Tutorial on Configuring BIND to use Response Policy Zones (RPZ)

This guide is based on a training Andrew Fried of Deteque gave at a M3AAWG Conference in February, 2017. At the time, the current version of BIND was 9.11.0-P3.

This tutorial assumes some working ability with Linux, but otherwise all the commands and steps are provided, including a very basic sample configuration. The configuration files used in this guide are available for download from: https://deteque.com/m3aawg-bind-training/

Contents

A) The five steps to setting up BIND to use RPZ are:
1. Install BIND
2. Create a BIND Configuration File (and the importance of closing your resolver)
3. Create a Local RPZ Server (Master)
4. Configure a Slave RPZ Zone
5. Enable RPZ Policy Zones (and the importance of testing a new zone file)

B) RPZ Triggers and Actions explained:
1. Triggers (and the importance of ordering)
2. Actions
3. Configuring Response Policy Zones (and the importance of a local RPZ Zone)

What is RPZ?

Developed by Paul Vixie (ISC) and Vernon Schryver (Rhyolite). RPZ was first publicly announced at Black Hat in July, 2010. Also referred to as a "DNS firewall".

RPZ provides a way to "rewrite" a DNS response. Normally a rewrite would return an NXDOMAIN, or "no such answer" response for a query whose return data matches an RPZ "trigger".

RPZ rules can be configured to rewrite DNS queries based on:

- IP Address/Subnet (RPZ-IP)
- Hostname/Domain (QNAME)
- Nameserver Name/Domain (RPZ-NSDNAME)
- Nameserver Address/Subnet (RPZ-NSIP)
- Client IP (RPZ-CLIENT-IP)

The RPZ rule set is carried in a DNS zone file. RPZ policy zones can be sent to slaves using AXFR/IXFR. TSIG can be used to authenticate the zone transfers.

RPZ requires decent hardware to run on. There are a lot of factors to consider when specifying hardware for use as an RPZ-enabled recursive DNS server, but a good starting point would be:

- 8 Core CPU with at least a 2.4 gHz clock speed
- 8 GB of RAM
- Servers should be bare metal - not virtualized

1. INSTALL BIND

The correct way to deploy Bind with RPZ is to download the source, compile it, then configure it. Using apt-get or yum will almost certainly install an older back-ported version of the application. Not good.

Assuming you're using Ubuntu 16.04.1, you'll need to apt-get two dependencies that bind needs - the compiler and the libssl library. We'll use apt-get:

apt-get clean
apt-get update
apt-get install build-essential
apt-get install libssl-dev
ldconfig

Let's begin by creating a couple of directories that bind will use for its files:

mkdir /etc/namedb

mkdir /etc/namedb/bind

Next, we need to download Bind. It's available from Internet System Consortium's (ISC) website. Go to their website at:
https://www.isc.org

Download the most current version. For example, download:
bind-9.11.2.tar.gz

One you download it, move it to
/etc/namedb/bind/
Now go to /etc/namedb/bind and untar the source file archive:
tar zxvf bind-9.11.2.tar.gz

You will see a new directory created called "bind-9.11.2"
Go into that directory:
cd bind-9.11.2
Now we need to compile the program. This is a three-stage process:

- configure
- make
- make install

To make this a lot easier, please download config files from:
https://deteque.com/m3aawg-bind-training/
Here's the files we'll need:

- CONFIGURE-BIND.sh
- db.rpz.local
- named.conf
- root.cache

Lets create a script to do the configuration:

```
#!/bin/sh
./configure \
        --enable-threads \
```

```
        --with-randomdev=/dev/urandom \
        --prefix=/usr \
        --sysconfdir=/etc \
        --datadir=/etc/namedb \
        --with-openssl=yes \
        --with-tuning=large \
        --enable-largefile \
        --with-aes \
 —with-libjson=no
```

This file should have been downloaded - it's called:
CONFIGURE-BIND.sh
Download it and be sure to make it executable:
chmod 700 CONFIGURE-BIND.sh

Now lets configure the source:
Be sure you're in the bind-9.11.2 directory, then lets execute that configure script you just created:

/etc/namedb/CONFIGURE-BIND.sh
You'll see a lot of text fly by. What you need to pay attention to is at the end of the compilation bind will complain about any missing libraries that you need.

Next, we're going to compile bind by typing:
make

Finally, we're going install the program on our system:
make install

We need to install the latest root.cache file into the /etc/namedb. Lets run this command to do that:
**/usr/bin/wget --user=ftp --password=ftp **
**ftp://ftp.rs.internic.net/domain/db.cache **

-O /etc/namedb/root.cache

The file should have been downloaded as: update-root-cache.sh

We're going to need an rndc key later on, so let's create it now:
rndc-confgen > /etc/rndc.conf
chmod 600 /etc/rndc.conf

That file contains a key we'll need to add to the main configuration file.
You can also run the script "create-rndc-key.sh" to create the rndc key.

2. CREATE A BIND CONFIGURATION FILE

Now comes the fun part - creating BIND's configuration file. For this exercise we're going to create a bare bones file. This file will need to go at /etc/named.conf, but once we're done we'll make a copy of it and store it under /etc/namedb just in case….
Please download the file: https://deteque.com/m3aawg-bind-training/named.conf"
then copy that file to /etc

2-Minute Intro to T.SIG
We won't need this, but just in case you find the need to generate TSIG keys, here's how you do it:
dnssec-keygen -a hmac-sha256 -b 256 -n HOST [keypair name]

You'll replace the [keypair name] with the name of the key,
dnssec-keygen -a hmac-sha256 -b 256 -n HOST testkey

That will create two files that look something like:
Ktestkey.+163+16005.key
Ktestkey.+163+16005.private
In the private file, you'll seen an entry that begins with "key:". That's the tsig key.

Private-key-format: v1.3
Algorithm: 163 (HMAC_SHA256)
Key: 2bRaxnyRwv2shCUJpnJWuW6EfrLackhGR+5PGjTSGlM=

Bits: AAA=
Created: 20170212211321
Publish: 20170212211321
Activate: 20170212211321

In your config, you'd add:

```
key testkey {
        algorithm hmac-sha256;
        secret "2bRaxnyRwv2shCUJpnJWuW6EfrLackhGR+5PGjTSGlM= ";
};
```

When using a TSIG key, you need to specify the key and the server that key works with. Lets make pretend you'll be pulling zones from a nameserver at 123.45.67.88 and TSIG authentication is required:

```
key testkey {
        algorithm hmac-sha256;
        secret "2bRaxnyRwv2shCUJpnJWuW6EfrLackhGR+5PGjTSGlM= ";
};
server 123.45.67.88 {
   keys { testkey; };
};
```

The next section we need to configure is logging:

Since we're running RPZ, we definitely want to log any RPZ rewrites. To do that, we need to set up two things under the "logging" header.

```
    channel rpzlog {
        file "rpz.log" versions unlimited size 1000m;
        print-time yes;
        print-category yes;
```

```
        print-severity yes;
        severity info;
    };
    category rpz { rpzlog; };
```

Some miscellaneous stuff:

```
controls {
        inet 127.0.0.1 port 953
        allow { 127.0.0.1; } keys { rndc-key; };
};
server fe80::/16 { bogus yes; };
trusted-keys {
};
```

3. CREATE A LOCAL RPZ SERVER

```
$TTL 300
@           IN SOA  localhost. need.to.know.only. (
                201702121 ; Serial number
                60      ; Refresh every minute
                60      ; Retry every minute
                432000   ; Expire in 5 days
                60 )    ; negative caching ttl 1 minute
            IN NS   LOCALHOST.
deteque.com                 IN CNAME rpz-passthru.
*.deteque.com               IN CNAME rpz-passthru.
spamhaus.org                IN CNAME rpz-passthru.
*.spamhaus.org              IN CNAME rpz-passthru.
32.25.195.194.32.rpz-ip     IN CNAME rpz-passthru.      ; whitelist 34.194.195.25/32
32.71.219.156.35.rpz-ip     IN CNAME rpz-passthru.      ; whitelist 35.156.219.71/32
example.com                 IN CNAME .                  ; local block against example.com
*.example.com               IN CNAME .                  ; local block against example.com
```

A sample rpz.local file is on the download server at https://deteque.com/m3aawg-bind-training/

4. CONFIGURE A SLAVE RPZ ZONE

```
zone "drop.rpz.spamhaus.org" {
        type slave;
        file "dbx.drop.rpz.spamhaus.org";
        masters {
                34.194.195.25;
                35.156.219.71;
        };
        allow-transfer { none; };
        allow-query { localhost; };
};
```

Note that your RPZ zones should only allow queries from localhost!

5. ENABLE RPZ POLICY ZONES

```
options {
        directory "/etc/namedb";
        key-directory "/etc/namedb";
        pid-file "/var/run/named.pid";
        recursing-file "named.recursing";
        statistics-file "rndc.stats";
        recursion yes;
        allow-transfer{ none; };
        ixfr-from-differences yes;
        empty-zones-enable yes;
        allow-recursion  {
```

```
        ::1;
        127.0.0.0/8;
        MY-COMPANY;
};
allow-query-cache {
        ::1;
        127.0.0.0/8;
        MY-COMPANY;
};
response-policy {
        zone "rpz.local";

        zone "drop.rpz.spamhaus.org";
```

At this point, BIND is ready to consume RPZ files for use in creating a 'DNS firewall'.

EXAMPLE OF POLICY OVERRIDE FOR TESTING ZONES

When first implementing a new rpz policy, you might want to use a pass-through rule for a testing period of several days. This will not change the behavior of the BIND server, it will continue to permit queries to complete, while logging whether there were any queries that triggered this RPZ rule.

```
response-policy {
        zone "rpz.local";
        zone "drop.rpz.spamhaus.org POLICY RPZ-PASSTHRU“;
};
```

TESTING

In order to take advantage of RPZ, you need data to put into the RPZ zones. There are commercial RPZ zone feeds available from a number of vendors. You should always evaluate a potential new zone for at least 10 days before "turning it on" to be absolutely certain that it won't be a cause of false positives or conflict with any special needs your site has. Have a plan for dealing with exigent issues. This is one of the main reasons you should have a local RPZ zone

A word or two about security
The day of running open recursive name servers is long gone for most of us. Failing to configure a name server to reject outside queries basically makes your name server an instrument for denial of service attacks. Always restrict who can query your recursive servers using either ACLs or hard firewall rules vial IPTABLES.

DO NOT PUT AN INSECURE DNS SERVER ON THE INTERNET. Lock them down!

A second point about security - if you're using commercial RPZ zones, failing to secure your server could be viewed as a violation of terms of service. You should restrict who can query the recursive nameserver and, additionally, make sure no one can download the zones from your server. Internal distribution of RPZ data should be done using both ACLs (addresses of the authorized servers) plus TSIG authentication.

RPZ Triggers and Rules (Actions)

RPZ rules can be configured to rewrite DNS queries based on:
- IP Address/Subnet (RPZ-IP)
- Hostname/Domain (QNAME)
- Nameserver Name/Domain (RPZ-NSDNAME)
- Nameserver Address/Subnet (RPZ-NSIP)
- Client IP (RPZ-CLIENT-IP)

Creating a trigger rule for a Domain
Let's say we want to rewrite any DNS queries for a domain, but allow queries for hosts in that domain:
example.com IN CNAME .

This would result in an NXDOMAIN response for a query for "example.com" but would allow a query for a hostname in the domain, i.e. "host.example.com"
Note: do not add a period after the "owner" name!

Creating a trigger rule for a Hostname
Let's say we want to rewrite any DNS queries for a specific hostname, but allow lookups to the domain and other hosts in that domain:
host.example.com IN CNAME .

This would result in an NXDOMAIN response for a query for "host.example.com" but would allow a query for the domain as well as other hosts in the domain.
Note: do not add a period after the hostname!

Creating a trigger rule for everything in a Domain
Let's say we want to rewrite any DNS queries for a given domain, as well as for all of the hosts in that domain:
example.com IN CNAME .
*.example.com IN CNAME .

This would result in an NXDOMAIN response for any query related to example.com. This is typically the most often used format for RPZ.
Note: do not add a period after the hostname!

Creating a trigger rule for an IP or Subnet (v4)
Let's say we want to rewrite any DNS queries for any hosts that resolve in the 172.16.3.0/24 subnet.
24.0.3.16.172.ns-ip IN CNAME .

As you can see, the octets in the subnet need to be reversed (similar to the way rbl in-addr.arpa works). The first number represents the subnet mask. If we only wanted to block a single ip, the first number would be 32, which represents a /32, i.e.:
32.1.3.16.172.ns-ip IN CNAME .

Creating a trigger rule for an IP or Subnet (v6)
Let's say we want to rewrite any DNS queries for any hosts that resolve to an IPv6 subnet like 2345:68::a/64. To do this we must normalize the v6 address to:
2345:68:0:0:0:0:0:a

Just like the v4, the subnet mask becomes the first entry, followed by the octets of the v6 address is reverse order:
64.a.zz.68.2345.ns-ip IN CNAME .
64.a.0.0.0.0.0.68.2345.ns-ip IN CNAME .

The "zz" is a shorthand equivalent for "::"

Creating a trigger rule for a Nameserver
Let's say we want to rewrite any DNS queries for all hosts and domains whose authoritative name server is ns1.badnameser.com:
ns1.badnameserver.com.rpz-nsdname IN CNAME .

Creating a trigger rule for a Nameserver IP
Let's say we want to rewrite any DNS queries for all lookups where the authoritative name server resides on 192.168.55.68 (/32 implied)
32.68.55.168.192.rpz-nsip IN CNAME .

A special use for the Client IP
Let's our company has cutting edge security and has RPZ enabled recursive nameservers used throughout the organization. Your incident response folks, whose subnet is 192.168.20.0/23 needs unrestricted access to the Internet. The Client IP trigger can be used to allow their queries to bypass the rest of the RPZ checks as long as this rule set (i.e. rpz.local) appears BEFORE subsequent rule sets that would otherwise rewrite responses 23.0.20.168.192.rpz-client-ip IN rpz-passthru.

A note about "ordering"
There is no special ordering for rules within a given rule set. However, there is an ordering based on rule sets themselves. This is why you should always list the most egregious rule sets at the "top of the list" in the response-policy section of the Bind configuration. If you create a local rule set, it should be the first rule set that appears in the response-policy section!

RPZ Actions

Now that we covered the RPZ triggers it's time to look at RPZ actions. The triggers are used to identify objects that require RPZ intervention. The actions are the

"right hand" portion of the RPZ rules that determine what type of action/rewrite the resolver should make. We've already worked with the first, which is "CNAME ."
"CNAME ." synthesizes an NXDOMAIN response for a domain/hostname

RPZ Actions
The currently supported RPZ policies are:

- GIVEN
- DISABLED
- PASSTHRU
- DROP
- TCP-ONLY
- NODATA
- NXDOMAIN

RPZ Actions - GIVEN
This is only useful when used within the configuration file and basically tells Bind to let the RPZ determine its own actions. This is the default.

RPZ Actions - DISABLED
The testing override DISABLED policy causes policy zone records to do nothing but log what they would have done if the policy zone were not disabled. The response to the DNS query will be written (or not) according to any triggered policy records that are not disabled. Disabled policy zones should appear first, because they will often not be logged if a higher precedence trigger is found first. This is used in the response-policy section of the bind configuration file.

RPZ Actions - PASSTHRU
Specifically allows the query to resolve correctly but will produce logs of the query in the rpz.log file.
This is most often used to punch holes in existing subnets or test new zones by logging potential rewrites without actually performing the rewrites. This is normally used in the local RPZ zone or as an override action within the response-policy section of the configuration file

RPZ Actions - DROP

This policy will simply discard the query and will not return any response to the client. This is normally used to block a client:

16.0.0.16.192.rpz-client-ip IN CNAME rpz-drop.

RPZ Actions - TCP-ONLY
This policy forces the resolver to use TCP for the query.

; force some DNS clients and responses in the example.com
; zone to TCP
16.0.0.1.10.rpz-client-ip CNAME rpz-tcp-only.
example.com CNAME rpz-tcp-only.
*.example.com CNAME rpz-tcp-only.

RPZ Actions - NODATA
Returns an empty response for the matching trigger but will return nxdomain responses for subdomains.
The format for the these records is a little different:

example.com CNAME *.
*.example.com CNAME *.

RPZ Actions - NXDOMAIN
By far the most common policy used in RPZ. The rule uses the "CNAME ." as the policy, i.e.:
baddomain.com IN CNAME .
*.baddomain.com IN CNAME .

Configuring Response Policy Zones
RPZ zones are specified in the response-policy section:

```
response-policy {
        zone "rpz-local";
```

```
        zone "tor-exit-nodes.local";
        zone "bogon.rpz.spamhaus.org";
        zone "botnetcc.rpz.spamhaus.org";
        zone "malware.rpz.spamhaus.org";
        zone "malware-adware.rpz.spamhaus.org";
        zone "malware-aggressive.rpz.spamhaus.org";
        zone "bad-nameservers.rpz.spamhaus.org";
        zone "drop.rpz.spamhaus.org";
        zone "abused-legit.rpz.spamhaus.org";
        zone "dbl.rpz.spamhaus.org";
        zone "malware.rpz.oitc.com";
        zone "phish.rpz.oitc.com";
        zone "misc.rpz.oitc.com";
        zone "rpz.surbl.org";
};
```

Configuring Response Policy Zones

Benefits of using separate zone files:

- Allows prioritization of the zones - the most egregious at the top
- Results in much more meaningful logging
- Allows zones with varying TTLs to replicate more efficiently
- Less “breakage” from Bind journal file issues
- Bind currently has a 32 zone limit
- Best Common Practice is to **ALWAYS** create a local RPZ zone

Your local RPZ zone should whitelist your domain(s) and addresses you control and should be placed at the top of the list in the response-policy section. This zone can also be used to whitelist items that may appear in other zones - which gives you a way of mitigating false positives or modifying your policy to allow PASSTHRU queries for your clients.

A sample template for a standard DNS zone file:

```
$TTL 300
```

```
@       IN SOA  n1.my.domain. dnsadmin.my.domain. (
              1486847439 ; Serial number
     60             ; Refresh every 1 minutes
              60            ; Retry every minute
              432000        ; Expire in 5 days
              60 )          ; negative caching ttl 1 minute
          IN NS   LOCALHOST.
; RPZ Data goes below
```

RPZ.local example:

```
$TTL 300
@       IN SOA  localhost. need.to.know.only. (
              1486847439 ; Serial number
     60             ; Refresh every 1 minutes
              60            ; Retry every minute
              432000        ; Expire in 5 days
              60 )          ; negative caching ttl 1 minute
          IN NS   LOCALHOST.
deteque.com         IN CNAME rpz-passthru.
*.deteque.com       IN CNAME rpz-passthru
onion.link          IN CNAME .                  ; High risk tor gateways
*.onion.link        IN CNAME .                  ; High risk tor gateways
```

Example named.conf file

(available for download from https://deteque.com/m3aawg-bind-training/)

```
acl MY-COMPANY {
     199.168.90.180;
```

```
};

logging {
        channel null {
          null;
     };

        channel bindlog {
          file "bind.log";
          print-time yes;
          print-category yes;
          print-severity yes;
          severity info;
     };

        channel rpzlog {
                file "rpz.log" versions unlimited size 1000m;
                print-time yes;
                print-category yes;
                print-severity yes;
                severity info;
        };

        category default { bindlog; };
        category general { bindlog; };
        category database { null; };
        category config { bindlog; };
        category resolver { null; };
        category xfer-in { bindlog; };
        category xfer-out { bindlog; };
        category notify { bindlog; };
        category client { null; };
        category unmatched { null; };
        category network { bindlog; };
        category update { bindlog; };
```

```
        category update-security { bindlog; };
        category queries { null; };
        category dispatch { null; };
        category lame-servers { null; };
        category delegation-only { bindlog; };
        category edns-disabled { null; };
     category rpz { rpzlog; };
};

key rndc-key {
     algorithm hmac-md5;
     secret "rEbX202KhpwchN2OAsEn/A==";
};

key testkey. {
    algorithm hmac-sha256;
    secret "gDy28KboVkKvx7S/F05QQKgsRMjPN51a6oVQPt3AuJI=";
};

server 1.2.3.4 {
     keys { testkey.; };
};

controls {
     inet 127.0.0.1 port 953
     allow { 127.0.0.1; } keys { rndc-key; };
};

options {
     directory "/etc/namedb";
     key-directory "/etc/namedb";
     pid-file "/var/run/named.pid";
     recursing-file "named.recursing";
     statistics-file "rndc.stats";
     recursion yes;
```

```
     allow-transfer{ none; };
     ixfr-from-differences yes;
     empty-zones-enable yes;
     allow-recursion  {
          ::1;
          127.0.0.0/8;
          MY-COMPANY;
     };
     allow-query-cache {
          ::1;
          127.0.0.0/8;
          MY-COMPANY;
     };
     response-policy {
          zone "rpz.local";
          zone "drop.rpz.spamhaus.org";
     };
};

#------------------------------------------------------------------------
# Spamhaus RPZ Files
#------------------------------------------------------------------------

zone "rpz.local" {
     type master;
     file "db.rpz.local";
     allow-update { none; };
     allow-transfer { none; };
     allow-query { localhost; };
};

#------------------------------------------------------------------------
# Spamhaus RPZ Files
#------------------------------------------------------------------------
```

```
zone "drop.rpz.spamhaus.org" {
     type slave;
     file "dbx.drop.rpz.spamhaus.org";
     masters {
          34.194.195.25;
          35.156.219.71;
     };
     allow-transfer { none; };
     allow-query { localhost; };
};

#---------------------------------------------------------------------------
# Root hints
#---------------------------------------------------------------------------

zone "." {
     type hint;
     file "root.cache";
};
```