

Secured Dynamic Updates

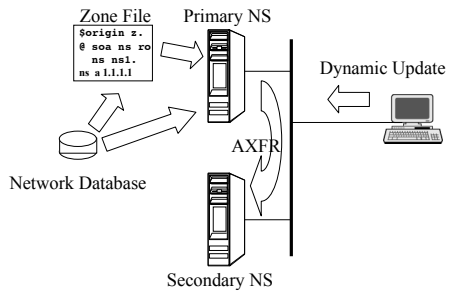
Caution

- Portions of this slide set present features that do not appear in BIND until BIND 9.3
 - Snapshot code is available for this
- BIND 9.2 can perform most of the dynamic update features

Outline

- Dynamic Update Basics
- Setting Up A Dynamic Zone
- Tools
- Securing It
- Authorization Configuration
- Playing with Update Commands
- Interactions with DHCP

Getting Data Into DNS



Advantages of Dynamic Updates

- Change DNS data quickly
- Make changes from anywhere
- No need to reload whole zone
- Data becomes more current

Uses of Dynamic Update

- Adding a new delegation to a large zone
 - Cut down on reload times
- Conference attendees
 - Laptops can use same name, new IP

Risks of Dynamic Update

- Authoritative servers listen to the network for data
- Authorization checks needed before accepting a request
- Server risks being tied up with updates
- Dynamic zones are hard to edit via "the old ways"

Other Considerations

- Once a zone goes dynamic, it is hard to edit
- Mixing dynamic data and critical static data is a bad idea, even neglecting security concerns
- This isn't meant to scare you from dynamic update, but to alert you

"Secure" Dynamic Update

- Secure refers to the safety of the update requests
 - Only the right clients will be able to get data into the zone
- Limitations on the term "secure"
 - Won't stop anyone issuing bad requests
 - Doesn't address DNSSEC, adding digital signatures to the zone

Tools

- In order to do any of this, we need tools (software)
- All are part of a BIND 9 distribution
 - named - the server, concentrating on conf file
 - dig - a query/response tool
 - nsupdate - issues dynamic update messages
 - rndc - remote name server daemon control
 - dnssec-keygen - makes the keys needed

A static zone

```
zone "myzone.example." {  
    type master;  
    file "myzone.example.";  
    allow-transfer { any; };  
};
```

Adding a dynamic zone

```
zone "myzone.example." {  
    type master;  
    file "myzone.example.";  
    allow-transfer { any; };  
};  
zone "dynamic.myzone.example." {  
    type master;  
    file "dynamic.myzone.example.";  
    allow-transfer { any; };  
    allow-update { any; };  
};  
//note: on-line slide is different
```

dynamic.myzone.example

```
> cat db.dynamic.myzone.example

$ORIGIN dynamic.myzone.example.
$TTL 1d      ; 1 day
@           IN      SOA      ns1 root (
                        1      ; serial
                        30m    ; refresh (30 minutes)
                        15m    ; retry (15 minutes)
                        19h    ; expire (19h12m)
                        18min  ; minimum (18min))

NS         ns1.myzone.example
```

Journal Files

- Once a dynamic zone begins running
 - A journal file (<zonefile>.jnl) is created when the first dynamic update has been made
 - This binary, non-text file maintains all updates in recent times
 - Updates aren't immediately reflected in the original <zonefile>, but they are eventually
 - Journal entries are written to the zone file at server shutdown (and on demand)

dig

- Basic debugging aid
- dig @server domain.name type
- Used to verify that change has been made
- Used to verify that SOA number increments

dig examples

```
> dig @127.0.0.1 version.bind chaos txt
> dig @127.0.0.1 myzone.example. soa
+multiline
> dig @127.0.0.1 dynamic.myzone.example. soa
```

nsupdate

- Generates updates based upon user input
- Used to make requested updates

nsupdate example

```
% nsupdate
> server 127.0.0.1
> zone dynamic.myzone.example.
> update add alu.dynamic.myzone.example. 600 A
192.168.160.1
> update add dynamic.myzone.example. 600 MX 10 alu
> send
> quit
```

- Just to check our work...

```
% dig @127.0.0.1 alu.dynamic.myzone.example. A
% dig @127.0.0.1 dynamic.myzone.example. MX
```

rndc

- "Remote" management of server, usually across 127.0.0.1
- Used to stop, reload server
- Used to freeze and unfreeze dynamic zone { available in BIND 9.3 }

rndc examples

```
% rndc -c rndc.conf status
% rndc -c rndc.conf reload
% rndc -c rndc.conf freeze dynamic.myzone.example
% rndc -c rndc.conf unfreeze
dynamic.myzone.example
% rndc -c rndc.conf stop
```

NOTE: "freeze" and "unfreeze" are introduced in BIND 9.3

dnssec-keygen

- Simple tool to generate keys
- Used here to generate TSIG keys

dnssec-keygen tsig example

```
% dnssec-keygen -a HMAC-MD5 -b 128 -n host
sample.tsig.key
Ksample.tsig.key.+157+02308

% ls Ksample*
Ksample.tsig.key.+157+02308.key
Ksample.tsig.key.+157+02308.private
```

"Secured" Dynamic Update

- Limited to the security of the requests
- Dynamic Updates to a DNSSEC zone is a work in progress
- Two steps
 - Identify and authenticate the updater
 - Determine if updater is authorized

Steps

- Create a separate zone for dynamic updates
 - (Done)
- Configure keys
- Configure policy

Configuring Keys

- TSIG
 - works in 9.2, secret needed in named.conf (or “include”) and in client

TSIG keys

- Issue: Naming the key
 - Name is arbitrary, but must be consistent between the named.conf and client
 - There is an advantage to making it the same as a domain in the zone
- To test the keys, turn on key-based authorization of AXFR - just for testing

Making TSIG keys

```
· dnssec-keygen -a HMAC-MD5 -b 128 -n host \  
  slave1.dynamic.myzone.example.  
· dnssec-keygen -a HMAC-MD5 -b 128 -n host \  
  slave2.dynamic.myzone.example.  
  
· ls:  
Kslave1.dynamic.myzone.example.+157+42488.key  
Kslave1.dynamic.myzone.example.+157+42488.private  
  
Kslave2.dynamic.myzone.example.+157+57806.key  
Kslave2.dynamic.myzone.example.+157+57806.private
```

Adding TSIG to named.conf

```
key "slave1.dynamic.myzone.example." {
    algorithm HMAC-MD5;
    secret
    "sd7qi6tiw+N5fK3mGNDNJU9TwIju+1ye7r2shgfkxIg=";
};

key "slave2.dynamic.myzone.example." {
    algorithm HMAC-MD5;
    secret
    "KXMoZHZIIxVsxKp4aUp6YTy3EswUN9CeDEpneJDOgVM=";
};
```

Configuring TSIG AXFR

- Just so we can see that the keys work

```
zone "dynamic.myzone.example." {
    type master;
    file "dynamic.myzone.example.";
    allow-transfer {
        key slave1.dynamic.myzone.example.;
        key slave2.dynamic.myzone.example.;
    };
    allow-update { 127.0.0.1; };
};
```

Testing with dig

- Fails:

```
% dig @127.0.0.1 dynamic.myzone.example. axfr
```

- Succeeds:

```
% dig @127.0.0.1 dynamic.myzone.example. axfr -y
slave1.dynamic.myzone.example.:KXMoZHZIIxVsxKp4aUp6
YTy3EswUN9CeDEpneJDOgVM=
```

- This shows that the TSIG key is properly configured in named.conf

Key based dynamic updates (TSIG)

```
zone "dynamic.myzone.example." {
    type master;
    file "dynamic.myzone.example.";
    allow-transfer {
        key
    };
    slave1.dynamic.myzone.example.;
    slave2.dynamic.myzone.example.;
};
allow-update {
    key user1.dynamic.myzone.example.;
    key user2.dynamic.myzone.example.;
};
};
```

"Keying" nsupdate

- The next three slides show different ways to add key information to nsupdate
 - first hides key from "ps -aux" by entering it interactively
 - second hides it by referencing the file it is in
 - last puts the secret on the command line

Keyed nsupdate #1

```
% nsupdate
> zone dynamic.myzone.example.
> server 127.0.0.1
> key user1.dynamic.myzone.example.
sd7qi6tiw+N5fK3mGNDNJU9TwIju+1ye7r2shgfkxIg=
> update add puri.dynamic.myzone.example. 600 A
192.168.50.1
> send

% dig @127.0.0.1 puri.dynamic.myzone.example A
```

Keyed nsupdate #2

```
% nsupdate -k
Kuser1.dynamic.myzone.example.+157+57806.
> zone dynamic.myzone.example.
> server 127.0.0.1
> update add alu.dynamic.myzone.example. 900 A
192.168.50.2
> Send
```

```
% dig @127.0.0.1 alu.dynamic.myzone.example A
```

Keyed nsupdate #3

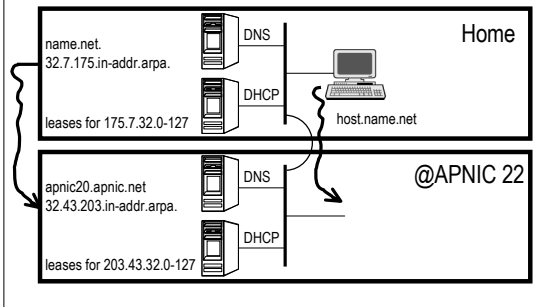
```
% nsupdate -y
Kuser1.dynamic.myzone.example.:sd7qi6tiw+N5fK3mGND
NJU9TwIju+1ye7r2shgfkxIg=
> zone dynamic.myzone.example.
> server 127.0.0.1
> update add palak.dynamic.myzone.example. A 90
192.168.50.2
> send
```

```
% dig @127.0.0.1 palak.dynamic.myzone.example. A
```

Interaction with DHCP

- See the following URL for in-depth information
 - <http://ops.ietf.org/dns/dynupd/secure-ddns-howto.html>

How DHCP and DynUp Look



How This Happens, part 1

- Host has a TSIG to update the entry
host.name.net. A 175.7.32.17
- Home DHCP can change 32.7.175.in-addr.arpa. (via TSIG)
17.32.7.175.in-addr.arpa PTR
host.name.net.
- APNIC22 DHCP can change 32.43.320.in-addr.arpa.
17.32.43.203.in-addr.arpa PTR
host.name.net.

At Lease Change Time

- When releasing home address
 - Home DHCP removes the PTR record
 - Host alters/removes its A RR
 - Done via scripts (depends on DHCP software)
- When gaining APNIC 22 lease
 - APNIC 22 DHCP adds a PTR record
 - Host registers an A RR with the home server

Questions?
