.168.0.2
```

Now adjust your firewall and SELinux settings (or just turn them off for now, with service iptables stop and setenforce 0). Start the DNS service (service named start).

Configuring the slave DNS server

The first two steps are the same as for the master server, but the lines to be appended in*named.conf* will be:

zone □example.com □ IN {
type slave;
file □slaves/example.com□;
masters { 192.168.0.1; };
};

Again, adjust your firewall and SELinux settings or turn them off, and start the named service.

Now the master and slave DNS server is running. For the master machine, set the DNS server to 192.168.0.1, and for the slave server, set it to 192.168.0.2. Then verify the configuration by getting both machines to ping each other. If they are able to do so, then our master-slave DNS server configuration is working. On the slave server, check the /var/named/slaves directory and you □ll see a file example.com, which is the zone file for the example.com zone, transferred to the slave server from the master DNS server.

Security Pillar 1

It's time to switch to security, and the first pillar of this trilogy is transaction signatures (TSIG). First, run:

dnssec-keygen a HMAC-MD5 b 128 n HOST server.example.com

You □ll see two files generated, in the format Kserver.example.com.+157+08837.key and Kserver.example.com.+157+08837.private. The number 157 denotes the DNSSEC algorithm, and 08837 is the key's fingerprint, which will differ on your machine. Open the private file and copy the key (you will find it in the third line of the file). On both machines, create a file */etc/server.key* with the following contents (replacing the secret string with the actual copied key):

key "server.example.com." {
algorithm hmac-md5;
secret "the key that you've copied";
};

Now run chown root.named /etc/server.key and on both servers, in named.conf, add include "/etc/server.key"

allow-transfer { key server.example.com.; }; under the options field in named.conf while for the slave server's named.conf, add the following lines just below the new include line:

server 192.168.0.1 {
keys { server.example.com.; };
};

Delete the /var/named/slaves/example.com file from the slave machine. Restart both the DNS servers and verify by pinging. You will see that the example.com file appears again under*/var/named/slaves.*

So far, as seen in our initial configuration, the example.com zone file is transferred to the slave DNS server via zone transfer. But under that configuration, zone files can be transferred to any machine. We have now implemented zone transfer security using TSIG in our first security drill now, only machines with a valid key can get a zone file transferred from the master DNS server. You can verify this, if you like, by changing the slave's configuration. Again delete the example.com file under /var/named/slaves and comment the include and server line in the slave's named.conf file. Restart the named service, and check the /var/named/slaves directory. This time, you won't find the example.com file, because you are not using the key;so it is not being transferred.

Security Pillar 2

So far, we have seen the method of allowing only trusted machines to do a zone transfer. Now let's look at how to validate the authenticity of the zone file itself. On your master server, run the following commands:

dnssec-keygen □a DSA □b 512 □n ZC

dnssec-keygen □f KSK □a D

The first will create a Zone Signing Key (ZSK) in the familiar format Kexample.com.003+46600.key while the second will create a Key Signing Key (KSK) in the same format. Now open your zone file (forward) and append the following lines (the paths of the key files):

$include /var/named/ Kexample.com.+003+46600.key ;ZSK

$include /var/named/ Kexample.com.+003+44487.key ;KSK

Then run this command:

dnssec-signzone \□o example.com

□k /var/named/Kexample.com.+003+44487.key \

forward \

/var/named/Kexample.com.+003+46600.key

This will create a file, forward.signed, which is the signed zone file. The -o switch specifies the zone name; k the full path to the KSK file; and next is the name of the zone file (forward), followed by the path to the ZSK file.

Now change named.conf to use this signed file as the zone file; in the file line for the zone □example.com□ block, □forward.signed□ so the block looks like what□s shown below:

zone □example.com □ IN {

type master;
file □forward.signed□;
allow-update { none; };
};

We now have our signed zone ready, so just restart the named service on both machines.

Security Pillar 3

The third and final security measure is to serve different information to different machines. This will be implemented by using the view directive in named.conf. On the master server, start with a fresh copy of the named.conf file; so (back up and) delete your named.conf file and create a new one with the following lines:

```
acl                □internal□ { 192.168.0.1; };
acl                □external□ { 192.168.0.2; };
options {
directory          □/var/named□;
recursion no;
#you can also add further options here
};
view               □internal□ {
match-clients {          □internal□; };
zone                     □example.com□ IN {
type master;
file □forward□;
};
};
view               □external□ {
```

```
match-clients {                 □external□; };
zone                            □example.com□ IN {
type master;
file □forward_new□;
};
};
```

After this, run the following commands:

```
cp /var/named/forward /var/named/forward_new
chown root.named /var/named/forward_new
```

Now edit the contents of the forward_new file, and delete the last line so that it contains a record only for the server machine:

```
$TTL 1D
@ IN SOA server.example.com. root.server.example.com. (
0 ; serial
1D ; refresh
1H ; retry
1W ; expire
3H ) ; minimum
@ IN NS server.example.com.
server IN A 192.168.0.1
```

Set the DNS server to 192.168.0.1 for both the machines, and restart the named service. Try to ping both the machines from the first machine, which will happen successfully. Next, switch to the second machine with the IP 192.168.0.2. Now try to ping both the machines. You will see that it pings server.example.com successfully, but fails to ping *client.example.com*. This is because the DNS server is using the forward_new file for this machine, as directed in the view directive,

and since there is no record for client.example.com in that file, it is unable to find that machine. So, we have two different machines, two different zone files, and two different views.

As I end this article, I hope you now have a better insight into deploying a DNS server in a secure manner. Keep exploring and wait for our next encounter!

Source : http://www.opensourceforu.com/2013/12/deploy-dns-server-secure-way/