

# Securing an Internet Name Server

Cricket Liu

[cricket@verisign.com](mailto:cricket@verisign.com)



# Securing an Internet Name Server

- Name servers exposed to the Internet are subject to a wide variety of attacks:
  - Attacks against the name server code
  - Denial of service attacks
  - Spoofing attacks, which try to induce your name server to cache false resource records
- You should make every effort to protect these name servers from these attacks



# Run a New Version of Your Name Server

- While running the newest version doesn't guarantee your name server's security, it minimizes the possibility of attack
  - Virtually all older name servers have widely known vulnerabilities
- The newest versions of BIND are:
  - 9.1.0 (recommended)
  - 8.2.3
  - 4.9.8 (deprecated)
- For a matrix of vulnerabilities and which versions they exist in, see:
  - <http://www.isc.org/products/BIND/bind-security.html>

- **BIND**

- 9.1.0:
  - <ftp://ftp.isc.org/isc/bind9/9.1.0/bind-9.1.0.tar.gz>
- 8.2.3:
  - <ftp://ftp.isc.org/isc/bind/src/cur/bind-8/bind-src.tar.gz>
- 4.9.8:
  - <ftp://ftp.isc.org/isc/bind/src/4.9.8/bind-4.9.8-REL.tar.gz>



# Eliminate Single Points of Failure

- To combat denial of service attacks and prevent accidental service outages, eliminate single points of failure in your DNS infrastructure
  - Don't put all of your name servers on a single subnet
  - Don't put all of your name servers behind a single router
  - Don't put all of your name servers behind a single leased line
  - Arrange to have someone run an offsite slave name server for you



# Filter Traffic to Your Name Server

- If the hosts that run your name servers don't do anything else, filter out all unnecessary traffic to those name servers
  - Everything but traffic from the Internet to UDP and TCP to port 53

- Restricting zone transfers prevents
  - Others from taxing your name server's resources
  - Hackers from listing the contents of your zones
    - To identify targets
      - Mail servers
      - Name servers
    - To gain "host demographic" information
      - How many hosts you have
      - What makes and models you have
      - What their names are (valuable if you name them after people or projects)



# Restricting Zone Transfers with BIND 4

- With BIND 4.9.x, use the *xfrnets* directive:

```
xfrnets 206.168.119.178&255.255.255.255
```

- This controls which name servers can transfer *any* zones from this name server





# Restricting Zone Transfers with BIND 8

- With BIND 8 or 9, use the *allow-transfer* substatement:

```
options {  
    allow-transfer { 206.168.119.178; };  
};
```

or, specific to a zone:

```
zone "verisign.com" {  
    type master;  
    file "db.verisign.com";  
    allow-transfer { 206.168.119.178; };  
};
```



# Notes on Restricting Zone Transfers

- Remember to restrict zone transfers from slave name servers, not just the primary master
  - It's just as easy to transfer a zone from a slave as it is from the primary master
- *nslookup's ls* command and *dig's axfr* option are implemented as zone transfers



# Authenticate Zone Transfers with TSIG

- With BIND 8.2 and later name servers, you can use transaction signatures, or TSIG, to cryptographically authenticate and verify zone data
- This requires that you configure a key on your primary master name server and slave name servers and instruct the name servers to use the key to sign communication with the other name servers



# Configuring TSIG with BIND 8.2 and Later

- Primary master name server's *named.conf*:

```
key huskymo-tornado. {  
    algorithm hmac-md5;  
    secret "mZiMNOUYQPMNwsDzrX2ENw==";  
};  
  
zone "verisign.com" {  
    type master;  
    file "db.verisign.com";  
    allow-transfer { key huskymo-tornado; };  
};
```

- This tells the name server to expect zone transfer requests from the name server at 206.168.119.178 to be signed with the TSIG key *huskymo-tornado*.

- Slave name server's *named.conf*:

```
key huskymo-tornado. {  
    algorithm hmac-md5;  
    secret "mZiMNOUYQPMNwsDzrX2ENw==";  
};  
  
server 208.8.5.250 {  
    transfer-format many-answers;  
    keys { huskymo-tornado.; };  
};  
  
zone "verisign.com" {  
    type slave;  
    file "bak.verisign.com";  
    allow-transfer { none; };  
};
```

- Remember that TSIG requires time synchronization between the name servers involved
- The name of the key, not just the secret, must match on the servers

- **Dynamic updates are both useful and dangerous**
  - An authorized updater can delete all the records from a zone and add in completely different records
- **If you use dynamic update at all, restrict it as much as possible**
  - To individual addresses
  - To a list of TSIG keys
- **If you use addresses for authentication, make sure you have strong anti-spoofing mechanisms in place**
  - On your border router or
  - On your bastion host



# Restricting Dynamic Updates with BIND

- BIND 8 and 9 understand dynamic updates
- BIND 8 and 9 won't accept dynamic updates to a zone by default
  - You must add an access list to enable dynamic updates
  - Use the *allow-update* substatement:

```
zone "verisign.com" {  
    type master;  
    file "db.verisign.com";  
    allow-update { localhost; key verisign-update-key.; };  
};
```





# Restricting Dynamic Updates with BIND

- If you're only updating records attached to one domain name, you can create a new zone that contains just that name
- For example, if you're just updating the address of *www.verisign.com*:

(in *db.verisign.com*, delegating the new zone)

```
www.verisign.com.    IN    NS    ns1.verisign.com.  
                     IN    NS    ns2.verisign.com.
```



# Restricting Dynamic Updates with BIND

- In *named.conf* on the primary master name server for *www.verisign.com*:

```
zone "www.verisign.com" {  
    type master;  
    file "db.www.verisign.com";  
    allow-update { key www.verisign.com; };  
};
```

- At worst, a malicious updater could change the address of *www.verisign.com* or add subdomains of *www.verisign.com*, but couldn't update the *verisign.com* zone



# Restricting Dynamic Updates with BIND

- With BIND 9, you can use the new *update-policy* substatement to restrict which domain names in a zone can be updated, and by which TSIG keys
- For example:

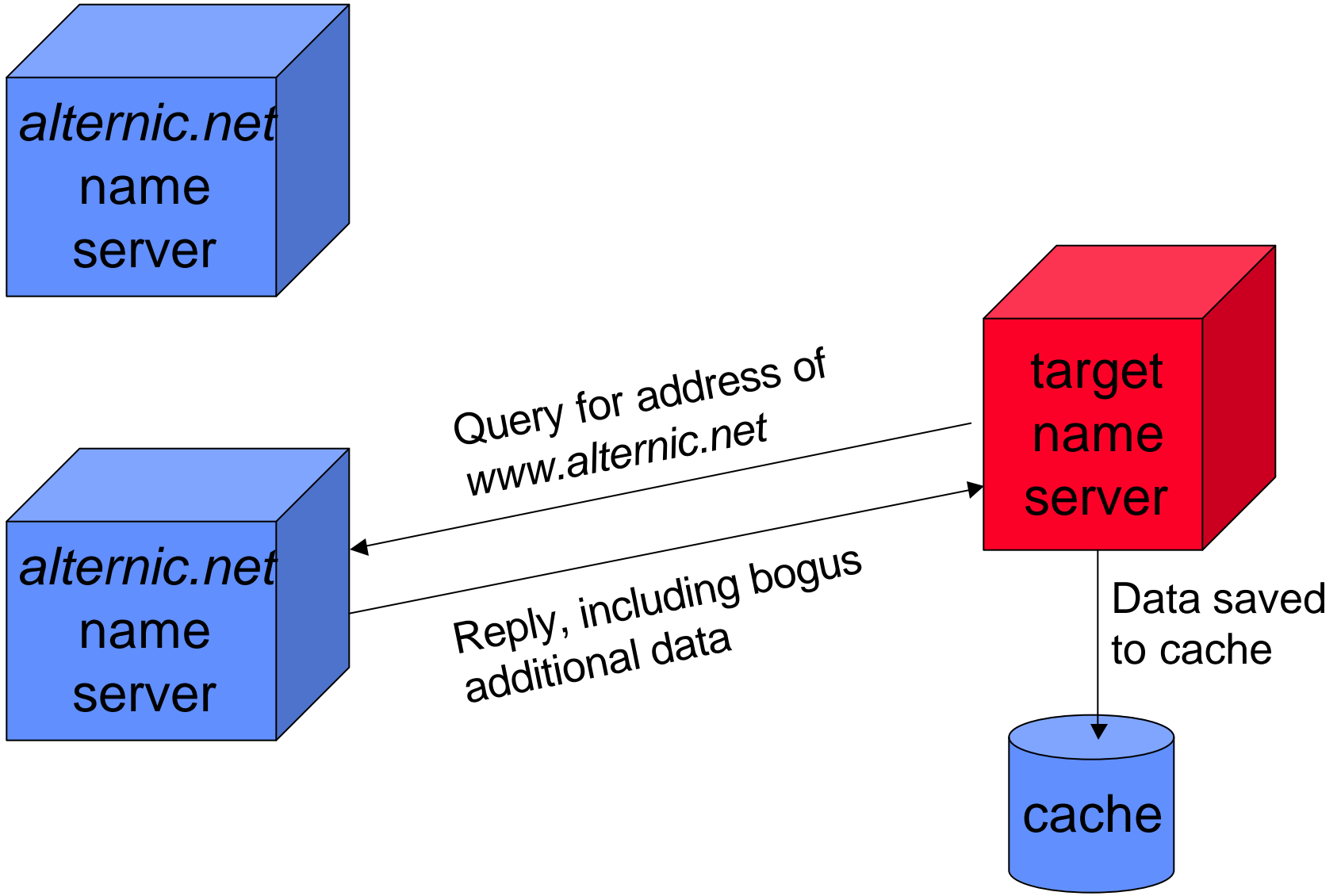
```
zone "verisign.com" {  
    type master;  
    file "db.verisign.com";  
    update-policy {  
        grant www.verisign.com self www.verisign.com A;  
    };  
};
```

- This lets the TSIG key www.verisign.com update only the domain name www.verisign.com's address records.

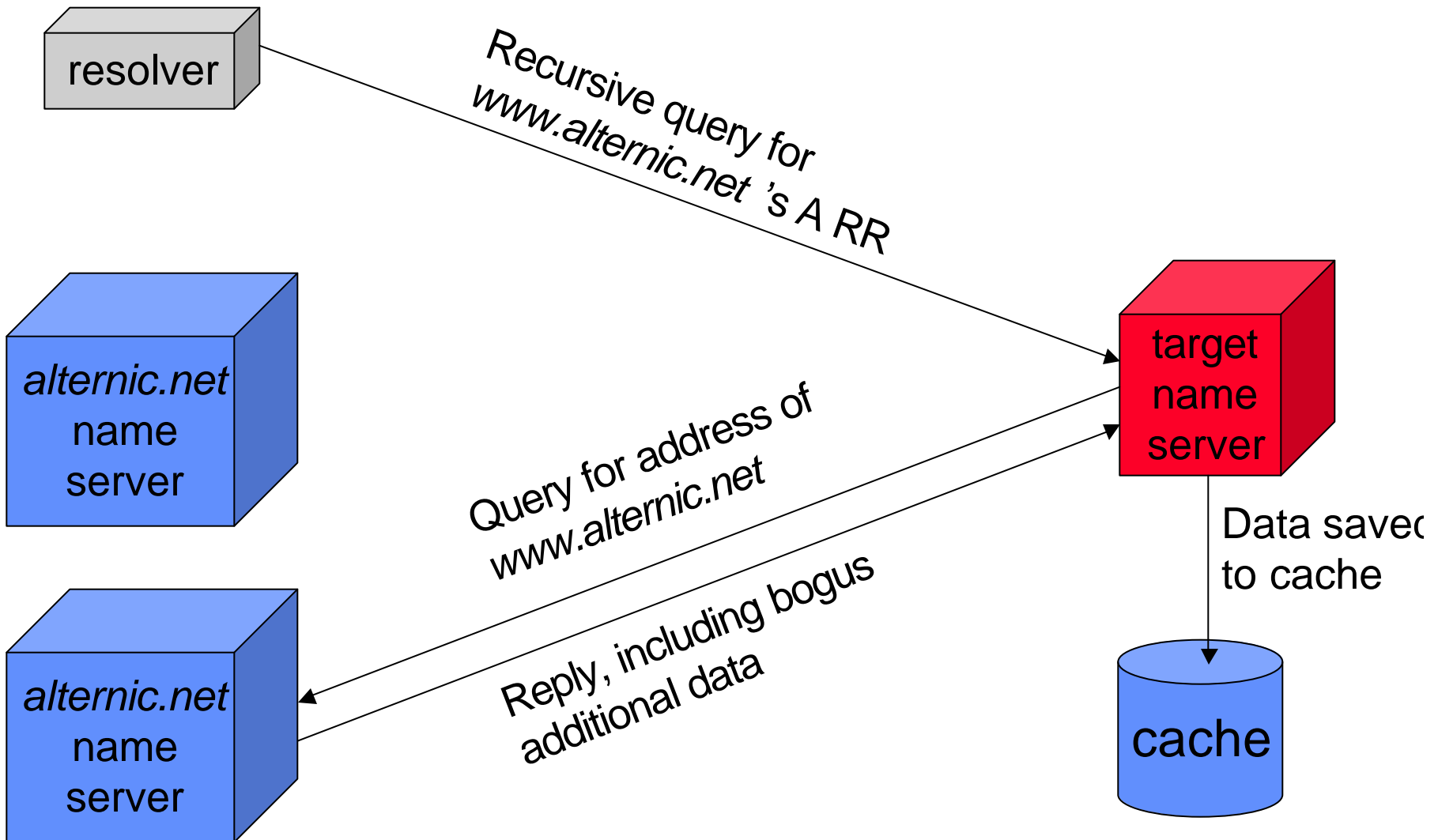
- Accepting recursive queries from the Internet makes your name servers vulnerable to spoofing attacks
  - Hackers can query your name server for information in zones under their control
  - This forces your name server to query their evil name servers, which may spit back bogus data
- To deal with this, you can
  - Turn off recursion, if possible
  - Restrict the addresses the name server will respond to queries from
  - Restrict the addresses the name server will respond to *recursive* queries from



# A Sample DNS Spoofing Attack



# A Sample DNS Spoofing Attack



- Disabling recursion puts your name servers into a passive mode, telling them never to send queries on behalf of other name servers or resolvers
  - A non-recursive name server is very difficult to spoof, since it doesn't send queries, and hence doesn't cache any data
  - You can't disable recursion on a name server if any legitimate resolvers query it, or if other name servers use it as a forwarder
  - If you can't disable recursion, restrict the queries the name server will accept, shown later

- Normally a name server returning NS records for which it does not have A records will attempt to retrieve them
  - This is called *glue fetching*
  - A potential source of a spoofed response
- Turning off glue fetching prevents this lookup
  - The name server will never generate any queries
  - And will not build up a cache





# Turning Off Recursion and Glue Fetching With BIND

- With BIND 4.9, use the *options* directive:

```
options no-recursion  
options no-fetch-glue
```

- With BIND 8, use the *options* statement:

```
options {  
    recursion no;  
    fetch-glue no;  
};
```

- BIND 9's syntax is the same as BIND 8's, but BIND 9 doesn't fetch glue

- If you can't turn off recursion, restrict the queries that your name servers accept to:
  - The addresses they should come from
  - The zones they should ask about
- On most name servers
  - Queries for records in authoritative zones can come from anywhere, because the zones are delegated to the name server
  - Queries for records outside of authoritative zones should only come from internal addresses

- Unfortunately, only BIND 8 and 9 (i.e., not BIND 4) will let you establish such fine-grained access
- Use the *allow-query* substatement:

```
acl internal { 206.168.119.176/29; };

options {
    directory "/var/named";
    allow-query { internal; };
};

zone "verisign.com" {
    type master;
    file "db.verisign.com";
    allow-query { any; };
};
```

- BIND 8.2.1 and later allow you to restrict the IP addresses you accept *recursive* queries from
  - Queriers from other IP addresses will have their recursive queries processed as non-recursive
- Use the *allow-recursion* substatement:

```
acl internal { 206.168.119.178/29; };

options {
    directory "/var/named";
    allow-recursion { internal; };
};

zone "verisign.com" {
    type master;
    file "db.verisign.com";
};
```



# Restrict Recursive Queries with BIND 9

- BIND 9 name servers let you create multiple views
- Clients may see different views based on their IP addresses
  - One view has recursion on
  - One view has recursion off
- For example:

```
view "internal" {  
    match-clients { 206.168.119.176/29; };  
    recursion yes;  
};  
  
view "external" {  
    match-clients { any; };  
    recursion no;  
  
    zone "verisign.com" {  
        type master;  
        file "db.verisign.com";  
    };  
};
```



# More Protection Against Spoofing

- On a BIND 8 name server that queries name servers on the Internet, use *use-id-pool* to randomize message IDs and make spoofing harder

```
options {  
    directory "/var/named";  
    use-id-pool yes;  
};
```

- All BIND 9 name servers use an ID pool



# Run Your Name Server as a User Other Than Root

- To ensure that a vulnerability in BIND doesn't give a hacker root access to your host, run *named* as a user besides root
  - Make sure this user can read your *named.conf* and zone data files
    - And, if you use dynamic update for some zones, can write those zone data files
  - Make sure this user can write to *named*'s PID file
  - To tell *named* to change users to the user *named*, for example, start it with:

```
# named -u named
```

- To ensure that a vulnerability in BIND doesn't give a hacker access to your host's entire filesystem, run your name server *chroot()*d, or confined to a particular directory
  - You'll need copies of important libraries and system files in that directory
  - For more information, see:
    - Chapter 10 of *DNS and BIND*
    - <http://www.linuxdoc.org/HOWTO/Chroot-BIND-HOWTO.html>
    - <http://www.etherboy.com/dns/chrootdns.html>





# Example Configurations

- Here are some example configurations showing you how to put it all together

- A BIND 8 or 9 name server, primary master for a zone, supporting no resolvers, not used as a forwarder:

```
acl slaves { 207.69.231.3; 209.86.147.1; };

options {
    directory "/var/named";
    recursion no;
    fetch-glue no; // for BIND 8 only
    allow-query { any; }; // the default
};

zone "verisign.com" {
    type master;
    file "db.verisign.com";
    allow-transfer { slaves; };
};
```

- This configuration allows anyone to query this name server, but treats all queries as non-recursive

- A BIND 8 or 9 name server, primary master for a zone, that supports one or more resolvers:

```
acl internal { 206.168.119/24; };
acl slaves { 207.69.231.3; 209.86.147.1; };

options {
    directory "/var/named";
    recursion yes;           // the default
    allow-query { internal; };
    use-id-pool yes;         // for BIND 8 only
};

zone "verisign.com" {
    type master;
    file "db.verisign.com";
    allow-transfer { slaves; };
    allow-query { any; };
};
```

- A BIND 8.2.1+ or 9 name server, slave for a zone, that's used as a forwarder:

```
acl internal { 206.168.119/24; };

options {
    directory "/var/named";
    recursion yes;           // the default
    allow-recursion { internal; };
    use-id-pool yes;         // for BIND 8 only
};

zone "verisign.com" {
    type slave;
    masters { 207.69.231.2; };
    file "bak.verisign.com";
    allow-query { any; };    // the default
    allow-transfer { none; };
};
```

- A BIND 8 or 9 caching-only name server:

```
acl internal { 206.168.119/24; };

options {
    directory "/var/named";
    recursion yes;           // the default
    use-id-pool yes;         // for BIND 8 only
    allow-query { internal; };
};

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

- Consider creating two kinds of name server, each optimized for a particular function:
  - *Advertising* name servers:
    - Authoritative for zones to “advertise” to the Internet
    - Listed in parent zones’ NS records
    - Queried only by other name servers
    - Non-recursive
  - *Resolving* name servers:
    - (May be) authoritative for “internal” zones
    - Queried only by known resolvers (or forwarding name servers)
    - Answer recursive queries from trusted sources



# Split-Service Name Server Configs

- An advertising name server:

```
acl slaves { 207.69.231.3; 209.86.147.1; };

options {
    directory "/var/named";
    recursion no;
    fetch-glue no;           // for BIND 8 only
    allow-query { any; };    // the default
};

zone "verisign.com" {
    type master;
    file "db.verisign.com";
    allow-transfer { slaves; };
};
```



# Split-Service Name Server Configs

- A resolving name server:

```
acl internal { 192.168.0/24; };

options {
    directory "/var/named";
    recursion yes;          // the default
    use-id-pool yes;        // for BIND 8 only
    allow-query { internal; };
};

zone "." {
    type hint;
    file "db.cache";
};

zone "verisign.com" {
    type slave;
    masters { 207.69.231.2; };
    file "bak.verisign.com";
    allow-transfer { internal; };
};
```





# Follow Relevant Newsgroups and Mailing Lists

- (Unfortunately) New vulnerabilities are found in name servers all the time
- (Fortunately) These vulnerabilities are usually patched quickly
- If you follow the relevant newsgroups and mailing lists closely, you'll find out about the vulnerabilities and any necessary reconfiguration or patches quickly



# Newsgroups and Mailing Lists Relevant to BIND

- *comp.protocols.dns.bind*
  - (and *bind-users@isc.org*, its mailing list equivalent; join by sending mail to *bind-users-request@isc.org*)
- *comp.protocols.tcp-ip.domains*
- The CERT mailing list (join by sending mail to *cert-advisory-request@cert.org*)
- Your UNIX vendor's security announcement mailing list



## For More Information...

- To learn more about how VeriSign can provide your company with a secure, outsourced DNS solution, please visit our web site at [www.verisign-grs.com](http://www.verisign-grs.com)