

September 2006

Geoff Huston

DNSSEC - The Practice

Last month's column looked at the theory of DNSSEC, examining the additional Resource Records that are defined by DNSSEC, how these Resource Records are interpreted, and the manner in which these additional RR's can be used to authenticate DNS responses. All this is fine in theory, but how well does all this work in practice? In this column I'd like to describe my experiences in configuring a DNSSEC-aware DNS resolver, and setting up a DNSSEC-secured zone, and see how easy, or hard, it is to use DNSSEC in practice.

The Objective

In this exercise I would like to use DNSSEC to sign two DNS zones. The first zone, *dnssec.potaroo.net*, is to be a signed zone which is a descendant from the unsigned zone *potaroo.net*. The second zone, *sub.dnssec.potaroo.net*, is to be a signed immediate descendant zone from the first signed zone. In theory the one trust anchor should be sufficient to validate both zones. I also would like to check that the signing was successful, so I'll need to construct a dnssec-aware local name resolver.

I'd like to complete this exercise without calling for any expert assistance, and, if possible, restrict myself to online documentation during this exercise. After all, if DNSSEC really is ready for deployment then installation of DNSSEC-aware tools and services should be a mundane exercise in following the documentation. As my primary guide through the steps of DNSSEC configuration using BIND I'll be using the latest version I can locate of a RIPE tutorial document, "DNSSEC HOWTO", <http://www.ripe.net/disi/dnssec_howto/dnssec_howto.pdf>. I'm using version 1.7 of this document, from April 2005.

My platform operating system environment is FreeBSD, and I'm currently running version 6.1 of this operating system. I'll be using the BIND implementation of DNS for both the resolver tools and library, and the BIND name server.

Step 1 - Getting Bind

As of September 2006 the latest BIND version is 9.3.2-P1, and its obtainable from the Internet Software Consortium <<http://www.isc.org>>. The source code for this release is at <<http://ftp.isc.org/isc/bind9/9.3.2-P1/bind-9.3.2-P1.tar.gz>>. This code pack also appears in the FreeBSD Ports collection (<http://www.freebsd.org/cgi/cvsweb.cgi/ports/dns/bind9/>), so in this case I'll use the FreeBSD ports version, and follow the instructions associated with installation of this application.

Step 2 - Installing Bind9

I notice in looking through the Makefile for Bind there is a reference to OpenSSL:

```
.if defined(WITH_OPENSSL_PORT)
CONFIGURE_ARGS+=      --with-openssl=${LOCALBASE}
.else
CONFIGURE_ARGS+=      --with-openssl
.endif
```

So I'll install OpenSSL before moving on to Bind

```
# cd /usr/ports/security/openssl
# make
==> Vulnerability check disabled, database not found
=> openssl-0.9.8c.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch from http://www.openssl.org/source/.
openssl-0.9.8c.tar.gz          100% of 3236 kB   119 kBps
```

```

===> Extracting for openssl-0.9.8c
=> MD5 Checksum OK for openssl-0.9.8c.tar.gz.
=> SHA256 Checksum OK for openssl-0.9.8c.tar.gz.
...
#make install
...
# /usr/local/bin/openssl version
OpenSSL 0.9.8c 05 Sep 2006

```

And on to installing BIND9

```

#cd /usr/ports/dns/bind9
# make install
...
*****
*
*          AATTENTION
*
* BIND 9 requires a good source of randomness to operate.
* It also requires configuration of rndc, including a
* "secret" key. If you are using FreeBSD 4.x, visit
* http://people.freebsd.org/~doug/randomness.html for
* information on how to set up entropy gathering. Users
* of FreeBSD 5.x or later do not need to do this step. If
* you are running BIND 9 in a chroot environment, make
* sure that there is a /dev/random device in the chroot.
*
* The easiest, and most secure way to configure rndc is
* to run 'rndc-confgen -a' which will generate the proper
* conf file, with a new random key, and appropriate file
* permissions.
*
*****
===> Compressing manual pages for bind9-base-9.3.2.1
===> Registering installation for bind9-base-9.3.2.1
===> SECURITY REPORT:
    This port has installed the following files, which may act as network
    servers and may therefore pose a remote security risk to the system.
/usr/sbin/named-checkconf
/usr/sbin/rndc
/usr/sbin/lwresd
/usr/bin/nsupdate
/usr/bin/dig
/usr/sbin/named
/usr/bin/host
/usr/sbin/dnssec-sigzone
/usr/bin/nslookup
/usr/sbin/named-checkzone

    If there are vulnerabilities in these programs there may be a security
    risk to the system. FreeBSD makes no guarantee about the security of
    ports included in the Ports Collection. Please type 'make deinstall'
    to deinstall the port if this is a concern.

    For more information, and contact details about the security
    status of this software, see the following webpage:
http://www.isc.org/index.pl?sw/bind/bind9.3.php

```

Looks like I also need to configure a random sources as well

```

# cd /etc/namedb
# rndc-confgen -a
Write key file "/etc/namedb/rndc.key"

```

Now to configure BIND. The first task is to configure named.conf.. It looks like the Bind installation has provided a handy shell script to generate the localhost zone files for IPv4 and IPv4:

```

# /bin/sh make-localhost
# ls ./master
localhost-v6.rev      localhost.rev

```



```

XH0ti vUBDe00oT44rmt5HP3ZGKXCnwa1Nj OZ
pJwak9AXi K4Vmht+8cz0dEg8FDDR36bn7Qn5
V1Nf Jz9GRKRfMbT/pvfkFdW/aELVbarZ50
R6okHFpdTstS3aQhkUSnEG7Lyj QSXI Q==
se.          3600      IN      DNSKEY  256 3 5  AQPX2HyqyQ/hZSo+Ra2l 2q10th6/3f34L42
+vXJl mGduyp0UANYAKdT6x/ct0M9HgX2wh38
ZTB1T1vtxGmVMnHNwE6JSzZM9w0FAwnNh7aS
j z0Wwsj l Jkm6c+Q+5e3Suy05H2NnYxgZXnWU
ChGpgDA5tTYD9p55VH6qF7okHAddew==
se.          3600      IN      DNSKEY  256 3 5  AQPPrfmP4Rnc6pFUuHwdi AsCsAtl l hhgG3+3k
+52j E5507LAJw7ve9vv7he2yNrQl Hy1cl R/m
l GAwp0i j NxAyOH1eNhrj 7xzXBAqZda7bWrFZ
ON39NGsNUVFFEp0qoD6+WgMh4aesl j vZ2tY2
MGBmLZuHQDtp6SKcV6Ty6Zl Zrrx11w==
AwEAAaxPMcR2x0HbQV4WeZB6oEDX+r0QM65K
bhTj rW1ZaArMPhEZZe3Y9i fgEuq7vZ/zGZud
EGNWy+JZzusOl Uptwgj GwhUS1558Hb4JKUbb
OTcM 8pwXl j OEi X3oDFVmj H0444gLkBOUKUf
/mC7HvfwYH/Be22GnCl ri nKJp10g4yww09Wg
l Mk7j bfW33gUKvi rThr25GL7STOUzBb5Usxt
8l gnyTUHs 1t3JwCY5hKZ6CqFxmAVZP20i gT
i xi n/1LcrgX/KMEGd/buvF4qJCydui eHukuY
3H4XMAcR+xi a2nl UPvm/oyWR8BW/hWdzOvnS
CThl Hf3xi Yl eDb t/o10TQ09A0=
se.          3600      IN      RRSIG   DNSKEY  5 1 3600 20060915054256 20060909010558
32327 se.
mc5o/4wR1ttfugzj MT8Kyh08P2HcZSKdg1hzA
VRqFGxi Bt6+xtm6Jki Yui KLxFkh9SG/EFx+v
aj uQa3aGQFPo5GSuEl L/+4OUHLoEFz/Fu+Z
rLEl l uNYcKj RshazRmgXPrZP8mS2Ykp7l Vs
fRnGUxl F1utN6bi vvdI qm81KP174=
se.          3600      IN      RRSIG   DNSKEY  5 1 3600 20061014000000 20060831093257
17686 se.
dS9hM5y4z/tWc5MPAAti rei dpfLdu+oKzc60
i dV7LEt5GYN1a0D9RcGj OFZMZ0gu4zyr7ri
W/FovBB4anGtzi YXbxcpsqSRYs0FOci kml p1
vYa9GQbfffUQBau8Gj ckdSU/l 8LsGi Ai PI Pm9
XI S0a8oYZo6eVCTPtCi XEJaRYuwoEAUJ8xQ3
akBxPyPqol fX96o5Y0U0W0twdAS08NI XG897
Yp1FnDl azPhn5guWSraoovn3MG4j vM3ruBpn
Lzv3ZZVew4rWZ7RXmG76mwg461vgj hl +3cQf
fzAi ffl qr+DSx3Le2dnJbsSESGKRMrTeOgf8
78ni BGwOr0aj VbFUml Q==
;; AUTHORITY SECTION:
se.          172800     IN      NS       a. ns. se.
se.          172800     IN      NS       b. ns. se.
se.          172800     IN      NS       c. ns. se.
se.          172800     IN      NS       d. ns. se.
se.          172800     IN      NS       e. ns. se.
se.          172800     IN      NS       f. ns. se.
se.          172800     IN      NS       g. ns. se.
se.          172800     IN      NS       h. ns. se.
se.          172800     IN      NS       i. ns. se.
se.          172800     IN      RRSIG   NS 5 1 172800 20060916234318 20060910050559 32327 se.
OpfexJaa98S1Fd0wEmI dc687K100CW0oKMG0hOhMTaR
KE0SmVl 1WvDsRAEtT4xj s71orLzj i XOR8e33QuX9nDs
6E8110F5W4HfCe2dSURCVdvMUI ql eB21Roq/i yOj Ji k
vFhI MF2feX6+Oj zFnDokA3SDNemWUKZj cX7LUwR PLg=
;; Query time: 471 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Sep 10 15:41:29 2006
;; MSG SIZE rcvd: 1652

```

The critical line of output here was:

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 7, AUTHORITY: 10, ADDITIONAL: 1
```

The “ad” flag (“ad stands for “Authenticated Data” – which appears to be a rather important flag for DNSSEC!) is missing from the response flags. I expected this, as the configuration file omitted to load any trust keys.

This manual configuration of trust keys is perhaps the most frustrating and broken part of DNSSEC, and I suspect is the characteristic that will be the major factor in the demise of DNSSEC, if it turns out that DNSSEC fails to gain self-sustaining levels of deployment. The DNS root is not signed, nor are many, if not most, of the top level domains immediately under the root. And yet, somehow, I need to tell my resolver what zone keys should be trusted as roots of a trust model. Frankly, I don’t have a clue as to which key values I should trust and which I should reject. So to make the

resolver work I'll take an amazingly insecure short cut and place the DNSKEY for the .se domain I just retrieved into my named configuration file.

```
# tail named.conf
trusted-keys {
"se." 257 3 5 "AwEAAaxPMcR2x0HbQV4WeZB6oEDX+r0QM65KbhTj rW1ZaARmPhEZZe3Y
9i fgEuq7vZ/zGZUdEGNWy+JZzus0I Uptwgj GwhUS1558Hb4JKUbb0TcM
8pwXl j OEi X3oDFVmj H0444gLkBOUKUf/mC7HvfWYH/Be22GnCl ri nKJp
10g4yWz09Wgl Mk7j bfW33gUKvi rThR25GL7STQUzBb5Usxt8l gnyTUHs
1t3JwCY5hKZ6CqFxmAVZP20i gTi xi n/1LcrgX/KMEGd/buvF4qJCydui
eHukuY3H4XMAcR+xi a2nI UPvm/oyWR8BW/hWdz0vnSCTHl HF3xi Yl eDb
t/o10TQ09A0=";
};
```

Once more I'll start up the resolver daemon:

```
# /usr/sbin/named -c /etc/namedb/named.conf
# tail /var/log/messaging
named[29605]: starting BIND 9.3.2-P1 -c /etc/namedb/named.conf -d 3
named[29605]: command channel listening on 127.0.0.1#953
named[29605]: command channel listening on ::1#953
named[29605]: running
```

Now lets perform a test on .se.

```
# dig +dnssec +multiline DNSKEY se. @127.0.0.1
; <<>> DiG 9.3.2-P1 <<>> +dnssec +multiline DNSKEY se. @127.0.0.1
; (1 server found)
; global options: printcmd
; Got answer:
; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 45272
; flags: qr rd ra ad; QUERY: 1, ANSWER: 7, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; QUESTION SECTION:
; se. IN DNSKEY
; ANSWER SECTION:
se. 3600 IN DNSKEY 256 3 5 (
AQ0tFn92ppHBGvcCKDMJQ+KZE+N0owsHbj PB4mKruGCz
VhKrdl k0gu7Wj gj pw/5mVOXZel 5zKmt1I EXwO6MvrLyc
shUN3SDI m/hL6e9I JBCdyd2BVSqLAXs4eG1YPk5SzBGV
eKSNMxYj ekno/BT7cj 38nCdb1PLusP/wRLds0YI G7Q==
) ; key id = 17474
se. 3600 IN DNSKEY 256 3 5 (
AQP4Rkyd28gj 8v445LECVuJyRMEc+S2evl qXH0ti vUB
De0oT44rmt5HP3ZGKXCnwa1Nj OZpJwak9AXi K4Vmht+
8cz0dEg8FdDR36bn7Qn5V1NfJz9GRKRfMbt/pvfkFdW
/aELVbaRz50R6okHFpdTstS3aQhkUSnEG7Ljy QSXI Q==
) ; key id = 54245
se. 3600 IN DNSKEY 256 3 5 (
AQPhX2HyqyQ/hZSo+Ra2l 2q10th6/3F34L42+vXJl mGd
uyPOUANYAKdT6x/ctOM9HgX2wh38ZTB1T1vtxGmVmnHN
wE6JSzZM9wOfAwnNh7aSJ z0WWSj l Jkm6c+Q+5e3Suy05
H2NnYxgZXnWUChGpgDA5tTYD9p55VH6qF7okHAdew==
) ; key id = 32327
se. 3600 IN DNSKEY 256 3 5 (
AQPrFmP4Rnc6pFUuHwdi AsCsAtI l hhgG3+3k+52j E550
7LAJw7ve9vv7he2yNrQl Hy1cl R/ml GAwpOij NxAyOH1e
NHRj 7xzXBAqZda7bWrfZON39NGsNUVFFEp0qoD6+WgMh
4aeslj vZ2tY2MGBmLZuHQDtp6SKcV6Ty6ZI Zrrx11w==
) ; key id = 20825
se. 3600 IN DNSKEY 257 3 5 (
AwEAAaxPMcR2x0HbQV4WeZB6oEDX+r0QM65KbhTj rW1Z
aARmPhEZZe3Y9i fgEuq7vZ/zGZUdEGNWy+JZzus0I Upt
wgj GwhUS1558Hb4JKUbb0TcM8pwXl j OEi X3oDFVmj H04
44gLB0UKUf/mC7HvfWYH/Be22GnCl ri nKJp10g4yWz0
9Wgl Mk7j bfW33gUKvi rThR25GL7STQUzBb5Usxt8l gny
TUHs1t3JwCY5hKZ6CqFxmAVZP20i gTi xi n/1LcrgX/KM
EGd/buvF4qJCydui eHukuY3H4XMAcR+xi a2nI UPvm/oy
WR8BW/hWdz0vnSCTHl HF3xi Yl eDbt/o10TQ09A0=
) ; key id = 17686
se. 3600 IN RRSIG DNSKEY 5 1 3600 20060915054256 (
20060909010558 32327 se.
mc5o/4wR1tfugzj MT8Kyh08P2HcZSKdg1hzAVRqFGxi B
t6+xtm6Jki Yui KLxFkh9SG/EFx+vaj uQa3aGQFPo5GSu
EIL/+40UHL0eFz/Fu+ZrLEl i uNYCkj RshazRmgXPrZP
```

```

se. 3600 IN RRSIG DNSKEY 5 1 3600 20061014000000 (
      20060831093257 17686 se.
      dS9hM5y4z/tWc5MPAAAti rei dpfLdu+oKzc60i dV7LEt5
      GYN1a0D9RcGj OFZMZ0gu4zyr7ri W/FovBB4anGtzI YXb
      xcpsqSRYSOF0Ci kml p1vYa9GQbfFUQBau8Gj ckdSU/I 8
      LsGi Ai PI Pm9XI SOa8oYZo6eVCTPtCi XEJaRYuwoEAUJ8
      xQ3aKBxPyPqoi fX96o5YOU0W0twdAS08NI XG897Yp1Fn
      DI azPhn5guWSraoovn3MG4j vM3ruBpnLzv3ZZVew4rWZ
      7RXmG76mwg461vgj hI +3cQffzAi ffl qr+DSx3Le2dnJb
      sSESGKRMrtE0gf878ni BGw0rOaj VbFUmI Q== )

;; Query time: 423 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Sep 10 15:45:30 2006
;; MSG SIZE rcvd: 1343

```

The ad flag is set in the answer section, indicating that the answer was authenticated. Success! What did the DNSSEC debug log make of this? It seems that the local resolver had managed to decide to trust this key.

```

# tail /var/log/named-dnssec.log
validating @0x824c000: se DNSKEY: starting
validating @0x824c000: se DNSKEY: attempting positive response validation
validating @0x824c000: se DNSKEY: verify rdataset: success
validating @0x824c000: se DNSKEY: signed by trusted key; marking as secure
validator @0x824c000: dns_validator_destroy

```

Now who else has published keys that I may want to configure? This may be a fine question, but, once more, I have no idea what the answer should be. It seems that this technology was intended to be a top-down comprehensively signed structure, where all I would need use to seed DNSSEC was the current key for the DNS root, and all other DNS keys could be validated by performing a backward walk towards the root. And for as long as the root remains unsigned, and for as long as the next level down, the top level domains remain unsigned, then anyone attempting to perform DNSSEC checks on resolver outcomes is pretty much indulging in a pointless exercise.

So right now I have a DNSSEC-aware resolver, and because I have decided to dynamically load a DNS zone key arbitrarily, I can perform key resolution on a limited set of domains under .se. Precisely which set of domains under .se can be validated using DNSSEC I can't tell in advance. Some work in terms of validation, some do not. I cannot tell if a response for a query relating to a sub-domain of .se should have additional authentication data or not. As far as I can tell this is not a terribly useful outcome, so I'll spend a little time looking around for some more trust anchors. The RIPE NCC publish a set of trust anchors at <<https://www.ripe.net/projects/dnssec/keys/>>. I'll load these as trust anchors into my named.conf file and test these:

```

# dig +dnssec www.ripe.net A @127.0.0.1
; <<>> DiG 9.3.2-P1 <<>> +dnssec www.ripe.net A @127.0.0.1
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 51789
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;www.ripe.net.          IN      A

;; ANSWER SECTION:
www.ripe.net.          600    IN      CNAME   kite-www.ripe.net.
www.ripe.net.          600    IN      RRSIG   CNAME 5 3 600 20061010051144 20060910051144 15917
ripe.net. BcSb934sDbz8Gi ouhj T8z0Aum5v1hBq7+82JhXCucdORFsAqHtk1cRSn
Oxhp89Mu/EbugkrBfbAzoV6l j BhR0DL1/j LD2pQqyZ8j WWre4EI cAo7s
wE/xSol rrutwyA5RB090dtGI NXOJbarkaE9LBVQb1v0MDKdGj pNzoNPW WVxsZ7i oddqCJax5+r7W+q1HI mhSKU8f
kite-www.ripe.net.    3600   IN      A        193.0.0.214
kite-www.ripe.net.    3600   IN      RRSIG   A 5 3 172800 20061010051144 20060910051144 15917
ripe.net. OBF0I ty7FDxAppqEMBGCOMbqgFP+ePVKeL/ZLI BmV9j 6j 1B97U++/6NI Q
pqw8PJYx826y/7tCsT3L0m3dnzJdVkcE9No/VPtB/kyRcQLRI YnSL4F8
gT5sw5H7NBD4+w8orwfm1GG9Lqmff2qOrZU7ti RP1i 7U9/ASLTJ 2i /9H D+ASg/I ye66pcLdOoPQVCNfCL1m9GFH

;; Query time: 1657 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Sep 11 10:56:07 2006
;; MSG SIZE rcvd: 460

```

Again the “ad” flag bit appears to be set, so this response appears to be valid. How about a non-existent name within the ripe.net zone?

```
# dig +dnssec A nonexistent.ripe.net @127.0.0.1
; <<>> DiG 9.3.2-P1 <<>> +dnssec A nonexistent.ripe.net
; global options: printcmd
; Got answer:
; -->HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 42805
; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; QUESTION SECTION:
;nonexistent.ripe.net.          IN      A

; AUTHORITY SECTION:
ripe.net.          3600    IN      SOA     ns-pri.ripe.net. ops.ripe.net. 2006091101 43200 7200
1209600 7200
ripe.net.          3600    IN      RRSIG   SOA 5 2 172800 20061011051206 20060911051206 15917
ripe.net. jKMONWMIg4YnzZyotmcRJ9VsbHdNTkEyDbrV/RYnq9urgu8DP0ca6TSC
rZQpwEaMStH4PKHfcfo9hx3uKCP1Hcl +Dx8U91KwWZQ5s+8o3FAW76I9
FhZLQmg9GVDYU202VAcGWvmuYFe0uhl 6Br2d7h3zSGi vkax81Vd40Igd sGA2sb2NacT+TF3Sm0ots01JJPhs1VfW
ripe.net.          3600    IN      NSEC    adsl.ripe.net. A NS SOA MX RRSIG NSEC DNSKEY
ripe.net.          3600    IN      RRSIG   NSEC 5 2 7200 20061011051206 20060911051206 15917
ripe.net. koQhAbkByga7YCGwu6pGuqi k9y1j 4aeRU9gbkpA7YJULj pYj AnFvOdG
+4xBydI T7j T/9NOYF90pKRdp3Cgv8ZKTMfI yD/rkyxWQXYuj zhPnLS1H
iOwv+Wj hRMv5oJ7HmZSI CrcJ34zq+SWMhKuCOH60hnrRay7eTTpQqt1dd +ptj ZsXctw2bl tY0826a2ukRrD8HRI KJ
niletest.ripe.net. 3600    IN      NSEC    np-consol.e.ripe.net. A RRSIG NSEC
niletest.ripe.net. 3600    IN      RRSIG   NSEC 5 3 7200 20061011051206 20060911051206 15917
ripe.net. wr1LMfpPhtal xe1Kcx/SSmB/Wq9cez+Dey00PX05i do2nMD8pdpSQzGs
168Y0oI 5JaCOfP6dgj tTFq3AqabF+egPJwmQbJS2hRGYPXWq74UQS81L
SG2hs6Bj orj /MNNL/5eMoPrDeG8I ALTI cavfGfI Goet0ToKwa0SU+bEp OqK3/LMUuQ9i esUzUx0x8KWoR+fG/Mt9

; Query time: 1112 msec
; SERVER: 127.0.0.1#53(127.0.0.1)
; WHEN: Tue Sep 12 08:11:14 2006
; MSG SIZE rcvd: 752
```

The “ad” bit is set, which tells me that the answer came from the original zone file, and the relevant NSEC record tells me that there are no names between niletest.ripe.net and np-consol.e.ripe.net. How about querying nlnetlabs.nl?

```
# dig +dnssec DNSKEY nlnetlabs.nl @127.0.0.1
; <<>> DiG 9.3.2-P1 <<>> +dnssec DNSKEY nlnetlabs.nl @127.0.0.1
; (1 server found)
; global options: printcmd
; Got answer:
; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 24068
; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 1

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; QUESTION SECTION:
;nlnetlabs.nl.          IN      DNSKEY

; ANSWER SECTION:
nlnetlabs.nl.          86386   IN      DNSKEY  257 3 5
AQPzzTWMz8qSWI Ql fRnPkcx2Bi VmkVN6LPupO3mbz7FhLSnm26n6i G9N
Lby97Ji 453aWZY3M5/xJBS0S2vWtco2t8CO+xe01bc/d6ZTy32DHchpW
6rDH1vp86LI +ha0tmwyy9QP7y2bVw5zSbFCrefk8qCUBgFhm9bHZMG1U BYtEIQ==
nlnetlabs.nl.          86386   IN      RRSIG   DNSKEY 5 2 86400 20060902172237 20060803172237 43791
nlnetlabs.nl. PcxHI UzBwPgqx7uVzjk gN5KI 3Z4yq3I e5708TQu3A2w/J+2/eLQdbBTC
R9yI d6We2YtRTwNfPumdUVFQWTMuRj 6tg8TVggSYVv/uHs2ySC24NdpT
9QXsMwJnKmj 59DmhroLvH8svHnZ7W1I 8y+tvS7pWWE2twOKF+pPKYVvK1 61g=

; AUTHORITY SECTION:
nlnetlabs.nl.          86386   IN      NS       open.nlnetlabs.nl.
nlnetlabs.nl.          86386   IN      NS       omval.tednet.nl.
nlnetlabs.nl.          86386   IN      NS       ns7.domain-registry.nl.
nlnetlabs.nl.          86386   IN      RRSIG   NS 5 2 86400 20060902172237 20060803172237 43791
nlnetlabs.nl. fcoLufPkFtOhl zDp6cvT4ZXEfnHAI D0sWz10Wr5cnsSDW9/TKmEi PerYy
sb0HocNNI Tg1GI +wzi VVI MEK87CJYI 11 +E3I XOLZrOfz/14I z0I ZI r7N
DEwwYMGlyFopWpKh3ThJoi otVktEkJHOzF/KI e+OZHFLol fcf8dVMOTu ntg=

; Query time: 17 msec
; SERVER: 127.0.0.1#53(127.0.0.1)
; WHEN: Mon Sep 11 11:05:41 2006
; MSG SIZE rcvd: 611
```

Actually this one looks like it was never going to work – the dates on the RRSIG Resource Records indicate key expiry on the 2nd September, and I'm conducting this test on the 11th of September. The key has expired for this zone.

So it looks like the DNSSEC resolver works, after a fashion, although I have to say that the treatment of trust anchors is relatively useless for all but the most trivial of exercises.

Can I do anything better in terms of discovery of trust keys in the absence of a comprehensively signed DNS structure starting at the root? It appears that RFC4431 contains a potential answer to this problem of fragmented trust anchors. This document describes a DNSSEC Lookaside Validation (DLV) DNS Resource Record. In this approach the trust anchors are retrieved from a lookaside location, and by configuring the credentials of the DLV publisher into the resolver, then the resolver will be aware to recognise as trust anchors all those zones that have registered with the DLV publisher. The configuration details for the ISC-operated DLV service are published at <<http://www.isc.org/index.pl?/ops/dlv/>>. Using the configuration steps at that URL my named.conf now has an additional trusted key, this one for dlv.isc.org.

So into the trusted-keys section of named.conf I've added

```
dlv.isc.org. 257 3 5
"AQPap3+2+i tqZpuuj LA/j /eI EYl s9HGo9W8rm1uVpW0zZX4vi yFQyGL91YkGUA2uTQ1ZHwbJ36KYI Jpt8ZZ+
tuI smJw9/AUnZl PgwCfq5C2MOGVh33nF60k67ppi apMYS0aDFbAQf5Vcc3L+BwfJvkXsZK73nD3gBEcdcmuJeJ eQ==";
```

and into the options section I've added

```
dnssec-lookaside . trust-anchor dlv.isc.org;
```

Now - how do I know this lookaside is working? I don't! I don't know who has lodged their zone keys with the operators of this service, nor can I see from any of the online information where such information can be obtained. So how am I to know how to test this lookaside service? Which domains have I implicitly added to my trusted set? It seems to me that there is some critical information that is lacking here, and in its current state the lookaside exercise is one that looks rather like one of futility.

So in the absence of any decent details, lets try a guess - if this lookaside service, operated by ISC, is working, then I should expect that a query for the A record of www.isc.org should validate via this lookaside mechanism - right?

Wrong.

```
# dig +dnssec A www.isc.org
; <<>> DiG 9.3.2-P1 <<>> +dnssec A www.isc.org
; global options: printcmd
; Got answer:
; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 15499
; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; QUESTION SECTION:
;www.isc.org.                IN      A

; ANSWER SECTION:
www.isc.org.                600     IN      A      204.152.184.88
www.isc.org.                600     IN      RRSIG  A 5 3 600 20061010193344 20060910193344 57956
isc.org. T/1vQD0mTrmf7fKAJ8gkRH4t/J2zqCx4YPkLzj PAnexJbj sxrTkDq/uo
WgL050sXHmHxVTPwZA96+Q+oFkoq2/MXBi YbnM3U1xTFFTFWT3fcuxmL
Hc6kCa9Ubd/7hPwt/VrvB38Y9pHLeDj 7Ehr8+EfaR0cdJzuvpfSEN/SS 084=

; Query time: 1593 msec
; SERVER: 127.0.0.1#53(127.0.0.1)
; WHEN: Tue Sep 12 08:01:02 2006
; MSG SIZE rcvd: 223

#
```

The "ad" flag is missing in the response. What went wrong?

The log file relating to this query indicates that dl.v.isc.org appears to validate, but that www.isc.org is not included in the DLV service (or at least that's my interpretation of the following voluminous debug output):

```
validating @0x824b000: . NS: starting
validating @0x824b000: . NS: looking for DLV
validating @0x824b000: . NS: plain DNSSEC returns unsecure (.): looking for DLV
validating @0x824b000: . NS: looking for DLV dl.v.isc.org
validating @0x824b000: . NS: finddlvsep: creating fetch for dl.v.isc.org DLV
validating @0x824b000: . NS: DLV lookup: wait
validating @0x824b800: dl.v.isc.org DLV: starting
validating @0x824b800: dl.v.isc.org DLV: attempting negative response validation
validating @0x824b800: dl.v.isc.org DLV: nsecvalidate: creating validator for dl.v.isc.org SOA
  validating @0x824f000: dl.v.isc.org SOA: starting
  validating @0x824f000: dl.v.isc.org SOA: attempting positive response validation
  validating @0x824f000: dl.v.isc.org SOA: get_key: creating fetch for dl.v.isc.org DNSKEY
validating @0x824f800: www.isc.org A: starting
validating @0x824f800: www.isc.org A: looking for DLV
validating @0x824f800: www.isc.org A: plain DNSSEC returns unsecure (.): looking for DLV
validating @0x824f800: www.isc.org A: looking for DLV www.isc.org.dlv.isc.org
validating @0x824f800: www.isc.org A: finddlvsep: creating fetch for www.isc.org.dlv.isc.org DLV
validating @0x824f800: www.isc.org A: DLV lookup: wait
validating @0x825a000: dl.v.isc.org DNSKEY: starting
validating @0x825a000: dl.v.isc.org DNSKEY: attempting positive response validation
validating @0x825a000: dl.v.isc.org DNSKEY: verify rdataset: success
validating @0x825a000: dl.v.isc.org DNSKEY: signed by trusted key; marking as secure
validator @0x825a000: dns_validator_destroy
  validating @0x824f000: dl.v.isc.org SOA: in fetch_callback_validator
  validating @0x824f000: dl.v.isc.org SOA: keyset with trust 7
  validating @0x824f000: dl.v.isc.org SOA: resuming validate
  validating @0x824f000: dl.v.isc.org SOA: verify rdataset: success
  validating @0x824f000: dl.v.isc.org SOA: marking as secure
validating @0x8258000: www.isc.org.dlv.isc.org DLV: starting
validating @0x8258000: www.isc.org.dlv.isc.org DLV: attempting negative response validation
validating @0x8258000: www.isc.org.dlv.isc.org DLV: nsecvalidate: creating validator for
dl.v.isc.org SOA
  validating @0x8258800: dl.v.isc.org SOA: starting
  validating @0x8258800: dl.v.isc.org SOA: attempting positive response validation
  validating @0x8258800: dl.v.isc.org SOA: keyset with trust 7
  validating @0x8258800: dl.v.isc.org SOA: verify rdataset: success
  validating @0x8258800: dl.v.isc.org SOA: marking as secure
  validator @0x8258800: dns_validator_destroy
validating @0x8258000: www.isc.org.dlv.isc.org DLV: in authvalidated
validating @0x8258000: www.isc.org.dlv.isc.org DLV: resuming nsecvalidate
validating @0x8258000: www.isc.org.dlv.isc.org DLV: nsecvalidate: creating validator for
isc.org.dlv.isc.org NSEC
  validating @0x8258800: isc.org.dlv.isc.org NSEC: starting
  validating @0x8258800: isc.org.dlv.isc.org NSEC: attempting positive response validation
  validating @0x8258800: isc.org.dlv.isc.org NSEC: keyset with trust 7
  validating @0x8258800: isc.org.dlv.isc.org NSEC: verify rdataset: success
  validating @0x8258800: isc.org.dlv.isc.org NSEC: marking as secure
  validator @0x824f000: dns_validator_destroy
validating @0x824b800: dl.v.isc.org DLV: in authvalidated
validating @0x824b800: dl.v.isc.org DLV: resuming nsecvalidate
validating @0x824b800: dl.v.isc.org DLV: nsecvalidate: creating validator for dl.v.isc.org NSEC
  validating @0x824f000: dl.v.isc.org NSEC: starting
  validating @0x824f000: dl.v.isc.org NSEC: attempting positive response validation
  validating @0x824f000: dl.v.isc.org NSEC: keyset with trust 7
  validating @0x824f000: dl.v.isc.org NSEC: verify rdataset: success
  validating @0x824f000: dl.v.isc.org NSEC: marking as secure
  validator @0x824f000: dns_validator_destroy
validating @0x824b800: dl.v.isc.org DLV: in authvalidated
validating @0x824b800: dl.v.isc.org DLV: looking for relevant nsec
validating @0x824b800: dl.v.isc.org DLV: nsec proves name exists (owner) data=0
validating @0x824b800: dl.v.isc.org DLV: resuming nsecvalidate
validating @0x824b800: dl.v.isc.org DLV: nonexistence proof found
validator @0x824b800: dns_validator_destroy
  validator @0x8258800: dns_validator_destroy
validating @0x8258000: www.isc.org.dlv.isc.org DLV: in authvalidated
validating @0x8258000: www.isc.org.dlv.isc.org DLV: looking for relevant nsec
validating @0x8258000: www.isc.org.dlv.isc.org DLV: nsec range ok
validating @0x8258000: www.isc.org.dlv.isc.org DLV: resuming nsecvalidate
validating @0x8258000: www.isc.org.dlv.isc.org DLV: in checkwildcard: *.isc.org.dlv.isc.org
validating @0x8258000: www.isc.org.dlv.isc.org DLV: looking for relevant nsec
validating @0x8258000: www.isc.org.dlv.isc.org DLV: nsec range ok
validating @0x8258000: www.isc.org.dlv.isc.org DLV: nonexistence proof found
validator @0x8258000: dns_validator_destroy
validating @0x824b000: . NS: in dlvfetched: ncache nxrrset
validating @0x824b000: . NS: DLV not found
validating @0x824b000: . NS: marking as answer
validator @0x824b000: dns_validator_destroy
validating @0x824f800: www.isc.org A: in dlvfetched: ncache nxdomain
validating @0x824f800: www.isc.org A: looking for DLV isc.org.dlv.isc.org
validating @0x824f800: www.isc.org A: finddlvsep: creating fetch for isc.org.dlv.isc.org DLV
```

```

validating @0x824f800: www.isc.org A: DLV lookup: wait
validating @0x824f000: isc.org.dlv.isc.org DLV: starting
validating @0x824f000: isc.org.dlv.isc.org DLV: attempting negative response validation
validating @0x824f000: isc.org.dlv.isc.org DLV: nsecvalidate: creating validator for dlv.isc.org SOA
validating @0x81cd000: dlv.isc.org SOA: starting
validating @0x81cd000: dlv.isc.org SOA: attempting positive response validation
validating @0x81cd000: dlv.isc.org SOA: keyset with trust 7
validating @0x81cd000: dlv.isc.org SOA: verify rdataset: success
validating @0x81cd000: dlv.isc.org SOA: marking as secure
validator @0x81cd000: dns_validator_destroy
validating @0x824f000: isc.org.dlv.isc.org DLV: in authvalidated
validating @0x824f000: isc.org.dlv.isc.org DLV: resuming nsecvalidate
validating @0x824f000: isc.org.dlv.isc.org DLV: nsecvalidate: creating validator for
isc.org.dlv.isc.org NSEC
validating @0x81cd000: isc.org.dlv.isc.org NSEC: starting
validating @0x81cd000: isc.org.dlv.isc.org NSEC: attempting positive response validation
validating @0x81cd000: isc.org.dlv.isc.org NSEC: keyset with trust 7
validating @0x81cd000: isc.org.dlv.isc.org NSEC: verify rdataset: success
validating @0x81cd000: isc.org.dlv.isc.org NSEC: marking as secure
validator @0x81cd000: dns_validator_destroy
validating @0x824f000: isc.org.dlv.isc.org DLV: in authvalidated
validating @0x824f000: isc.org.dlv.isc.org DLV: looking for relevant nsec
validating @0x824f000: isc.org.dlv.isc.org DLV: nsec proves name exists (owner) data=0
validating @0x824f000: isc.org.dlv.isc.org DLV: resuming nsecvalidate
validating @0x824f000: isc.org.dlv.isc.org DLV: nonexistence proof found
validator @0x824f000: dns_validator_destroy
validating @0x824f800: www.isc.org A: in dlvfetched: ncache nxrrset
validating @0x824f800: www.isc.org A: looking for DLV org.dlv.isc.org
validating @0x824f800: www.isc.org A: DNS_R_COVERINGNSEC
validating @0x824f800: www.isc.org A: covering nsec: not in range
validating @0x824f800: www.isc.org A: finddlvsep: creating fetch for org.dlv.isc.org DLV
validating @0x824f800: www.isc.org A: DLV lookup: wait
validating @0x824f000: org.dlv.isc.org DLV: starting
validating @0x824f000: org.dlv.isc.org DLV: attempting negative response validation
validating @0x824f000: org.dlv.isc.org DLV: nsecvalidate: creating validator for dlv.isc.org SOA
validating @0x81cd000: dlv.isc.org SOA: starting
validating @0x81cd000: dlv.isc.org SOA: attempting positive response validation
validating @0x81cd000: dlv.isc.org SOA: keyset with trust 7
validating @0x81cd000: dlv.isc.org SOA: verify rdataset: success
validating @0x81cd000: dlv.isc.org SOA: marking as secure
validator @0x81cd000: dns_validator_destroy
validating @0x824f000: org.dlv.isc.org DLV: in authvalidated
validating @0x824f000: org.dlv.isc.org DLV: resuming nsecvalidate
validating @0x824f000: org.dlv.isc.org DLV: nsecvalidate: creating validator for ns-ext.dlv.isc.org
NSEC
validating @0x81cd000: ns-ext.dlv.isc.org NSEC: starting
validating @0x81cd000: ns-ext.dlv.isc.org NSEC: attempting positive response validation
validating @0x81cd000: ns-ext.dlv.isc.org NSEC: keyset with trust 7
validating @0x81cd000: ns-ext.dlv.isc.org NSEC: verify rdataset: success
validating @0x81cd000: ns-ext.dlv.isc.org NSEC: marking as secure
validator @0x81cd000: dns_validator_destroy
validating @0x824f000: org.dlv.isc.org DLV: in authvalidated
validating @0x824f000: org.dlv.isc.org DLV: looking for relevant nsec
validating @0x824f000: org.dlv.isc.org DLV: nsec proves name exist (empty)
validating @0x824f000: org.dlv.isc.org DLV: resuming nsecvalidate
validating @0x824f000: org.dlv.isc.org DLV: nonexistence proof found
validator @0x824f000: dns_validator_destroy
validating @0x824f800: www.isc.org A: in dlvfetched: ncache nxrrset
validating @0x824f800: www.isc.org A: looking for DLV dlv.isc.org
validating @0x824f800: www.isc.org A: DLV not found
validating @0x824f800: www.isc.org A: marking as answer
validator @0x824f800: dns_validator_destroy

```

That's a massive amount of diagnostic output for an authentication failure!

I think about I've gone about as far as I can with the resolver configuration. If I, as an operator of a DNS resolver, am prepared to spend large amounts of time to track down trust anchor keys of DNS zones, and regularly monitor their currency, and re-fetch as necessary, then I can validate a subset of the DNS. Given the relatively small amount of time I'm prepared to spend on this task I suspect that this subset of the DNS will be exceptionally small. If I am prepared to trust a lookaside service then I can potentially validate a greater fraction of the DNS name space, but, frankly, I have no idea what is behind the DLV curtain, and as a consumer who is concerned enough about the validity of the DNS to arm my resolver with DNSSEC, then in its current incarnation DLV is not doing much for me at all.

Interestingly, if I want to use DNSSEC at the application level I can't see any readily available tools at all right now. The 'standard' application level queries to the DNS are through the `gethostbyname()` library call, and this interface has no clear way in which to add DNSSEC validation switches. It would appear that if I want to write an application that looks at

DNSSEC validation outcomes of DNS queries then I need to write my own code right down to the DNS call primitives. Personally, I liked the `gethostbyname()` abstraction of the interface to the DNS, and I'd strongly prefer the availability of a similar abstraction of a DNSSEC-aware name resolution call.

Even if I had a `gethostbyname_with_dnssec_validation()` call the situation is still not entirely clear – what should my application do if the DNSSEC validation check fails? Should it generate a popup dialogue with the user, alerting the user to the validation problem and requesting whether to proceed or not? The impression that many folk have is that the standard user response to certificate invalidity pop-up warnings is to direct the application to proceed in any case. If these were to be the case with DNSSEC pop-ups, and users would simply say “proceed” in any case then there is a real issue about the true value of DNSSEC to the end user. And, of course the DNS is used in various internal functions, where there is no clear mechanism for user alerts and intervention. If such a daemon detects a DNSSEC invalidity condition should it refuse to continue, or log the event and continue in any case? The more generic question is what should happen once you arm an application with sufficient capability to check the authenticity of information received over a network. Should the validation provide a yes/no condition for continuing with the application, or should it be interpreted as a preference, or should it be disregarded? As I don't have the time in this exercise to construct a DNSSEC-aware application this remains an open question here. If DNSSEC were universally deployed, then the answer may be clearer. Given a situation of partial deployment of DNSSEC the absence of DNSSEC credentials in a response does not necessarily mean that there has been a faked DNS response, but on the other hand you simply cannot be sure.

Step 4 – Signing a Zone

The next part of the exercise with DNSSEC is to sign a zone. I'll set up a zone just for this purpose, so I'll call it “`dnssec.potaroo.net`”.

This zone is relatively simple:

```
$TTL 86400
$ORIGIN      dnssec.potaroo.net.
@           IN      SOA      dns0.potaroo.net.  glh.potaroo.net. (
                2006090803 ; Serial
                3h      ; Refresh
                15      ; Retry
                1w      ; Expire
                3h ) ; Minimum
;
;
; name servers
;
                IN      NS      dns0.potaroo.net.
                IN      NS      dns1.potaroo.net.
;
; subdomains
;
sub         IN      NS      dns0.dnssec.potaroo.net.
                IN      NS      dns1.dnssec.potaroo.net.
;
; zone A records
;
www         IN      A       203.50.0.6
bgp         IN      A       203.50.0.159
bgp2        IN      A       203.50.0.33
dns0        IN      A       203.50.0.18
dns1        IN      A       203.50.0.6
;
; wildcard
;
*           IN      A       203.50.0.18
```

The first task is to generate some keys to sign the zone. I'll choose a 1024 bit key size, and use split Key Signing Keys and Zone signing keys. The first key I'll generate is the Zone signing key. I'll use the RIPE tutorial for the command syntax for this command:

```
# dnssec-keygen -r/dev/random -a RSASHA1 -b 1024 -n ZONE dnssec.potaroo.net
Kdnssec.potaroo.net.+005+03755
```

The command has generated two files, one with the public key information in the form of a DNSKEY record, and the other is the description of the private key. Here's the public key file:

```
# cat Kdnssec.potaroo.net.+005+03755.key
dnssec.potaroo.net. IN DNSKEY 256 3 5 A008xvbN4hZ8bn926wpM8c9Uqqhqcf45v73k4J/YSu+6o/QsPCKwJoDY
xMH3s5Z0NJIgLUQscI ZZKDYVHPW3Txt59bHrn739osnQ80Rb0GVTH/Vi
//L3BGj ZrZr+PwH2Vb3wl hruj Mej 2m4E2Mth/Xj SDAhYZVWCNhJG0nP H6G6Ww==
```

I'll repeat this process for the key signing key, adding "-f KSK" to the command:

```
# dnssec-keygen -r/dev/random -f KSK -a RSASHA1 -b 1024 -n ZONE dnssec.potaroo.net
Kdnssec.potaroo.net.+005+29022
```

I now have four key files:

```
# ls Kdns*
Kdnssec.potaroo.net.+005+03755.key      Kdnssec.potaroo.net.+005+29022.key
Kdnssec.potaroo.net.+005+03755.pri vate  Kdnssec.potaroo.net.+005+29022.pri vate
```

AS per the tutorial instructions, I'll copy the public keys into the named master zone area, and include them into the dnssec.potaroo.net zone file

```
# tail /etc/namedb/master/dnssec.potaroo.net
;
;
; w i l d c a r d
;
*
      IN      A      203.50.0.18
$include Kdnssec.potaroo.net.+005+03755.key ; zone signing key
$include Kdnssec.potaroo.net.+005+29022.key ; key signing key
```

Before signing the zone, maybe I should check it for syntactic correctness:

```
# named-checkzone -D dnssec.potaroo.net dnssec.potaroo.net
zone dnssec.potaroo.net/IN: loaded serial 2006090802
dnssec.potaroo.net.                86400 IN SOA      dns0.potaroo.net.  gi h.potaroo.net.
2006090802 10800 15 604800 10800
dnssec.potaroo.net.                86400 IN NS       dns0.potaroo.net.
dnssec.potaroo.net.                86400 IN NS       dns1.potaroo.net.
dnssec.potaroo.net.                86400 IN DNSKEY   256 3 5
A008xvbN4hZ8bn926wpM8c9Uqqhqcf45v73k4J/YSu+6o/QsPCKwJoDY
xMH3s5Z0NJIgLUQscI ZZKDYVHPW3Txt59bHrn739osnQ80Rb0GVTH/Vi
//L3BGj ZrZr+PwH2Vb3wl hruj Mej 2m4E2Mth/Xj SDAhYZVWCNhJG0nP H6G6Ww== ; key id = 3755
dnssec.potaroo.net.                86400 IN DNSKEY   257 3 5
AQP5OR9BUnuQQ8i en6Wi baSsKddzZstW4TEuJrSzezQL79DFqHe0vVuh
Jr+9JM0mJuQUGj VcXDG1gBRQboi FJ6e+G6si bl KI kzXCLX709YqYtyv
1AMyEbYWLTwRvKoj ZSZr2LyKqeKGFqWdoA8a1M6XRuChBl wxMwo5l 5fs edl yYw== ; key id = 29022
*. dnssec.potaroo.net.             86400 IN A        203.50.0.18
bgp. dnssec.potaroo.net.           86400 IN A        203.50.0.159
bgp2. dnssec.potaroo.net.          86400 IN A        203.50.0.33
dns0. dnssec.potaroo.net.          86400 IN A        203.50.0.18
dns1. dnssec.potaroo.net.          86400 IN A        203.50.0.6
sub. dnssec.potaroo.net.           86400 IN NS       dns0.dnssec.potaroo.net.
sub. dnssec.potaroo.net.           86400 IN NS       dns1.dnssec.potaroo.net.
www. dnssec.potaroo.net.           86400 IN A        203.50.0.6
OK
```

Looks good enough! Now to sign the zone. Again I'll use a command format following the RIPE tutorial.

```
# /usr/local/sbin/dnssec-signzone -r /dev/random -o dnssec.potaroo.net.
-k Kdnssec.potaroo.net.+005+29022 dnssec.potaroo.net Kdnssec.potaroo.net.+005+03755.key
dnssec.potaroo.net.signed
```

I should see a signed zone file in the file dnssec.potaroo.net.signed:

```
# cat dnssec.potaroo.net.signed
; File written on Fri Sep 8 19:08:32 2006
; dnssec_signzone version 9.3.2
dnssec.potaroo.net.      86400  IN  SOA      dns0.potaroo.net.  gi h.potaroo.net. (
                        2006090803 ; serial
                        10800      ; refresh (3 hours)
                        15         ; retry (15 seconds)
                        604800     ; expire (1 week)
```

```

)
10800 ; minimum (3 hours)
)
86400 RRSIG SOA 5 3 86400 20061008080832 (
20060908080832 3755 dnssec.potaroo.net.
syLogFkxP1KI EkYp4PI c6qgW1Nr16powl zx+
VbpdA/erzxRdARd1l 77F56N7TB+v3aS82aLh
BLI N+fOMzHEo/JNWVI 0xj n95pRDd3gyZSoE+
aWG21MokMbTBxF2pYmFA1ENNKKK+pSXuXvsS
dAP+kcVqT6PF067+m2chsqbh+uA= )
86400 NS dns0.potaroo.net.
86400 NS dns1.potaroo.net.
86400 RRSIG NS 5 3 86400 20061008080832 (
20060908080832 3755 dnssec.potaroo.net.
p2kKLLK4gzl m8nkr4l pXyz4Fi rWWXti yXc5X/
Ns2NYC3CNYDNI RFHzEI 14RZ008R9z4aoQl f0
j Xi di JZ2BgxzmykVJUaA7AwGi rVtr+6wDJrd
i f9tm7UdYN2powrP9o2l qODKhwYk8i 4Dyj dd
9kwt7/x44ZECzEj 7w30GfW4uvy8= )
10800 NSEC *.dnssec.potaroo.net. NS SOA RRSIG NSEC DNSKEY
10800 RRSIG NSEC 5 3 10800 20061008080832 (
20060908080832 3755 dnssec.potaroo.net.
h75DS6C1l GLPRbqtz9+KV4oSui dA+Bdt6geq
q6NRrneNGA6Rr00FK4Td9A0S1+JpM3Kri DI 5
LkqQM7yMarC7aE3v/23i W9YqFv3Z6Ppj W7Ze
oEhaLNCV3kG4tVml LsoGEp/EWtgNTnXkJdkD
hW+o91s7XVnGm07m9JkU0u8sS2E= )
86400 DNSKEY 256 3 5 (
AQ08xvbN4hZ8bn926wpM8c9UqqhqcF45v73k
4J/Ysu+6o/QsPCKwJoDYxMH3s5ZONJI gLUQs
cl ZZKDYVHPW3Ttxt59bHrn739osnQ80RbOGVT
H/Vi //L3BGj ZrZr+PwTh2Vb3wl hruij Mej 2m4
E2Mth/Xj SDAhYZVWCNhJGOnPH6G6Ww==
) ; key id = 3755
86400 DNSKEY 257 3 5 (
AQP5OR9BUu0Q8i en6Wi baSsKddzZstW4TEu
JrSzezQL79DFqHe0vVuhJr+9JMqmJuQGuj Vc
XDG1gBRQboi FJ6e+G6si bl Kl kzXCLSX709Yq
Ytyv1AMyEbYWLTwRvKoj ZSZr2LyKqKGfQWd
oA8a1M6XRuChBl wxMwo5l 5Fsedl yYw==
) ; key id = 29022
86400 RRSIG DNSKEY 5 3 86400 20061008080832 (
20060908080832 3755 dnssec.potaroo.net.
EMXe20wX8CNOeAg1i exEMSI GUuApeI B/zW1z
pHhZ+l /9YFE2bmmWaj 6+j tFMMW8tvj l qdEFH
8T0i hsMaPhuOnMQngTrKTNS4Y4DKHqt05N6a
3yS1h/urFRfBDn2rA5EquVNGZM6TRI Oi weDSn
1HsWy5+Fi QcCFubsVJj CyqG/RXo= )
86400 RRSIG DNSKEY 5 3 86400 20061008080832 (
20060908080832 29022 dnssec.potaroo.net.
plmpAtyi Ql NPi ORci bl cry9eofJhvm6mkxZS
nL5Qb4x/g+DC02kXMHfCsVvNSU9ATAwRI OhY
PG85LaC7FdfWd0ud5l +AVvVPRB+8aX1scS/8
/kQ5AbJuxT3b6ezCEhu2FSuRKN3uskV5Af4N
1nBBVmFwd7vXR53Q6KcucWj Bvmg= )
*.dnssec.potaroo.net. 86400 IN A 203. 50. 0. 18
86400 RRSIG A 5 3 86400 20061008080832 (
20060908080832 3755 dnssec.potaroo.net.
UTLI JPY60i aVo8skJKbl j kF+DzOZFJi PGSLM
EmmzHVI YNel j pQNK/o5j cl dDv7S4MZ+MJg31
MLPXuStBWe8El fwU4w+eQX38dXP1fPs2Mj z2
RyG/dw2KrgvVRFQDa27UJvurxDxo0TykEwW7
yYzAdA6oVfl Ekj yTF80/CxrGVy0= )
10800 NSEC bgp.dnssec.potaroo.net. A RRSIG NSEC
10800 RRSIG NSEC 5 3 10800 20061008080832 (
20060908080832 3755 dnssec.potaroo.net.
ThBl Ngb7kHEq5t+wunmN/uTxI E5Z3nXl 29e9
eFFi dmBmMo459/oXeuc8w8kh9UOX2TQ1og8L
3GQwLNO75JrbsgMOSGzhNVD5b7Yj 7PZNPWa7
M408z7ok3Dru5X0Yf4NV5fORUsvHhBn0Br+/
6wTSdnpI /mQvGk5EmCKPkwvhzqE= )
bgp.dnssec.potaroo.net. 86400 IN A 203. 50. 0. 159
86400 RRSIG A 5 4 86400 20061008080832 (
20060908080832 3755 dnssec.potaroo.net.
Bj Qffi Lai 0xP4rKI zT90vteuVR3kR2NBYZgM
WMQxbSK6X4b84hE6HTRparY71bXXBvcXI t7
MpWX97m1A9KScR7b37h084ZE1l 6b86eaN3f9
Ad+9X1NXPw/RdrQZxby5xkyNSB0oI pM8ROJz
kKGGi +005tn703TyBWMrI CznI aA= )
10800 NSEC bgp2.dnssec.potaroo.net. A RRSIG NSEC
10800 RRSIG NSEC 5 4 10800 20061008080832 (
20060908080832 3755 dnssec.potaroo.net.
Sj j K2OpKnv514pUdOcfTMkqpgggl j vcf+1NP
fi zuFXMj OewJbdsKxkE9FaRHwrDNvQpnwdyy
adgv+TBRLZhtHr1p07aFfPyXCnsABffnPhWc

```

bpg2. dnssec. potaroo. net.	86400	IN A	Os/xb1mAhaMptf3f7Ri /CxrF5HFQF/I HHbHW
	86400	RRSIG	UUHzU2dkM8wOHzkGP/OPv5oND0o=) 203. 50. 0. 33 A 5 4 86400 20061008080832 (
	10800	NSEC	20060908080832 3755 dnssec. potaroo. net.
	10800	RRSIG	fuYkcuJ F/mi EDcfSEPPAC/5wvYg6MmEqml sg xFTwDykT0otCSdsy5R/20meDtWbYWqwl 1Wb5 7zTuBmSJprki Ti eq69j 8Ptr4y3JGEsNeGA14 1FDmpqdT29kgvWJHki ZyEJ7Hj 2zV9WuOrpu6 6hzW7pKz/xm9+Xv2ssx+u5nfrXU=)
dns0. dnssec. potaroo. net.	86400	IN A	dns0. dnssec. potaroo. net. A RRSIG NSEC
	86400	RRSIG	NSEC 5 4 10800 20061008080832 (
	10800	NSEC	20060908080832 3755 dnssec. potaroo. net.
	10800	RRSIG	S+X4oHey+hi zmSF8d73877qYGK782uJzLg0A dHn0D9RC0Sol Sri KFSd+l /q+47ckBhsaMx5B 9j MTRwor1fKZH8XKkyi NsuQj JqHi 84sh2ZeK fGmGPEZpDZR+Pk2bi QSRpJo9tH29B0fsSE/0 fGDj l mgkRhuj nMI A/7RA10i l PFO=) 203. 50. 0. 18 A 5 4 86400 20061008080832 (
dns0. dnssec. potaroo. net.	86400	IN A	20060908080832 3755 dnssec. potaroo. net.
	86400	RRSIG	LpbfDJtud6wqXVLnurpxkCuYti FakQOHFKI F qJfs/R9xGwNi zeS4f2+dR/rGnwXTDw522qdT JFI BXbBR9RG9pSEqOck/i vNSF8dPc7URbi 4e EORWkgf9fE87x6cd2CHEaOrcgHDXbCZX594R oWeutR9WohUPovs0aT1f0t2C9Gs=)
	10800	NSEC	dns1. dnssec. potaroo. net. A RRSIG NSEC
	10800	RRSIG	NSEC 5 4 10800 20061008080832 (
dns0. dnssec. potaroo. net.	86400	IN A	20060908080832 3755 dnssec. potaroo. net.
	86400	RRSIG	fGmGPEZpDZR+Pk2bi QSRpJo9tH29B0fsSE/0 fGDj l mgkRhuj nMI A/7RA10i l PFO=) 203. 50. 0. 18 A 5 4 86400 20061008080832 (
	10800	NSEC	20060908080832 3755 dnssec. potaroo. net.
	10800	RRSIG	LpbfDJtud6wqXVLnurpxkCuYti FakQOHFKI F qJfs/R9xGwNi zeS4f2+dR/rGnwXTDw522qdT JFI BXbBR9RG9pSEqOck/i vNSF8dPc7URbi 4e EORWkgf9fE87x6cd2CHEaOrcgHDXbCZX594R oWeutR9WohUPovs0aT1f0t2C9Gs=)
dns1. dnssec. potaroo. net.	86400	IN A	dns1. dnssec. potaroo. net. A RRSIG NSEC
	86400	RRSIG	NSEC 5 4 10800 20061008080832 (
	10800	NSEC	20060908080832 3755 dnssec. potaroo. net.
	10800	RRSIG	fxw5MRcKkj R6cbRbaD4u/28sOLKZbVVTj Yas 1dBcYyx0aw3l l Upvyl sj ERU+oEG+g2DUQui + 2LA6PVntaCbKwfezwGkbtZBGkbuUcfnCdEa7 dNKQv3Aki 5qGw1EAl kbahkt1FGbQLw0/Wl 4g JFmOpYfcmALtZdhgyWX60KBGI oY=) 203. 50. 0. 6 A 5 4 86400 20061008080832 (
sub. dnssec. potaroo. net.	86400	IN NS	20060908080832 3755 dnssec. potaroo. net.
	86400	IN NS	C1NTVm64mJDTDpM+aX070LWhi 92G9l 5hki W5 QbBmml TLq1x7QhMpasSPH41PRpa+teyeByFI /46QGRpVb8l P4KmpBURd1YkPwAJbBBwb20+s dXA6vfz3R/GSa62vSb2aCPfpvAAPkE3Hs66m DF3DwVONpGuSgAWph3A3H+1KbQs=)
	10800	NSEC	sub. dnssec. potaroo. net. A RRSIG NSEC
	10800	RRSIG	NSEC 5 4 10800 20061008080832 (
www. dnssec. potaroo. net.	86400	IN A	20060908080832 3755 dnssec. potaroo. net.
	86400	RRSIG	Rfj ymAnoHG3TQ909fU/l env3Gsl ZtEqR6fs7 fa/KJ4o4/OZU7+/VGz3CgUwB0LeMBab9F+Yr KuFi 83KvAt/W4EOnGxeDwgtnkTzUQJpkv7l A AStqMI rqsZc8FyGJuZPJgU8Fzvn7+Ju7qsPU Ntwi 658ZRkoUl /K7uok607HmGSE=)
	10800	NSEC	dns0. dnssec. potaroo. net.
	10800	RRSIG	dns1. dnssec. potaroo. net. www. dnssec. potaroo. net. NS RRSIG NSEC NSEC 5 4 10800 20061008080832 (
	10800	NSEC	20060908080832 3755 dnssec. potaroo. net.
	10800	RRSIG	XNgHeGnZnmdg8AwHcnsVW6DzZEtNB0n5HVpk 1m2/oFoVgFr7MBuJT1t0beN8p/2zMuLF3Wad HLmwLXOGqYBE/f+6afl A33aWTrLkuB11cmUj i Ek/4xMKAUgRAN04V6j OvVDnyESXY6g6afd3 J9yhhNvDukG3/8l q1bUyVI RSKz8=) 203. 50. 0. 6 A 5 4 86400 20061008080832 (
	10800	NSEC	20060908080832 3755 dnssec. potaroo. net.
	10800	RRSIG	gWxzDRdi VRWxMCseWPQ2oi OQl HQxMZHT+Oj + nk+tJMW3gvEVH+i P6uLGkwewywey8Ek1bLMe UwqI h6z8B35pBBn0Hl j w03x00Ly3ELHvtHUB Q/2/bDbFaFDaXNA5l Qn8l 4RGLuaExDKqOdI F tL/hq9y4rNHg7WTcNw9Q3pRfNUA=)
	10800	NSEC	dnssec. potaroo. net. A RRSIG NSEC
	10800	RRSIG	NSEC 5 4 10800 20061008080832 (
			20060908080832 3755 dnssec. potaroo. net.

```
LRLFql s+FF2DqvUPOrnl oRe60cl swCG/RL38
X1NL0shkpYj K4GcCsgsoyYcXh2vmt2va+0U
RqVgLO6brBi zmmG7raS4kK9yd0bP+91Ci kWF
HuN8GOLj Z0SeI 8Cty0eahtj y7cdqVovPkcj e
P1yj DR8cl 58wVsdvSCWI aoeCx9k= )
```

The zone file has been sorted into canonical order, NSEC records have been inserted for every label, each RRSet has been signed with the Zone signing key, and the DNSKEY RRset has been signed by both the zone signing key and the key signing key.

Now to change named.conf to reference the signed zone file. This is the new section in the named conf file for the zone server:

```
zone "dnssec.potaroo.net" {
    type master;
    file "master/dnssec.potaroo.net.signed";
};
```

I also need to add the public key signing key of the domain to my trusted key set in the trusted-keys section of named.conf of my local DNS resolver and signal a HUP to the named process to get it to re-read the named.conf file. Here's my new trusted key for the zone dnssec.potaroo.net on my local name resolver:

```
trusted-keys {
[...]
```

```
"dnssec.potaroo.net." 257 3 5 "AQPSOR9BUUnuQ08i en6Wi baSsKddzZstW4TEuJrSzezQL79DFqHeOvVuh
Jr+9JMQmJuQGUj VcXDG1gBRQboi FJ6e+G6si bl KI kzXCLSX709YqYtyv
1AMyEbYwLTwrVvKoj ZSZr2LyKqeKGFqWdoA8a1M6XRuChBI wxMwo5I 5Fs
edI yYw=";
};
```

Did it work?

```
# dig +dnssec +multiline DNSKEY dnssec.potaroo.net
; <<>> DiG 9.3.2-P1 <<>> +dnssec +multiline DNSKEY dnssec.potaroo.net
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27239
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;dnssec.potaroo.net.      IN DNSKEY

;; ANSWER SECTION:
dnssec.potaroo.net.      86400 IN DNSKEY 256 3 5 (
                          AQ08xvbN4hZ8bn926wpM8c9UqqhqcF45v73k4J/YSu+6
                          o/QsPCKwJoDYxMH3s5ZONJI gLUQscl ZZKDYVHPW3Txt5
                          9bHrn739osnQ80Rb0GVTH/Vi //L3BGj ZrZr+PwH2Vb3
                          wl hruj Mej 2m4E2Mth/Xj SDAhYZVWCNhJG0nPH6G6Ww=
                          ) ; key id = 3755
dnssec.potaroo.net.      86400 IN DNSKEY 257 3 5 (
                          AQPSOR9BUUnuQ08i en6Wi baSsKddzZstW4TEuJrSzezQL
                          79DFqHeOvVuhJr+9JMQmJuQGUj VcXDG1gBRQboi FJ6e+
                          G6si bl KI kzXCLSX709YqYtyv1AMyEbYwLTwrVvKoj ZSZr
                          2LyKqeKGFqWdoA8a1M6XRuChBI wxMwo5I 5FsedI yYw=
                          ) ; key id = 29022
dnssec.potaroo.net.      86400 IN RRSIG DNSKEY 5 3 86400 20061008080832 (
                          20060908080832 3755 dnssec.potaroo.net.
                          EMXe20wX8CNOeAg1i exEMSI GUuApel B/zW1zpHhZ+I /9
                          YFE2bmmWaj 6+j tFMmW8tvj l qdEFH8TOi hsMaPhuOnMQn
                          qTrKtNS4Y4DkHqt05N6a3yS1h/uFRfBDn2rA5EquVNGZ
                          M6TRI Oi weDSn1HsWY5+Fi QcCFubsVJj CyqG/RXo=
                          )
dnssec.potaroo.net.      86400 IN RRSIG DNSKEY 5 3 86400 20061008080832 (
                          20060908080832 29022 dnssec.potaroo.net.
                          plmpAtyi QI NPi ORci bl cry9eofJhvm6mkxZSnL5Qb4x/
                          g+DC02kXmHfCsVvNSU9ATAwRI OhYPG85LaC7FdfWd0ud
                          5I +AVvVPRB+8aX1scS/8/kQ5AbJuxT3b6ezCEhu2FSuR
                          KN3uskV5AF4N1nBBVmfWd7vXR53Q6KCucWj Bvmg=
                          )

;; Query time: 412 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Sep 12 09:44:37 2006
```

```
; MSG SIZE rcvd: 695
```

The “ad” bit has been set in the flags of the response,, so it looks good.

The name resolver’s debug log also confirms this:

```
validating @0x8247000: dnssec.potaroo.net DNSKEY: starting
validating @0x8247000: dnssec.potaroo.net DNSKEY: attempting positive response validation
validating @0x8247000: dnssec.potaroo.net DNSKEY: verify rdataset: success
validating @0x8247000: dnssec.potaroo.net DNSKEY: signed by trusted key; marking as secure
validator @0x8247000: dns_validator_destroy
```

Lets check for a name that exists within this zone:

```
# dig +dnssec +multiline A www.dnssec.potaroo.net
; <<>> DiG 9.3.2-P1 <<>> +dnssec +multiline A www.dnssec.potaroo.net
; global options: printcmd
; Got answer:
; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 64058
; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; QUESTION SECTION:
;www.dnssec.potaroo.net.      IN A
;
; ANSWER SECTION:
www.dnssec.potaroo.net. 86317 IN A 203.50.0.6
www.dnssec.potaroo.net. 86317 IN RRSIG A 5 4 86400 20061008080832 (
    20060908080832 3755 dnssec.potaroo.net.
    gWXzDRdi VRWxMCseWPQ2oi 0QI HQxMZHT+Oj +nk+tJMW3
    gvEVH+i P6uLGkewywey8Ek1bLMeUwql h6z8B35pBBn0
    hl jw03x00Ly3ELHvtHUBQ/2/bDbFaFDaXNA5I Qn8I 4RG
    LuaExDKq0di FtL/hq9y4rNHg7WtcNw9Q3pRFNUA= )
;
; Query time: 75 msec
; SERVER: 127.0.0.1#53(127.0.0.1)
; WHEN: Tue Sep 12 13:56:27 2006
; MSG SIZE rcvd: 245
```

And check for a name that exists through the wildcard entry

```
# dig +dnssec +multiline A wilddcard.dnssec.potaroo.net
; <<>> DiG 9.3.2-P1 <<>> +dnssec +multiline A wilddcard.dnssec.potaroo.net
; global options: printcmd
; Got answer:
; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 55385
; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 5, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; QUESTION SECTION:
;wilddcard.dnssec.potaroo.net. IN A
;
; ANSWER SECTION:
wilddcard.dnssec.potaroo.net. 10800 IN A 203.50.0.18
wilddcard.dnssec.potaroo.net. 10800 IN RRSIG A 5 3 86400 20061008080832 (
    20060908080832 3755 dnssec.potaroo.net.
    UTLI JPY60i aVo8skJKbl j kF+DzOZFJi PGSLMEmmzHVI Y
    Nel j pQNK/o5j cl dDv7S4MZ+MJg31MLPXuStBWe8EI fwU
    4w+eQX38dXP1fPs2Mj z2RyG/dw2krqvVRfQDa27UJVur
    xDxoQTykEwW7yYzAdA6oVfi Ekj yTF80/CxrGVy0= )
;
; AUTHORITY SECTION:
sub.dnssec.potaroo.net. 10800 IN NSEC www.dnssec.potaroo.net. NS RRSIG NSEC
sub.dnssec.potaroo.net. 10800 IN RRSIG NSEC 5 4 10800 20061008080832 (
    20060908080832 3755 dnssec.potaroo.net.
    XNgHeGnZnmdg8AwHcnsW6DzZEtnBOn5HVpk1m2/oFoV
    gFr7MBuJT1t0beN8p/2zMuLF3WadHLmwLX0GqYBE/f+6
    afI A33aWTrLkuB11cmUj i Ek/4xMKAUgRANO4V6j OvVDn
    yESXY6g6afd3J9yhhNvDukG3/8I q1bUyVI RSKz8= )
;
; AUTHORITY SECTION:
dnssec.potaroo.net. 86294 IN NS dns0.potaroo.net.
dnssec.potaroo.net. 86294 IN NS dns1.potaroo.net.
dnssec.potaroo.net. 86294 IN RRSIG NS 5 3 86400 20061008080832 (
    20060908080832 3755 dnssec.potaroo.net.
    p2kKLK4gzl m8nkr4I pXyz4Fi rWWXti yXc5X/Ns2NYC3C
    NYDNI RFHzEI 14RZ008R9z4aoQl f0j Xi di JZ2Bgxzmykv
```

```

JUA7AwGi rVtr+6wDJrdi f9tm7UdYN2powrP9o2l qODK
hwYk8i 4Dyj dd9kwt7/x44ZECzEj 7w30GfW4uvy8= )
;; Query time: 81 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Sep 12 13:56:50 2006
;; MSG SIZE rcvd: 693

```

Again this looks good. It appears that as long as the resolver is willing to trust the zone key then I've managed to correctly sign a zone and publish the outcomes.

Now I'd like to extend this by signing the delegated zone "sub.dnssec.potaroo.net". Because this is an immediate descendant of a signed zone then if I set this up correctly then the same trusted key for the parent zone should be able to be used to validate the child zone. The same initial process of zone signing is followed:

1. Generate the zone key and the key-signing keys

```

# dnssec-keygen -r/dev/random -a RSASHA1 -b 1024 -n ZONE sub.dnssec.potaroo.net
Ksub.dnssec.potaroo.net.+005+55625
# dnssec-keygen -r/dev/random -a RSASHA1 -f KSK -b 1024 -n ZONE sub.dnssec.potaroo.net
Ksub.dnssec.potaroo.net.+005+49350

```

2. Add the keys to the zone file, and check the result

```

# xemacs sub.dnssec.potaroo.net &
[add include lines to the zone file for the host and KSK key files]
# named-checkzone -D sub.dnssec.potaroo.net sub.dnssec.potaroo.net
zone sub.dnssec.potaroo.net/IN: loaded serial 2006091201
sub.dnssec.potaroo.net. 86400 IN SOA dns0.dnssec.potaroo.net.
gih.potaroo.net. 2006091201 10800 15 604800 10800
sub.dnssec.potaroo.net. 86400 IN NS dns0.dnssec.potaroo.net.
sub.dnssec.potaroo.net. 86400 IN NS dns1.dnssec.potaroo.net.
sub.dnssec.potaroo.net. 86400 IN DNSKEY 256 3 5
AQ0oTGeVj i 8FFqWNqezyx/C5omVEg98JPzEFSLMbGL32b+0v9GXxtPM
SGxc+i i JLe2nGuMwgWqLFEUxUeK9P9ZNDR1CurtDKh1KPGQJj f30Jq9B
CffeVaGy902K0vLFH9BqVo9L/J5i TcyR3zE0kMoE2rgyYtI UsmUnM1+j usFDrw== ; key id = 55625
sub.dnssec.potaroo.net. 86400 IN DNSKEY 257 3 5
AQ04JdCpTvv9I Z87bJHA8ZY0GZZNr5ul 9Nc7/Sh1toKI wQHh3DWLp3q/
yAto8m80wtTb7nZr+Jvem9eI fWRY6Rqgc/bETi AY0I U9f+qdE87KEi pw
RnRI MSCj mcJBhEYbJhWmONBhT8I dwqtWVFuI AsmI Xgl 5si G5YAKbl +X XHA0SQ== ; key id = 49350
another.sub.dnssec.potaroo.net. 86400 IN A 203.50.0.18
example.sub.dnssec.potaroo.net. 86400 IN A 203.50.0.6
www.sub.dnssec.potaroo.net. 86400 IN A 203.50.0.6
OK

```

3. Bump the SOA value and Sign the child zone

I'm not sure why I put the "-d ." command option to the dnssec-signzone command here. I thought it has something to do with generating the DS and keyset files that I need to pass to the parent zone, but the man page for this command seems to suggest otherwise. Indeed the man page for this command is sufficiently unclear on the entire topic of dsset and keyset files as to be entirely useless to me. I had the distinct impression that I was walking in the dark at this point in time.

```

# /usr/local/sbin/dnssec-signzone -r /dev/random -d . -o sub.dnssec.potaroo.net -k
Ksub.dnssec.potaroo.net.+005+49350 sub.dnssec.potaroo.net Ksub.dnssec.potaroo.net.+005+55625.key
sub.dnssec.potaroo.net.signed
# cat sub.dnssec.potaroo.net.signed
; File written on Tue Sep 12 14:53:06 2006
; dnssec_signzone version 9.3.2
sub.dnssec.potaroo.net. 86400 IN SOA dns0.dnssec.potaroo.net. gih.potaroo.net. (
2006091201 ; serial
10800 ; refresh (3 hours)
15 ; retry (15 seconds)
604800 ; expire (1 week)
10800 ; minimum (3 hours)
)
86400 RRSIG SOA 5 4 86400 20061012035306 (
20060912035306 55625 sub.dnssec.potaroo.net.
0h6j xu5LXSfD40pWTMOi l U2rj N74GJNkvdJ1
GOI u+j ywnNDKAwAJ4oRAFmp64/P+KZRWHUnM
qMpbkWc+12yFANXPGW0xhRFG0dOXUV5i YZSh

```

```

rTzS134CzksmDDQoTI j Qf68vD6Oowm5vBGTL
3PmJDDMI Vyk9MPJ4xZHGHmQZUs= )
86400 NS dns0. dnssec. potaroo. net.
86400 NS dns1. dnssec. potaroo. net.
86400 RRSIG NS 5 4 86400 20061012035306 (
20060912035306 55625 sub. dnssec. potaroo. net.
BOI EobJvCHagl QR263P06FnYnHyGP7npbKzI
Off6uHn4OZYXHws8USEebC+I 6m/dCbBxPfi R
YOI P4p9G/3cdUG8TAWKoNLI hxZ7JnJEXmm01
M32PhZcBj EwtSMxxOp7LZ5+aBI NvS6JmWnY1
JXkPvSENLMpDTtNwz/BQJeE10hl= )
10800 NSEC another. sub. dnssec. potaroo. net. NS SOA RRSIG NSEC DNSKEY
10800 RRSIG NSEC 5 4 10800 20061012035306 (
20060912035306 55625 sub. dnssec. potaroo. net.
JqKfUvi 5CTI 1i SKXZxep1HFLNc4BUq4hj 14Z
ggD37Jo5VX8FWj 4pi RANQSWI 11GOp8G2FI +b
YTqX4mn0kM1hl YsyqH208wnJcpeGgaYfho15
7Hi wLOCNHeL9CSSpNbFR5J4A0Rm46b16M0Q/
tmcZHdzNoXKENUvi x6rUN8kYCU= )
86400 DNSKEY 256 3 5 (
AQOoTGeVj i 8FFqWNqezyx/C5omVEg98JPzE
FSLMbGL32b+Ov9GXxtPMSGxc+i i JLe2nGuMw
gWqLFEUxUeK9P9ZNDRI CurtDKh1KPGQJ f30
Jq9BCfFeVaGy902K0vLFH9BqVo9L/J5i TcyR
3zE0kMoE2rgyYtI UsmUnM1+j usFDrw==
) ; key id = 55625
86400 DNSKEY 257 3 5 (
AQO4JdCPtVV9I Z87bJHA8ZYOGZZNr5ul 9Nc7
/Sh1toKI wQHh3DWLp3q/yAto8m80wtTb7nZr
+Jvem9ei fWRy6Rqgc/bETi AY0I U9f+qdE87K
Ei pwRnRI MSCj mcJBhEYbJhWmONBhT8I dwqt
wVfUl AsmI Xgl 5si G5YAKbl +XXHAOSQ==
) ; key id = 49350
86400 RRSIG DNSKEY 5 4 86400 20061012035306 (
20060912035306 49350 sub. dnssec. potaroo. net.
D5sv0I t6E8v/I akXQyVdXqQXMMZvY1Yqi HA
2EeEt08TNUgHs1vSme3rW+88Yj fxwXk2GF1
r01RaabtCi 90cNk60cJ98FSyJ I VkvPrW2xB
k2S4SBWz0uwBq2VbFhQc1AkL0kZR97ef4GkC
hchvuf656sPYun3QZd285w0s0J8= )
86400 RRSIG DNSKEY 5 4 86400 20061012035306 (
20060912035306 55625 sub. dnssec. potaroo. net.
UJyhU+eTh/cSJ/4dsdvBj bT0c4MED3Z+mU1b
kmhuChTkbOLEWeVhDXFSBk7S3ugemZJI FQbL
I wOF4ZfemZq+V/PLnD3+u3wMywn5AKJ2cj oY
W2CWk25qyaZef23YtMI 1Guq3qsOd9KCb9NCW
wra3Yi OBx8j Tj r1VFMnpPzmxN58= )
another. sub. dnssec. potaroo. net. 86400 IN A 203. 50. 0. 18
86400 RRSIG A 5 5 86400 20061012035306 (
20060912035306 55625 sub. dnssec. potaroo. net.
ZOGW6QvuU03I S4cmnl j I NWyXUQ5/LcnJw1bU
hj e1MM+o1yuQMMUxpWPJSfe7R7mJ6u8k3LTU
P+wBi 7hF4S07xD08VLD5XmN6capRFUMkasK0
XOHg+2+mUSv00EY+Ovl 4/7R6vZM3Xj seoDRK
I /bgeoawKEf5dl dB+XI m425tu9l= )
10800 NSEC exampl e. sub. dnssec. potaroo. net. A RRSIG NSEC
10800 RRSIG NSEC 5 5 10800 20061012035306 (
20060912035306 55625 sub. dnssec. potaroo. net.
06m+Xoo0VuFI HVcLqd3Vqy7WeDos2CV2xShr
YPCmBoUKF774BnYrKMPe+H05p65ki yTh3w9L
z1I ayzh6SN+HVrrf6vMYgl VI xyqScnpMdZMT
vS5VP2bHerr6WudXUdCb6XZSACB2OCT5K2MW
A3/sYQoFQyGMamp5aETrReAJwxQ= )
exampl e. sub. dnssec. potaroo. net. 86400 IN A 203. 50. 0. 6
86400 RRSIG A 5 5 86400 20061012035306 (
20060912035306 55625 sub. dnssec. potaroo. net.
i/coth8MpAj I i Ei Oj wMsvNOT81aQOnSkuxW0
boLRgtDpF3Si 6oGMM2AhHLgnJDepvSGBqqEw
Yql agfhebF/j oC7F8K3GxUI /duj g1eosTY7w
eL3adFRKBMGqkkdw70rY417f8VvH2hg2wG+P
KMONLA3DmxQ16j qd3f9cc9Tz2Bg= )
10800 NSEC www. sub. dnssec. potaroo. net. A RRSIG NSEC
10800 RRSIG NSEC 5 5 10800 20061012035306 (
20060912035306 55625 sub. dnssec. potaroo. net.
OHxc6vMcbm4I ts2BMI bMs0QGLu6KdVroqCSA
F9i aZsOyvbnzRBDr6tyC3ANJh8Ld48CA9uPh
8HWI 13i U5ui DcoELyFUBZl 8vPakK0enDB52b
XBKbWCub51nvj s4PMNS4MnsJTC82ePeHh0JK
5cKW3uodx2TmaAaOpf8V2kugaMA= )
www. sub. dnssec. potaroo. net. 86400 IN A
86400 RRSIG A 5 5 86400 20061012035306 (
20060912035306 55625 sub. dnssec. potaroo. net.
Pyej oxvnwdMdmOBAaMWxg649TQLTFTL9JBwi
hXzxB1F3I 5YC1gHrNrZyUJbyaZOFB3JM9nY+

```

```

zOI +TF25hm3tMXI EPj N6eAYk7CHh5T8XHJmv
rgl 2Ermxs7+DFbLOB1CP0sSRS8m8/L7JI vCP
8l 0ZqKTbECUdqY8bTFwznhD/wvg= )
10800 NSEC sub.dnssec.potaroo.net. A RRSIG NSEC
10800 RRSIG NSEC 5 5 10800 20061012035306 (
20060912035306 55625 sub.dnssec.potaroo.net.
k7uDto4S6XVqafv01BNVnnU3EG/Fsz42HoYf
atqbL5a71Syu67ATQE5N0i mWB7LJI Bn74Go/
dQGc526tq2xCCs3L/zqk3Vgkyl oEE8ti GFaE
80XhATFnnpAj GGi ApLk9d+uWBaZ7cg3cR+V4
kKv51mj g8h+1bS5aydKgmq7QZdE= )

```

4. Serve the signed child zone.

```

# cd /etc/namedb
# xemacs named.conf
[reference sub.dnssec.potaroo.net.signed as the zone file for this domain]
# ps axuw | grep named | grep -v grep
root 68795 0.0 0.1 4328 3332 ?? Ss Sat10AM 0:00.77 /usr/local/sbin/named -c
/etc/namedb/named.conf -d 10
# kill -s HUP 68795

```

5. Check that the new signed zone has been loaded by the server

```

# dig SOA sub.dnssec.potaroo.net @127.0.0.1
; <<>> DiG 9.3.2 <<>> SOA sub.dnssec.potaroo.net @127.0.0.1
; (1 server found)
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19161
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
sub.dnssec.potaroo.net.          IN      SOA

;; ANSWER SECTION:
sub.dnssec.potaroo.net. 86400 IN      SOA      dns0.dnssec.potaroo.net.  ghi.potaroo.net.  2006091201
10800 15 604800 10800

;; AUTHORITY SECTION:
sub.dnssec.potaroo.net. 86400 IN      NS       dns0.dnssec.potaroo.net.
sub.dnssec.potaroo.net. 86400 IN      NS       dns1.dnssec.potaroo.net.

;; ADDITIONAL SECTION:
dns0.dnssec.potaroo.net. 86400 IN      A        203.50.0.18
dns1.dnssec.potaroo.net. 86400 IN      A        203.50.0.6

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Sep 12 14:59:09 2006
;; MSG SIZE rcvd: 150

```

6. Now I need to generate a DS Resource Record for the child zone that can be placed into the parent zone

A side effect of the dnssec-signzone call in step 3 was the generation of the keyset and dsset files

```

# ls -al *set-sub*
-rw-r--r-- 1 root wheel 81 Sep 13 09:20 dsset-sub.dnssec.potaroo.net.
-rw-r--r-- 1 root wheel 289 Sep 13 09:20 keyset-sub.dnssec.potaroo.net.
# cat keyset-sub.dnssec.potaroo.net.
$ORIGIN .
sub.dnssec.potaroo.net 10800 IN DNSKEY 257 3 5 (
AQ04JdCPtVV9I Z87bJHA8ZYOGZZNr5ul 9Nc7
/Sh1toKI wQHh3DWLp3q/yAto8m80wtTb7nZr
+Jvem9eI fWRy6Rqgc/bETI AY0I U9f+qdE87K
Ei pwRnRI MSCj mcJBhEYbJhWmONBHhT8I dwqt
wVFul AsmI Xgl 5si G5YAKbl +XXHAOSQ==
) ; key id = 49350
# cat dsset-sub.dnssec.potaroo.net.
sub.dnssec.potaroo.net. IN DS 49350 5 1 075734C803444B198C0681A8FD9A9E4378E4F4B2

```

I'm not sure which of these should be passed to the parent zone admin, so I'll hurl both files across the fence and leave it to the parent to figure out what to do!

7. Now to get the parent zone key to sign the dsset record for the child domain

I've added the DS RR to the parent zone file. Something feels wrong here, and I'm not sure why I'm doing this. The RIPE tutorial document gets pretty unclear at this stage, and the man page for the `dnssec-signzone` command appears to suggest that I put the keyset file in a special directory and get the `dnssec-signzone` command to generate the ds-set files via the `-d` command option referencing the location of the key-set files for the child zone. This approach did not work for me, and after a series of experiments I found that if I took the child zone's ds-set file contents and inserted this DS record directly into the parent zone file for the subdomain, then things appear to work. I couldn't find any documentation that confirmed that this was the 'correct' way to get the child's key information into the parent zone, so I probably did something wrong here. Did I mention already that I found the DNSSEC `dnssec-signzone` utility documentation terse and less than helpful?

```
# xemacs dnssec.potaroo.net
[add dsset-sub.dnssec.potaroo.net]
# cat dnssec.potaroo.net
$TTL 86400

$ORIGIN          dnssec.potaroo.net.

@                IN          SOA          dns0.potaroo.net.  ghi.potaroo.net. (
                2006091302 ; Serial
                3h       ; Refresh
                15      ; Retry
                1w       ; Expire
                3h      ) ; Minimum

;
;
; Name servers
;
                IN          NS          dns0.potaroo.net.
                IN          NS          dns1.potaroo.net.
;
; subdomains
;
sub              IN          IN          NS          dns0.dnssec.potaroo.net.
                IN          NS          dns1.dnssec.potaroo.net.
                IN          DS          49350 5 1 075734C803444B198C0681A8FD9A9E4378E4F4B2
;
; zone A records
;
www              IN          A          203.50.0.6
bgp              IN          A          203.50.0.159
bgp2             IN          A          203.50.0.33
dns0             IN          A          203.50.0.18
dns1             IN          A          203.50.0.6
;
; wildcard
;
*                IN          A          203.50.0.18

$include Kdnssec.potaroo.net.+005+03755.key ; zone signing key
$include Kdnssec.potaroo.net.+005+29022.key ; key signing key
```

8. Now re-sign this zone file

Not forgetting to bump the SOA value beforehand, of course.

```
# /usr/local/sbin/dnssec-signzone -r /dev/random -o dnssec.potaroo.net. -k
Kdnssec.potaroo.net.+005+29022 dnssec.potaroo.net Kdnssec.potaroo.net.+005+03755.key
dnssec.potaroo.net.signed
# cat dnssec.potaroo.net.signed
; File written on Wed Sep 13 09:34:58 2006
; dnssec_signzone version 9.3.2
dnssec.potaroo.net.      86400   IN  SOA  dns0.potaroo.net.  ghi.potaroo.net. (
                        2006091302 ; serial
                        10800      ; refresh (3 hours)
                        15         ; retry (15 seconds)
                        604800     ; expire (1 week)
                        10800     ; minimum (3 hours)
                        )
                        86400   RRSIG  SOA 5 3 86400 20061012223458 (
                        20060912223458 3755 dnssec.potaroo.net.
                        tpRo4CwylVHGBctDcZyhzc+Gqqca2avl bE9j
```

			Pfn3JwSti /eyHxEYB7ELOJ18bDXa7xfBoq59 00Ti zV4K1p1se6qRDxUEp6F032R6DMDkOxU3 YdMMkScy0R2+nZ/Vasi Pvl RLdhFrBhN53HmR L+VRuT+VGYDYFqq2AJco4NPV8Go=)
	86400	NS	dns0. potaroo. net.
	86400	NS	dns1. potaroo. net.
	86400	RRSIG	NS 5 3 86400 20061012223458 (20060912223458 3755 dnssec. potaroo. net. qCqS29j XtzVr9GYPrLsz0wBf0vsSWAcMwSpV NyrL1LFf6VgF4/trY0XdQYssR0n550RXqKYX mtI yij G/I 1RU8Hi Ofkji l Prx0dw6+DM1bcMS 6xI PN8GPLx6a0h4k6FE83xg5j KRbei 7xzyVy hM00rUxkDwi MhXSe1CI 07bcsUWo=)
	10800	NSEC	*. dnssec. potaroo. net. NS SOA RRSIG NSEC DNSKEY
	10800	RRSIG	NSEC 5 3 10800 20061012223458 (20060912223458 3755 dnssec. potaroo. net. XFI QUy+kU2fg6h1dKu1CCl v+B0Uj udRzMSH +VwVJvObsl i 4rdWarfpej gpoZ5YfQhrzFWb AB+0EmDrhAnkRj stB2Xzj Mt2UDHu9AgXMJaB XONFDtBvNWBquZ5VHdZd4i PH0aCcA577Gul 4 AdoFB9Ec9NZ50oE6po/h0t3NGww=)
	86400	DNSKEY	256 3 5 (AQ08xvbN4hZ8bn926wpM8c9Uqqhqcf45v73k 4J/YSu+6o/QsPCKwJoDYxMH3s5ZONJI gLUQs cI ZZKDYVHPW3Txt59bHrn739osnQ80Rb0GVT H/Vi //L3BGj ZrZr+PwH2Vb3wl hruij Mej 2m4 E2Mth/Xj SDAhYZVWCNhJG0nPH6G6Ww==) ; key id = 3755
	86400	DNSKEY	257 3 5 (AQPSOR9BUuQQ8i en6Wi baSsKddzZstW4TEu JrSzezQL79DFqHeOvVuhJr+9JMQmJuQUJ Vc XDG1gBRQboi FJ6e+G6si bl KI kzXCLSX709Yq Ytyv1AMyEbYWLTwRvKoj ZSZr2LyKqeKGFqWd oA8a1M6XRuChBl wxMwo5I 5FsedI yYw==) ; key id = 29022
	86400	RRSIG	DNSKEY 5 3 86400 20061012223458 (20060912223458 3755 dnssec. potaroo. net. AEm4Xej Ij 2Pwl ZZZUSrbaq051YOT9Q1UbWI H 835QmhWsdQl 7NW2rSp6CVnsTGpmLK4Xl A64b d72h6VGj I Fl zsuZqTBfu0/GcxY5FV+fSs0vQ Ei woDfK41CqVi QD9ogzH5P6uXyHbvk5FHDgN Qpfsh1UWe0h7oU6VcmSx/RvPSNE=)
	86400	RRSIG	DNSKEY 5 3 86400 20061012223458 (20060912223458 29022 dnssec. potaroo. net. xDH4b6m8NZcPgHRUZf2L12T8ZH0ymAso5nVB h42ZmaF5eK7Wqeq5BBYel EF31FwqF3zSxuj C BsFl tUGVxYyOeAru44pCMw8WR1Fk8o/D7B5 yp9vZdpl 6l Dl dK+i M06h0ce3zQF5dQ81T/XS hwcZQsQPgcHUN/ys3rBcfrkl uNY=)
*. dnssec. potaroo. net.	86400	IN A	203. 50. 0. 18
	86400	RRSIG	A 5 3 86400 20061012223458 (20060912223458 3755 dnssec. potaroo. net. rxx0XTi o72b5HY5ad7NI 6/JwxYl eP6P/sj oK PpTI A28uNA2E6RUryuj GRFO/P2EHap4r+eqI Pk0cl ba0tXs83/xTj nGEzsj WQNqz0e8gl KUP c/J6+/sTAM1j a7zLDr3fRqDunH2WYECj PWY/ 6Av20w6erl n5vWbsl yS+/2uxcvw=)
	10800	NSEC	bgp. dnssec. potaroo. net. A RRSIG NSEC
	10800	RRSIG	NSEC 5 3 10800 20061012223458 (20060912223458 3755 dnssec. potaroo. net. R5vkZvePoE/l St3/PUeZYMPDnseGxcI dxahu 5QyBEd1yaSzRjLNHeA1VMxgvchcH7L1tR8wY RTLnR173l cj JfHtTgQUy89Jvj dQLcVZZ+eRg 26t2fbvpzz4/kYDX67u0j SScFXmCyl 0tqgTS Rs+fgi OhA7Rl m9C1ZNzE2TqUi j 0=)
bgp. dnssec. potaroo. net.	86400	IN A	203. 50. 0. 159
	86400	RRSIG	A 5 4 86400 20061012223458 (20060912223458 3755 dnssec. potaroo. net. pl snUAtTuRI LC4A3TwdSx9TeGG2qutZLOhay Wc070F9AUZOHePi y9uhG08SBh0eGyJj 9dtsp i LhfPgB1WAax9Wj A4emAl wfbxoCj 6DhGvi QB x9krpVQAZmj aDGyQUhC/yl 1stU1mEmezWktw 23+BLSAI a6RyAc+5p0/i i Qx0BJO=)
	10800	NSEC	bgp2. dnssec. potaroo. net. A RRSIG NSEC
	10800	RRSIG	NSEC 5 4 10800 20061012223458 (20060912223458 3755 dnssec. potaroo. net. axWY/l mL+j T7d5Px8QqNzMQMxh+t/xnmoN05 n+oaAuHGfHBxbJhCxNGSb9eWfYl nQM0gg+Ta /a85LZp44c602xo46EI BuuYYt/V5kud/p4UN qskGuvDwVPT4aSkkQREDOULB0bR4Xgl P8Ttj 4mbFPNh+L6/Ggt/NpMpOK8Vm/n0=)
bgp2. dnssec. potaroo. net.	86400	IN A	203. 50. 0. 33
	86400	RRSIG	A 5 4 86400 20061012223458 (20060912223458 3755 dnssec. potaroo. net.

				LiWq7aC32VT+wI TxU9CMmOg5Rj gFDRersSYg 7DryptyYs7SHCl +Kn0I ksl zpal i vWC/HXTs1 Fi yR2bAKJz3qTs+VRaCp8o3Zcf65W1EZ+Mni WxEzWI l uu0e3WY3ofF4KY6Aha+Yu4trGkosn 95GGKxAxYzFKXni fcGLLzy2W6k=) dns0.dnssec.potaroo.net. A RRSIG NSEC NSEC 5 4 10800 20061012223458 (
	10800	NSEC		20060912223458 3755 dnssec.potaroo.net. Ywyqvugvmxu1hCD9l dzLooUR5Q9YFbCl rsQ3 96BsPJoQSu1xztu8ptNvvxcK1Nj VI e+cxJg/ /BeWA/4B6sdRcFwrDI 1LWWHRZDofMFSCOWW j kTqCVl rXzvD89c0cgTEtNqYxap/J 4zDTnmv j PB2bXT3mnci EdNaRSEpuOHU7L8=) 203.50.0.18
	10800	RRSIG		A 5 4 86400 20061012223458 (
dns0.dnssec.potaroo.net.	86400	IN A		20060912223458 3755 dnssec.potaroo.net. CudPXRmsA1/bgGC3XrYkj qpATG5bYxcLSc3q LkWGU27wl W49UKL7J8AdC6bENsrymo0FeyF QzBHxnvc+6CoedsVNUQ2wEbj SshMvl 3JHV4 Ou09oepvBh+QKkh/4VlbsvDa+y7wq28j nUvv l P33xU5g1TWl UYx03sPkuo/J crQ=)
	86400	RRSIG		20060912223458 3755 dnssec.potaroo.net. s8bl /U+CvSbfmxDd1X+MbH3eNtWtVSl QLWG Y3ZA5a2zMWml 1NTaCJ7rhUzKs4sXQuVHpBVI cTsueQK5TUwN7Y55Woi OFqCi RgeFd3Z81Rdm pHOt+t+3U1BCY9Qmpx80+RAvrVQSZ9H0muHY cA0ss5b1Qn4Cj HI SeJ1vWGqQBN4=) 203.50.0.6
dns1.dnssec.potaroo.net.	86400	IN A		A 5 4 86400 20061012223458 (
	86400	RRSIG		20060912223458 3755 dnssec.potaroo.net. ZnBFNuYbuegnNjHZckGvoHM8d0Val /RpXMI r V2B3l j 8Fi wREEmkVKNXqRj b5uVtKRPEVL/cl YZDgtI TpoE9RNh+kuoqfDa8W5g5Uj P/TpmLC afz1QUcPI etC00rsqJU0+KQXnbxZg0l rQCMQ UuPgFeBST5qweHGZK3GLfuK9d2s=)
	10800	NSEC		sub.dnssec.potaroo.net. A RRSIG NSEC NSEC 5 4 10800 20061012223458 (
	10800	RRSIG		20060912223458 3755 dnssec.potaroo.net. N5QJ2KguZSg0l Zf8BCJYenZbl nj 9ru0DD5yq UUh1gv1DxC+N2d1zKV6Q4QMGfQdBZYub5avi F28cTKVJR0YHAW3nL70aogdYnSoEpmI wL4B8 v4oHj eWofNgyLuOX9YRTUEVUHwS4l l nZvgDZ 5Eoo1h4vNDKcDvfj RShubz3KQxc=)
sub.dnssec.potaroo.net.	86400	IN NS		dns0.dnssec.potaroo.net.
	86400	IN NS		dns1.dnssec.potaroo.net.
	86400	DS		49350 5 1 (
				075734C803444B198C0681A8FD9A9E4378E4 F4B2)
	86400	RRSIG		DS 5 4 86400 20061012223458 (
				20060912223458 3755 dnssec.potaroo.net. Twi T6l fYScI aaxbJ5+/10JThRWXOz+Gyj j 7K Y4dJl oBH7oVyl PH3gUedxhQBubuxhj 1vZi Qz 55l Jxf1/m6ceS6d7AVTD8aM/P+Ogeu89v5qN aqJ4s/JB1EeHmpNqeuP0wZ0qX1AAI rnn5l Fc 9aYVGvbri 7908eUVEH4tLexY3xU=)
	10800	NSEC		www.dnssec.potaroo.net. NS DS RRSIG NSEC NSEC 5 4 10800 20061012223458 (
	10800	RRSIG		20060912223458 3755 dnssec.potaroo.net. KCRDoDQn89346cB/O8W1bx9zYj S9p2aaUB3W BLc0cop6sKP+pHvtDI t+FceGckg8l DEHXFi W swdcFGK6s07l l 9wmWewpw0hi UMM4oGhTxw4A OVU3Zl pOnkrvl j OFOLa9c6z+vnaLWH2bpx+X GaAEvl AaHq+i R9mqraC78DBCx6c=) 203.50.0.6
www.dnssec.potaroo.net.	86400	IN A		A 5 4 86400 20061012223458 (
	86400	RRSIG		20060912223458 3755 dnssec.potaroo.net. QA0QtsaGekXCg6Kb4PxmF8WYU9EOKVxAm+tT Lsy0Rnmj Sj okvAEHZeI dykKk7M5+TtdKVKHI qR20MaKXASRBPMROS/kRsq0i kZTA6zPj 70EU eEZ0ApNDx4TbYzZ7sgE9R2KsI wN2DJ9YK4FW 6abrFJ2Z0x4tFxrZqsbTEaDWL34=)
	10800	NSEC		dnssec.potaroo.net. A RRSIG NSEC NSEC 5 4 10800 20061012223458 (
	10800	RRSIG		20060912223458 3755 dnssec.potaroo.net. XhDJAbsgaKZrCGxl u0xBeXSZMDtgpL5HA08d nF8398zWxgasHJxl l R4ydfateJi PNUWkaweV VOX2b09Fu+pUpi hbDK1XReEQpGOExLxm/YGJ MuYTTfvWfd748CVZso5cEgahFYU++n5nMvBj EgwTti ZRoJSHDysj YLTxSwNnOKs=)

Note in that now-voluminous zone file that the parent has now signed the child domain's DS record.

9. Now signal the server's named process to re-read the config file

This will reload the zone information for the zone dnssec.potaroo.net in order to serve the updated zone file.

Phew! I think we are done! Over on the local DNS resolver, which is configured with the trust key of the parent zone (dnssec.potaroo.net) but not the child zone (sub.dnssec.potaroo.net), then the above steps should allow the resolver to validate answers from the child zone.

Lets check. The parent zone responses can be validated:

```
# dig +dnssec +multiline SOA dnssec.potaroo.net
; <<>> DiG 9.3.2-P1 <<>> +dnssec +multiline SOA dnssec.potaroo.net
; global options: printcmd
; Got answer:
; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 47873
; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; QUESTION SECTION:
; dnssec.potaroo.net.      IN SOA
;
; ANSWER SECTION:
dnssec.potaroo.net.      86400 IN SOA dns0.potaroo.net.  ghi.potaroo.net. (
                        2006091302 ; serial
                        10800   ; refresh (3 hours)
                        15      ; retry (15 seconds)
                        604800  ; expire (1 week)
                        10800   ; minimum (3 hours)
                        )
dnssec.potaroo.net.      86400 IN RRSIG SOA 5 3 86400 20061012223458 (
                        20060912223458 3755 dnssec.potaroo.net.
                        tpRo4Cwyl VHGBctDcZyhzc+Gqqa2avl bE9j Pfn3JwSt
                        i /eyHxEYB7ELOJ18bDXa7xfBoq5900Ti zV4K1p1se6qR
                        DxUEp6FQ32R6DMDk0xU3YdMMkScyOR2+nZ/VasI Pvl RL
                        dhFrBhN53HmRL+VRuT+VGyDYFqq2AJco4NPV8Go= )
;
; Query time: 391 msec
; SERVER: 127.0.0.1#53(127.0.0.1)
; WHEN: Wed Sep 13 09:34:09 2006
; MSG SIZE rcvd: 270
```

And the child zone responses can be validated:

```
# dig +dnssec +multiline SOA sub.dnssec.potaroo.net
; <<>> DiG 9.3.2-P1 <<>> +dnssec +multiline SOA sub.dnssec.potaroo.net
; global options: printcmd
; Got answer:
; -->HEADER<<- opcode: QUERY, status: NOERROR, id: 17696
; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; QUESTION SECTION:
; sub.dnssec.potaroo.net.  IN SOA
;
; ANSWER SECTION:
sub.dnssec.potaroo.net. 86400 IN SOA dns0.dnssec.potaroo.net.  ghi.potaroo.net. (
                        2006091201 ; serial
                        10800   ; refresh (3 hours)
                        15      ; retry (15 seconds)
                        604800  ; expire (1 week)
                        10800   ; minimum (3 hours)
                        )
sub.dnssec.potaroo.net. 86400 IN RRSIG SOA 5 4 86400 20061012035306 (
                        20060912035306 55625 sub.dnssec.potaroo.net.
                        Qh6j xu5LXSFD40pWTMOi IU2rj N74GJNkvdJ1G0I u+j yw
                        nNDKAwAJ4oRAfmp64/P+KZRWHUnMqMpbkWc+12yFANXP
                        GWOxhRFG0dOXUV5i YZShrTzS134CzksmDDQoTI j Qf68v
                        D6Oowm5vBGTL3PmJDDMI Vykk9MPJ4xZHGHmQZUS= )
;
; Query time: 124 msec
```

```
; SERVER: 127.0.0.1#53(127.0.0.1)
; WHEN: Wed Sep 13 09:34:38 2006
; MSG SIZE rcvd: 278
```

The named dnssec log on the resolved also indicates that this is functioning correctly:

```
validating @0x81cc000: dnssec.potaroo.net SOA: starting
validating @0x81cc000: dnssec.potaroo.net SOA: attempting positive response validation
validating @0x81cc000: dnssec.potaroo.net SOA: get_key: creating fetch for dnssec.potaroo.net DNSKEY
validating @0x81cc800: dnssec.potaroo.net DNSKEY: starting
validating @0x81cc800: dnssec.potaroo.net DNSKEY: attempting positive response validation
validating @0x81cc800: dnssec.potaroo.net DNSKEY: verify rdataset: success
validating @0x81cc800: dnssec.potaroo.net DNSKEY: signed by trusted key; marking as secure
validator @0x81cc800: dns_validator_destroy
validating @0x81cc000: dnssec.potaroo.net SOA: in fetch_callback_validator
validating @0x81cc000: dnssec.potaroo.net SOA: keyset with trust 7
validating @0x81cc000: dnssec.potaroo.net SOA: resuming validate
validating @0x81cc000: dnssec.potaroo.net SOA: verify rdataset: success
validating @0x81cc000: dnssec.potaroo.net SOA: marking as secure
validator @0x81cc000: dns_validator_destroy

validating @0x8249000: sub.dnssec.potaroo.net SOA: starting
validating @0x8249000: sub.dnssec.potaroo.net SOA: attempting positive response validation
validating @0x8249000: sub.dnssec.potaroo.net SOA: get_key: creating fetch for
sub.dnssec.potaroo.net DNSKEY
validating @0x8249800: sub.dnssec.potaroo.net DNSKEY: starting
validating @0x8249800: sub.dnssec.potaroo.net DNSKEY: attempting positive response validation
validating @0x8249800: sub.dnssec.potaroo.net DNSKEY: validatezonekey: creating fetch for
sub.dnssec.potaroo.net DS
validating @0x8251000: sub.dnssec.potaroo.net DS: starting
validating @0x8251000: sub.dnssec.potaroo.net DS: attempting positive response validation
validating @0x8251000: sub.dnssec.potaroo.net DS: keyset with trust 7
validating @0x8251000: sub.dnssec.potaroo.net DS: verify rdataset: success
validating @0x8251000: sub.dnssec.potaroo.net DS: marking as secure
validator @0x8251000: dns_validator_destroy
validating @0x8249800: sub.dnssec.potaroo.net DNSKEY: in dsfetched
validating @0x8249800: sub.dnssec.potaroo.net DNSKEY: dsset with trust 7
validating @0x8249800: sub.dnssec.potaroo.net DNSKEY: verify rdataset: success
validating @0x8249800: sub.dnssec.potaroo.net DNSKEY: marking as secure
validator @0x8249800: dns_validator_destroy
validating @0x8249000: sub.dnssec.potaroo.net SOA: in fetch_callback_validator
validating @0x8249000: sub.dnssec.potaroo.net SOA: keyset with trust 7
validating @0x8249000: sub.dnssec.potaroo.net SOA: resuming validate
validating @0x8249000: sub.dnssec.potaroo.net SOA: verify rdataset: success
validating @0x8249000: sub.dnssec.potaroo.net SOA: marking as secure
validator @0x8249000: dns_validator_destroy
```

Success! Of a sort...

Step 5 – Evaluating the Outcome

After working with this, off and on I admit, for more than two weeks, I've been able to partially achieve the initial objective. I've configured a DNSSEC-aware local resolver. I've failed in creating a dnssec-aware application interface that would support an analog of the `gethostbyname()` call, with an additional functional module that would perform a DNSSEC validation of the outcome and raise an alert if the validation pass failed.

The issue here is not the task of configuring the Bind software to perform DNSSEC validation (that's the easy bit!), but that of understanding how to configure trust in the resolver. It appears that the original vision of DNSSEC was that of a self-contained system that started with explicit configuration of the trust anchor of the DNS root, and then use DS chains to have the parent sign the keys of all of its delegated children, and so on. Authentication of a DNS response in such an environment would be that the response matches the signed data in the RRSIG record, and that the key validates against the zone's parent, and so on right up to the root's trust anchor. Like the `named.boot` file that loads the DNS resolver with an initial seed set of bindings for the DNS root servers, the original vision of DNSSEC appeared to encompass a similar single top level injection of trust, from which all other DNSSEC trust relationships could be derived. The current reality of DNSSEC falls far short of this all-encompassing vision, and the task of funding out which zones have been signed, and establishing in a secure manner the keys of each of these DNSSEC "islands" appears to be one where the effort far outweighs the resultant benefit. Even the efforts with the DLV lookaside trust validation tool have been unsatisfying for me, and I've been unable to use one instance of this tool to validate any DNS domains.

I can't see any applications that use DNSSEC-enabled queries into the DNS, nor can I see any realistic mechanism to load my resolver with trust keys that reflect the partial deployment of DNSSEC today. So, from the perspective of a DNS resolver client I must admit that can't see how DNSSEC is relevant today beyond the exercise of technology-proving, leading to the conclusion that DNSSEC could work, and possibly work reasonably well, as long as all DNS zone servers used it!

So how did the zone signing and serving exercise go? My efforts in zone signing appears to have been successful, despite the poor standard of the tool documentation at present. This part of the exercise has taken me over a week to complete, and the basic reason lies in the relatively incomplete state of documentation of the toolset. The changes in the tool interface across Bind versions has not helped, and there are still a number of out-of-date online guides that reference tools that are not part of the Bind9.3.2 toolset that are picked up by the more popular online search engines. The lack of precise step-by-step instructions that show how to publish a signed zone is also a weakness, and I certainly spent some time in a trial-and-error search as to what sequence of actions would produce a linkage between a parent and child zone key set. Yes, I could've requested assistance at any time, and I'd guess that with assistance from someone who had been through this in the past this entire effort could've been accomplished in a few minutes rather than the days it took me. The problem is not in the complexity of the procedures. The problem as I see it is one of scant documentation that does not take the perspective of a potential user of the technology.

I have taken an easy route through some of this work, such as using NSEC records rather than experimentation with NSEC3 records. I have not performed key rollover of the Zone signing key nor the Key-signing key. I have not attempted to perform trusted communication between the master zone server and its secondaries (using TSIG), nor did I attempt to secure the communication between the parent and child zones in order to pass the key set. Even so the experience has left me feeling that DNSSEC, as a packaged technology for mass consumption, is still very rough at the edges.

Further Reading

DNSSEC HOW TO, A Tutorial in Disguise, Olaf Kolkman, RIPE NCC, April 2005,

http://www.ripe.net/disi/dnssec_howto/dnssec_howto.pdf

Overall I found this to be a useful guide to DNSSEC deployment, although I have to admit that Chapter 3 left me slightly dazed and confused

RIPE NCC DNSSEC Training Course material

<http://www.ripe.net/training/dnssec/material/dnssec.pdf>

I wish I had seen this before I started writing this article rather than after. If you are thinking of playing with DNSSEC this is well worth the time to read through – preferably before you start playing!

Bind 9 Administrator Reference Manual, Internet Software Consortium, 2005

<http://www.isc.org/index.pl?sw/bind/arm93/>

Chapter 4 of this guide describes DNSSEC. They've managed to do so in just 636 words, which is perhaps erring too far on the side of brevity. If you weren't familiar with DNSSEC then this won't help much, and if you know all about DNSSEC tools already, then this will not contain anything you shouldn't already know!

DNS and Bind (5th Edition), Paul Albitz & Cricket Liu, O'Reilly, 2006

Yes, this venerable bible of the DNS is now up to its 5th edition, which brings it into sync with Bind 9.3.2. Carefully written, loads of examples. It's a pity that I didn't have the 5th edition beside me when I was attempting this exercise, as Bind 9.3.2 redid its DNSSEC utilities, and the 4th edition of this book is, sadly, outdated in some critical details of DNSSEC utilities.

Pro DNS and BIND, Roy Aithison, Apress, 2005

I've heard really good things about this book, but I have yet to review a copy of this publication.

<http://dnssec.nic.se>

A description of DNSSEC and some details of the initiative to sign the .se top level domain. Actually this contains the .se zone key that I was looking for as a trust anchor last week!

<http://www.dnssec.net>

A resource guide for DNSSEC. A reasonably well ordered set of pointers to DNSSEC material.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

About the Author

GEOFF HUSTON holds a B.Sc. and a M.Sc. from the Australian National University. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and is currently the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of the Internet Society from 1992 until 2001.