

Secure Dynamic Update A Tutorial



Edward Lewis
<edlewis@arin.net>

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

www.arin.net

Outline

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

Questions?



Please ask questions throughout the tutorial...

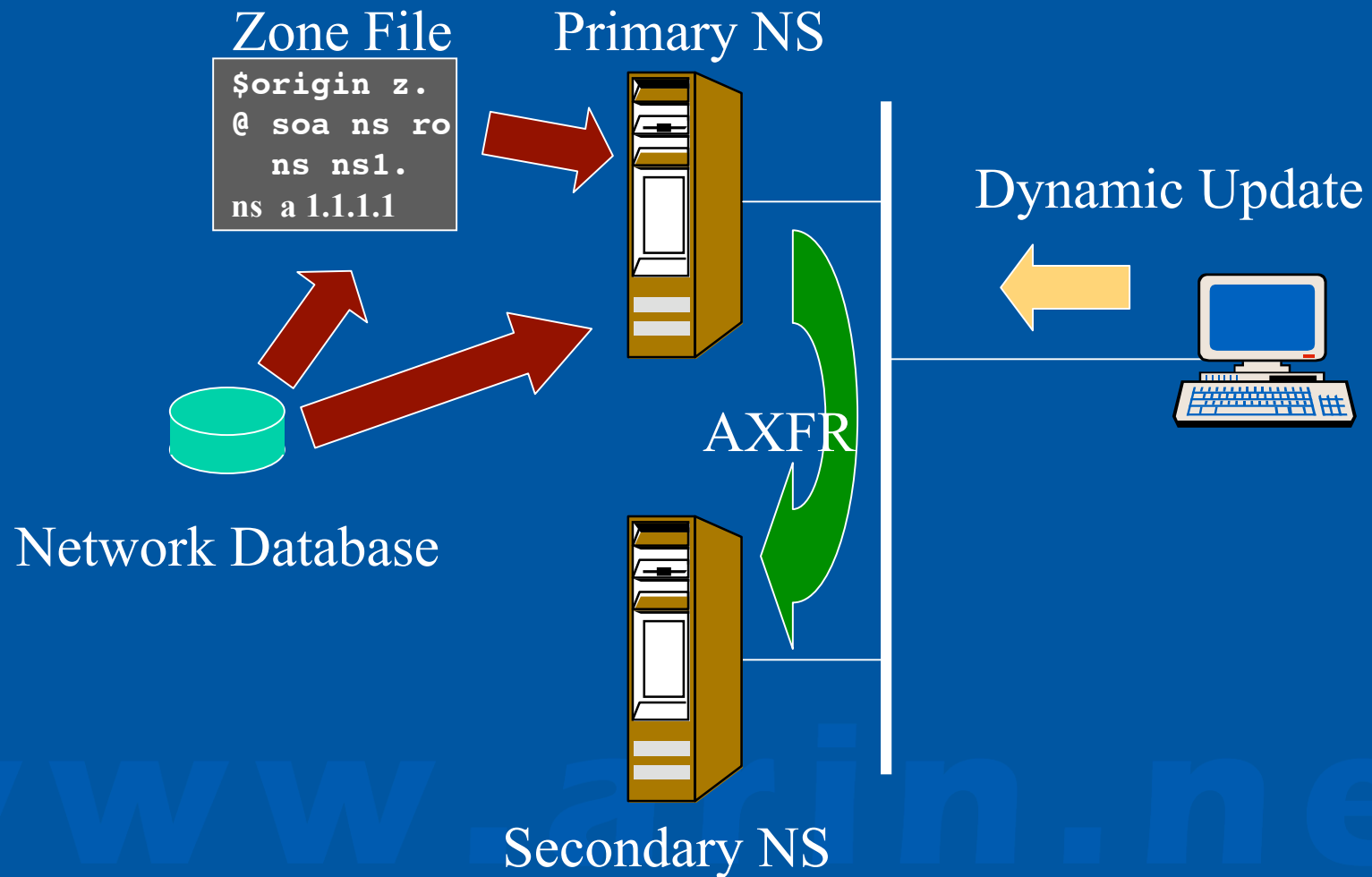
www.arin.net

Dynamic Update Basics

- What It Accomplishes
- What It Offers
- What Is At Risk
- What do I mean by security?

www.arin.net

Getting Data Into DNS



Advantages of Dyn Up's

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

www.arin.net

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
- Others...

www.arin.net

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"

www.arin.net

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

www.arin.net

"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

www.arin.net

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

www.arin.net

named

- It's pretty obvious we need the name server
- The next few slides will show the addition of a dynamic zone

www.arin.net

named.conf snippets

- Throughout the tutorial parts of named.conf will be shown
 - Because showing the named.conf in PowerPoint is difficult, pieces are shown
 - A full version of the tutorial named.conf will be available "some other way"

www.arin.net

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-update { any; };  
};  
//note: on-line slide is different
```

www.arin.net

dynamic.myzone.example

```
[lead] % cat dynamic.myzone.example.  
$ORIGIN dynamic.myzone.example.  
$TTL 86400          ; 1 day  
@ IN SOA localhost. root.localhost. (  
    1          ; serial  
    1800       ; refresh (30 minutes)  
    900        ; retry (15 minutes)  
    69120      ; expire (19h12m)  
    1080       ; minimum (18 minutes)  
    )  
@   NS   localhost.
```

www.arin.net

Adding logging

```
logging
{
    category update { update_log; };
    channel update_log
    {
        file "logs/dns-update.log"
        versions 2 size 20m;
        print-time yes;
        print-category yes;
        print-severity yes;
        severity info;
    };
};
```

www.arin.net

Journal Files

- Once a dynamic zone begins running
 - A journal file (<zonefile>.jnl) is created
 - 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

www.arin.net

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`

www.arin.net

nsupdate

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

www.arin.net

nsupdate example

```
% nsupdate
```

```
> server 127.0.0.1
```

```
> zone dynamic.myzone.example.
```

```
> update add a 900 TXT "new data"
```

```
> send
```

```
> quit
```

- Just to check our work...

```
% dig @127.0.0.1 a.dynamic.myzone.example. txt
```

- "server" needed because of "lab" setup

rndc

- "Remote" management of server, usually across 127.0.0.1
- Used to stop, reload server
- Used to freeze and unfreeze dynamic zone

www.arin.net

rndc examples

```
rndc -c rndc.conf status
```

```
rndc -c rndc.conf freeze dynamic.myzone.example
```

```
rndc -c rndc.conf unfreeze dynamic.myzone.example
```

```
rndc -c rndc.conf reload
```

```
rndc -c rndc.conf stop
```

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

www.arin.net

dnssec-keygen

- Simple tool to generate keys
- Used for DNSSEC too
- Used here to generate TSIG keys
- Used also to generate SIG(0) keys - in version 9.3

www.arin.net

dnssec-keygen tsig example

```
[lead]% dnssec-keygen -a HMAC-MD5 -b 128 -n host  
sample.tsig.key
```

```
Ksample.tsig.key.+157+02308
```

```
[lead]% ls Ksample*
```

```
Ksample.tsig.key.+157+02308.key
```

```
Ksample.tsig.key.+157+02308.private
```

- On older FreeBSD and on MacOS X, you will need to supply entropy

- -r option or you are asked to type randomly

www.arin.net

dnssec-keygen sig(0) example

```
[lead]% dnssec-keygen -a RSA -b 512 -n host  
sample.tsig.key
```

```
Ksample.tsig.key.+001+18681
```

```
[lead]% ls Ksam*
```

```
Ksample.tsig.key.+001+18681.key
```

```
Ksample.tsig.key.+157+02308.key
```

```
Ksample.tsig.key.+001+18681.private
```

```
Ksample.tsig.key.+157+02308.private
```

```
[lead]%
```

www.arin.net

"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

www.arin.net

Steps

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

www.arin.net

Configuring Keys

- Two styles
 - TSIG - shared secret
 - SIG (0) - public key
- TSIG
 - works in 9.2, secret needed in named.conf (or \$include) and in client
- SIG(0)
 - **needs 9.3**, public key listed in the zone file (not in named.conf) and private key in client

www.arin.net

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, I'll also turn on key-based authorization of AXFR - just for testing

Making TSIG keys

- `dnssec-keygen -a HMAC-MD5 -b 128 -n host \`
`four.dynamic.myzone.example.`
- `dnssec-keygen -a HMAC-MD5 -b 128 -n host \`
`five.dynamic.myzone.example.`
- `ls:`

`Kfive.dynamic.myzone.example.+157+42488.key`

`Kfive.dynamic.myzone.example.+157+42488.private`

`Kfour.dynamic.myzone.example.+157+57806.key`

`Kfour.dynamic.myzone.example.+157+57806.private`

Adding TSIG to named.conf

```
key "four.dynamic.myzone.example." {  
    algorithm HMAC-MD5;  
    secret  
    "sd7qi6tiw+N5fK3mGNDNJU9TwIju+1ye7r2shgfkxIg=";  
};  
  
key "five.dynamic.myzone.example." {  
    algorithm HMAC-MD5;  
    secret  
    "KXMoZHZIIxVsxKp4aUp6YTy3EswUN9CeDEpneJD0gVM=";  
};
```

www.arin.net

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 five.dynamic.myzone.example.;  
        key four.dynamic.myzone.example.;  
    };  
    allow-update { 127.0.0.1; };  
};
```

www.arin.net

Testing with dig

- Fails:
 - `dig @127.0.0.1 dynamic.myzone.example. axfr`
- Succeeds:
 - `dig @127.0.0.1 dynamic.myzone.example. axfr -y five.dynamic.myzone.example.:KXMoZHZIIxVsxKp4aUp6YTy3EswUN9CeDEpneJD0gVM=`
- This shows that the TSIG key is properly configured in named.conf

www.arin.net

Configuring a loose policy

```
zone "dynamic.myzone.example." {  
    type master;  
    file "dynamic.myzone.example."  
    allow-transfer {  
        key five.dynamic.myzone.example.;  
        key four.dynamic.myzone.example.;  
    };  
    allow-update {  
        key five.dynamic.myzone.example.;  
        key four.dynamic.myzone.example.;  
    };  
};
```

www.arin.net

"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

www.arin.net

Keyed nsupdate #1

```
[lead] % nsupdate
```

```
> zone dynamic.myzone.example.
```

```
> server 127.0.0.1
```

```
> key four.dynamic.myzone.example.  
sd7qi6tiw+N5fK3mGNDNJU9TwIju+1ye7r2shgfkxIg=
```

```
> update add four.dynamic.myzone.example. 900 TXT "I  
just added this"
```

```
> send
```

```
[lead] % dig @127.0.0.1 four.dynamic.myzone.example  
txt
```

www.arin.net

Look in the logs!

- Logs show this when an update succeeds (all on one line)
- `12-Aug-2002 13:56:35.512 update: info: client 127.0.0.1#49386: updating zone 'dynamic.myzone.example/IN': adding an RR at 'four.dynamic.myzone.example' TXT`

www.arin.net

Keyed nsupdate #2

```
[lead] % nsupdate -k
```

```
Kfour.dynamic.myzone.example.+157+57806.
```

```
> zone dynamic.myzone.example.
```

```
> server 127.0.0.1
```

```
> update add six.dynamic.myzone.example. 900 TXT "I  
just added this"
```

```
> send
```

```
[lead] % dig @127.0.0.1 four.dynamic.myzone.example  
txt
```

www.arin.net

Keyed nsupdate #3

```
[lead] % nsupdate -y
    four.dynamic.myzone.example.:sd7qi6tiw+N5fK3mG
    NDNJU9TwIju+1ye7r2shgfkxIg=

> zone dynamic.myzone.example.

> server 127.0.0.1

> update add seven.dynamic.myzone.example. 900
    TXT "I just added this"

> send

[lead] % dig @127.0.0.1
    four.dynamic.myzone.example txt
```

www.arin.net

A tighter policy

- Allow-update permits changes to "seven" with a key named "four"
- This may not be desirable
- **"update-policy"** is the new keyword

www.arin.net

an update-policy

```
zone "dynamic.myzone.example." {  
    type master;  
    file "dynamic.myzone.example."  
    allow-transfer {  
        key five.dynamic.myzone.example.;  
        key four.dynamic.myzone.example.;  
    };  
    update-policy {  
        grant * self * A TXT;  
    };  
};
```

www.arin.net

Previous slide's update-policy

- Restricts a key's authorization to make changes to just its matching domain name and to just A and TXT records
- This is why matching TSIG keys to domain names in the zone is a good idea
- There are more variations on the update-policy, this is just the simplest

www.arin.net

Retrying keyed nsupdate #1

```
[lead] % nsupdate
> zone dynamic.myzone.example.
> server 127.0.0.1
> key four.dynamic.myzone.example.
    sd7qi6tiw+N5fK3mGNDNJU9TwIju+1ye7r2shgfkxIg=
> update add four.dynamic.myzone.example. 900 TXT "I
    just added this"
> send

[lead] % tail -1 logs/dns-update.log
12-Aug-2002 14:12:51.045 update: info: client
127.0.0.1#49386: updating zone
'dynamic.myzone.example/IN': adding an RR at
'four.dynamic.myzone.example' TXT
```

Retrying #2

```
[lead] % nsupdate -k  
      Kfour.dynamic.myzone.example.+157+57806.  
> zone dynamic.myzone.example.  
> server 127.0.0.1  
> update add six.dynamic.myzone.example. 900 TXT "I  
      just added this"  
> send
```

```
[lead] % tail -1 logs/dns-update.log  
12-Aug-2002 14:13:58.432 update: info: client  
      127.0.0.1#49387: updating zone  
      'dynamic.myzone.example/IN': update failed:  
      rejected by secure update (REFUSED)
```

- "four" isn't permitted to change "six."

Closer look at update-policy

- `update-policy {grant * self * A TXT;};`
- Syntax definition
 - (grant | deny) identity nametype name [types]
 - First matching rule is used
 - Grant explicitly permits, Deny explicitly "denies"
- Why is this better?
 - Fine grained access control
 - Rarely will one key be allowed to change "anything"

www.arin.net

update-policy statement

- `(grant | deny) identity
nametype name [types]`
 - identity = key name
 - nametype = how to interpret name
 - name = name to match
 - types = permitted changes
 - types default to all but SOA, NS, SIG and NXT
 - type "all" means all but NXT

www.arin.net

more complex example

```
zone "dynamic.myzone.example." {  
    ...  
    update-policy {  
        grant admin.dynamic.myzone.example.  
            subdomain dynamic.myzone.example.  
            any;  
        grant intern.dynamic.myzone.example.  
            wildcard *.dynamic.myzone.example.  
            any;  
        grant * self * A TXT;  
    };  
};
```

www.arin.net

SIG(0) keys

- Available in BIND 9.3 (snapshots) only
- Trails TSIG in maturity
- Has some advantages over TSIG
 - Secret is only with client
 - Public key at server is visible (good for debugging)

www.arin.net

Generating a SIG(0) key

- `dnssec-keygen -a RSA -b 1024 -n host one.dynamic.myzone.example.`
- The *.key file is added to the zone file
 - `cat Kone*.key >> dynamic.myzone.example.`
- The *.private key is needed by `nsupdate`
 - only the "`nsupdate -k <file>`" will work

www.arin.net

New zone file

```
$ORIGIN dynamic.myzone.example.  
$TTL 86400 ; 1 day  
@ IN SOA localhost. root.localhost. (  
    1 ; serial  
    1800 ; refresh (30 minutes)  
    900 ; retry (15 minutes)  
    69120 ; expire (19 hrs 12 minutes)  
    1080 ; minimum (18 minutes)  
)  
    NS localhost.  
one KEY 512 3 1 AQOu...8RpAQ==  
two KEY 512 3 1 AQPU...UoU8Q==  
three KEY 512 3 1 AQOf.../FdiQ==
```

nsupdate with SIG(0)

```
[lead] % nsupdate -k  
      Kone.dynamic.myzone.example.+001+50281.
```

```
> server 127.0.0.1
```

```
> zone dynamic.myzone.example.
```

```
> update add one 900 txt "adding this"
```

```
> show
```

Outgoing update query:

```
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id: 0
```

```
;; flags: ; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
```

```
;; UPDATE SECTION:
```

```
one.dynamic.myzone.example. 900 IN TXT "adding this"
```

```
> send
```

Other Dynamic Updates

- A non-exhaustive list
- deletes

```
update delete one 900 txt "adding this"
```

```
update delete one 900 txt
```

```
update delete one
```

- prerequisites

```
prereq nxdomain one
```

```
prereq yxdomain one
```

```
prereq nxrrset one txt
```

```
prereq yxrrset one txt
```

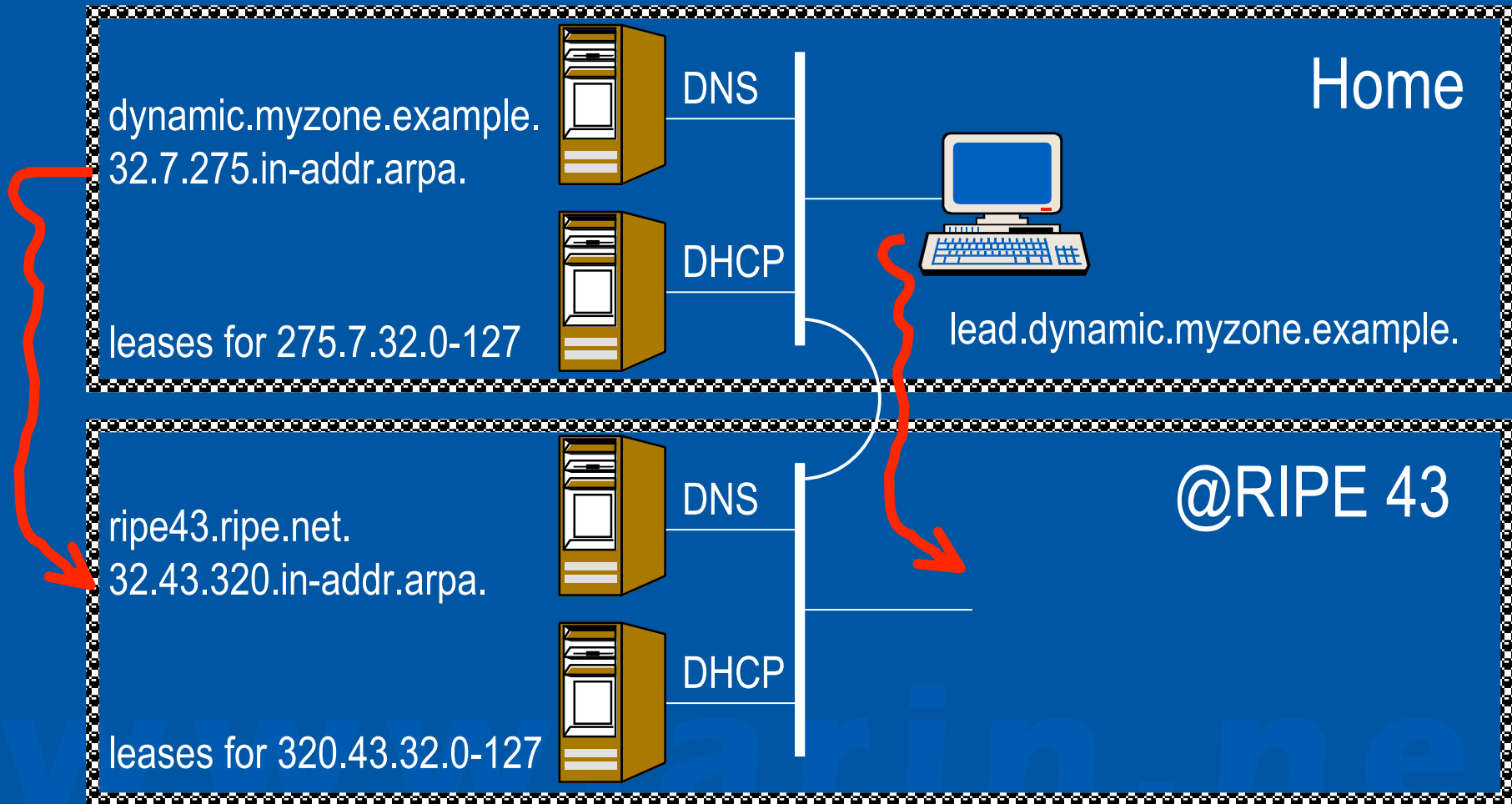
www.arin.net

Interaction with DHCP

- See the following URL for in-depth information
 - <http://ops.ietf.org/dns/dynupd/secure-ddns-howto.html>
- I'll cover some basics here, using the example of the RIPE 43 meeting network

www.arin.net

How DHCP and DynUp Look



How This Happens, part 1

- Host has a TSIG/SIG(0) to update the entry

```
lead.dynamic.myzone.example A 275.7.32.17
```

- Home DHCP can change 32.7.275.in-addr.arpa. (via TSIG/SIG(0))

```
17.32.7.275.in-addr.arpa PTR lead.dynamic...example.
```

- RIPE43 DHCP can change 32.43.320.in-addr.arpa.

```
17.32.43.320.in-addr.arpa PTR lead.dynamic...example.
```

www.arin.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 RIPE 43 lease
 - RIPE 43 DHCP adds a PTR record
 - Host registers an A RR with the home server

www.arin.net

Open Issues

- "Cleaning up"
- What happens when leases aren't explicitly released, e.g., when does DHCP remove "dead" PTR records
- What does a host do while it is in transit, i.e., in my backpack on a plane?

www.arin.net

Wrap-Up

- This presentation should have
 - Briefly discussed what Dynamic Update is
 - Described creating a securely updated zone
 - TSIG and SIG(0) keys for authentication
 - allow-update and update-policy for authorization
 - Covered how conference DHCP can make use of dynamic updates