DNSSEC deployment from server and client side

Petr Špaček & Tomáš Hozza

DNSSEC deployment from server and client side

- Brief introduction
- Generating signing keys
- Manual signing of a zone in BIND
  - messing with signatures
- Automatic zone signing in BIND
- Automagic zone signing in FreeIPA
- Client side with unbound and dnssec-trigger
  - `www.dnssec-failed.org`

This is a workshop! :-)

- Connect to **wired** network
- Install DNS utilities and daemons

```
$ sudo yum install bind bind-utils ldns-utils
```

  - bind-utils package will install DNS root trust anchor to `/etc/trusted-key.key` (What a great name!)
- Open port 53 in your firewall:

```
$ sudo firewall-cmd --add-service=dns
```

  or

```
$ sudo iptables -I INPUT -p tcp --dport 53 -j ACCEPT
$ sudo iptables -I INPUT -p udp --dport 53 -j ACCEPT
```

- **Be nice** and thankful to

  Brno University of Technology

Creating your own DNS zone

and joining global DNS tree

Creating a shiny new DNS zone

```
$ wget http://test.devconf.cz/fasnick.db \
   -O /var/named/dynamic/${fasnick}.db
```

- Change IP addresses in the file to match reality
- Fix permissions:

```
$ chgrp named /var/named/dynamic/${fasnick}.db
```

- Add the zone to `/etc/named.conf`:

```
zone "${fasnick}.test.devconf.cz." IN {
      type master;
      file "dynamic/${fasnick}.db";
};
```

- Restart BIND: `$ systemctl restart named`
- Test is locally:

```
$ dig @localhost ${fasnick}.test.devconf.cz.
```

Testing the new zone

- Test if your new zone is **reachable from the Internet**

- Make sure you **do not** have your own local server in **`/etc/resolv.conf`**!

```
$ dig ${fasnick}.test.devconf.cz.
```

- Does it work?

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR ???
```

Fixing the new zone

```
$ dig ${fasnick}.test.devconf.cz.
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN
```

- Delegation from the parent zone is missing!

```
$ nsupdate -y HMAC-SHA512:keyname:keyvalue
> server testns.devconf.cz.
> update add ns.${fasnick}.test.devconf.cz. 10 IN A \
              ${public IP address of your machine}
> update add ${fasnick}.test.devconf.cz. 10 IN NS \
              ns.${fasnick}.test.devconf.cz.
> send
```

Testing the new zone

- Test if your new zone is reachable **from the Internet**

- Make sure you **do not** have your own local server in **`/etc/resolv.conf`**!

```
$ dig ${fasnick}.test.devconf.cz.
```

- Does it finally work?

DNSSEC theory

finally ;-)

What is DNSSEC

- DNS Security Extensions
- DNS data + signatures (public key crypto)
  - Effectively prevents DNS spoofing
  - Enables new DNS applications:
    - Binding X.509 CA and a domain name
    - TLS certificate validation
      (e.g. for self-signed certs)
    - PGP key distribution
    - IPsec key distribution
    - more to come

How DNSSEC works

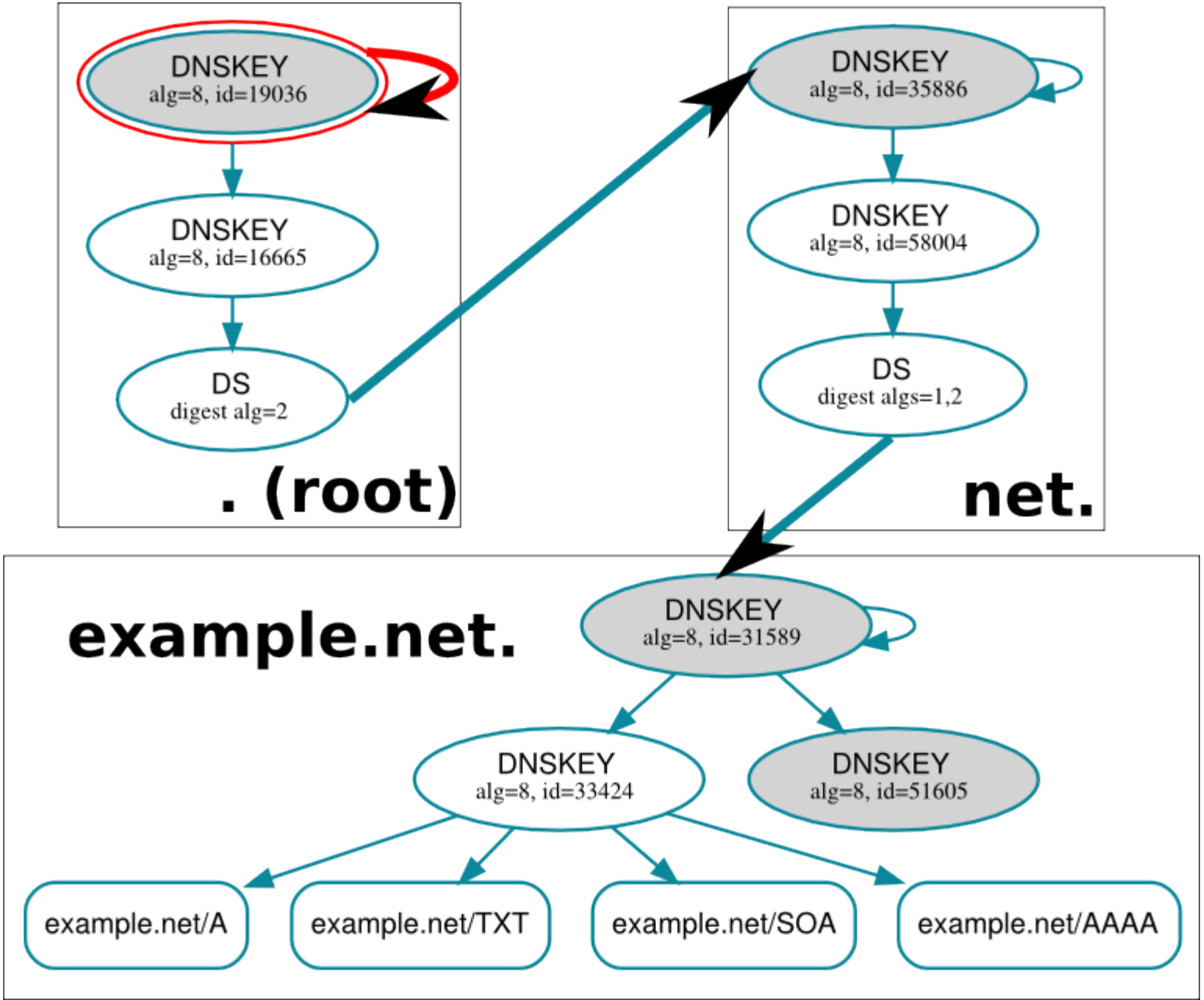
- Root of DNS tree = well-known entry point
  - `redhat.com.`

- **Validating** resolvers know public key of DNS root (the default trust anchor)
  - This public key is distributed with resolver software

- Parent DNS zone publishes hashes of public keys used by its child domains

➢ **Chain of trust from root downwards**

Chain of trust from root downwards



See dnsviz.net

Chain of trust from root downwards: DNS root trust anchor

- Public key used by DNS root "." is installed on every validating resolver:

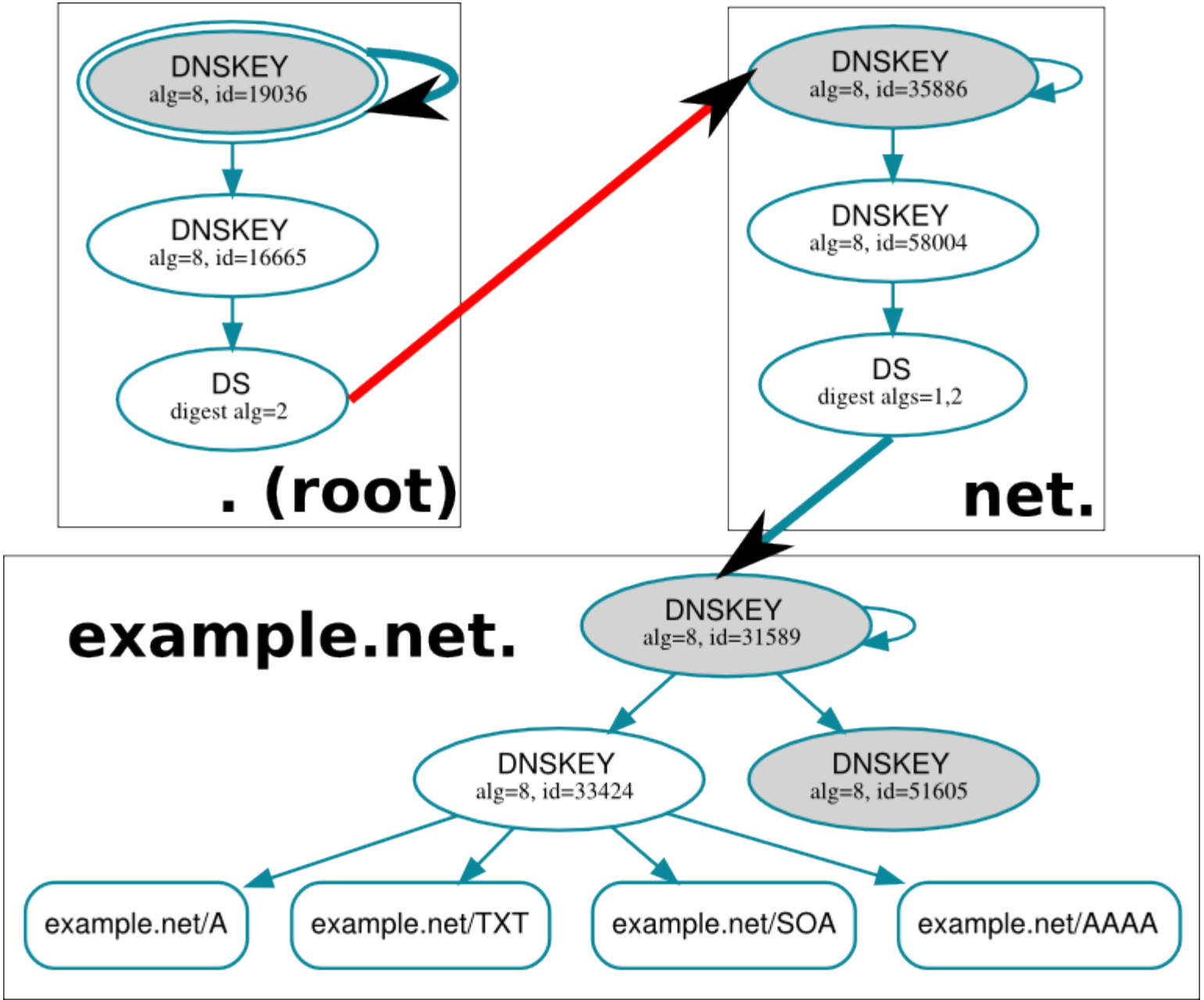
```
$ cat /etc/trusted-key.key
. 3600 IN DNSKEY 257 3 8
AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQbSEW0O8gcCjFFVQU
<snip>

$ dig . DNSKEY
```

- Alternative: hashes in form of DS records

```
$ dnssec-dsfromkey -f /etc/trusted-key.key .
. IN DS 19036 8 1 B256BD09DC8DD59F0E0F0D8541B8328DD986DF6E
. IN DS 19036 8 2 49AAC11D7B6F6446702E54A1607371607A1A41855200FD2CE1CDDE32F24E8FB5
```

Chain of trust from root downwards: DS



See dnsviz.net

Chain of trust from root downwards: Delegation Signer

- DNS root "." publishes hashes of public keys used by top-level domain "net.":

```
$ dig +trace +dnssec net. DS
.           393056   IN    NS    l.root-servers.net.
<snip>
net.        86400    IN    DS    35886 8 2
7862B27F5F516EBE19680444D4CE5E762981931842C465F00236401D 8BD973EE

net.        86400    IN    RRSIG    DS 8 1 86400 20150124050000
20150114040000 16665 .
MfJ2jhBa+tswIOrZIOBqbjRmhh4E+6xkWstRRe/uxmVAZ7/lrifqM01
<snip>

;; Received 239 bytes from 199.7.83.42#53(l.root-servers.net) in 37 ms
```

Chain of trust from root downwards: DNSKEY

- Public keys used by “net.” can be obtained directly from “net.” DNS servers:

```
$ dig +trace +dnssec net. DNSKEY
net.          172800   IN    NS    j.gtld-servers.net.
<snip>

net.          86400    IN    DNSKEY    257 3 8
AQOYBnzqWXIEj6mlgXg4LWC0HP2n8eK8XqgHlmJ/69iuIHsa1TrHDG6T
<snip>

;; Received 889 bytes from
192.48.79.30#53(j.gtld-servers.net) in 466 ms
```

Chain of trust from root downwards: Matching DS and DNSKEY records

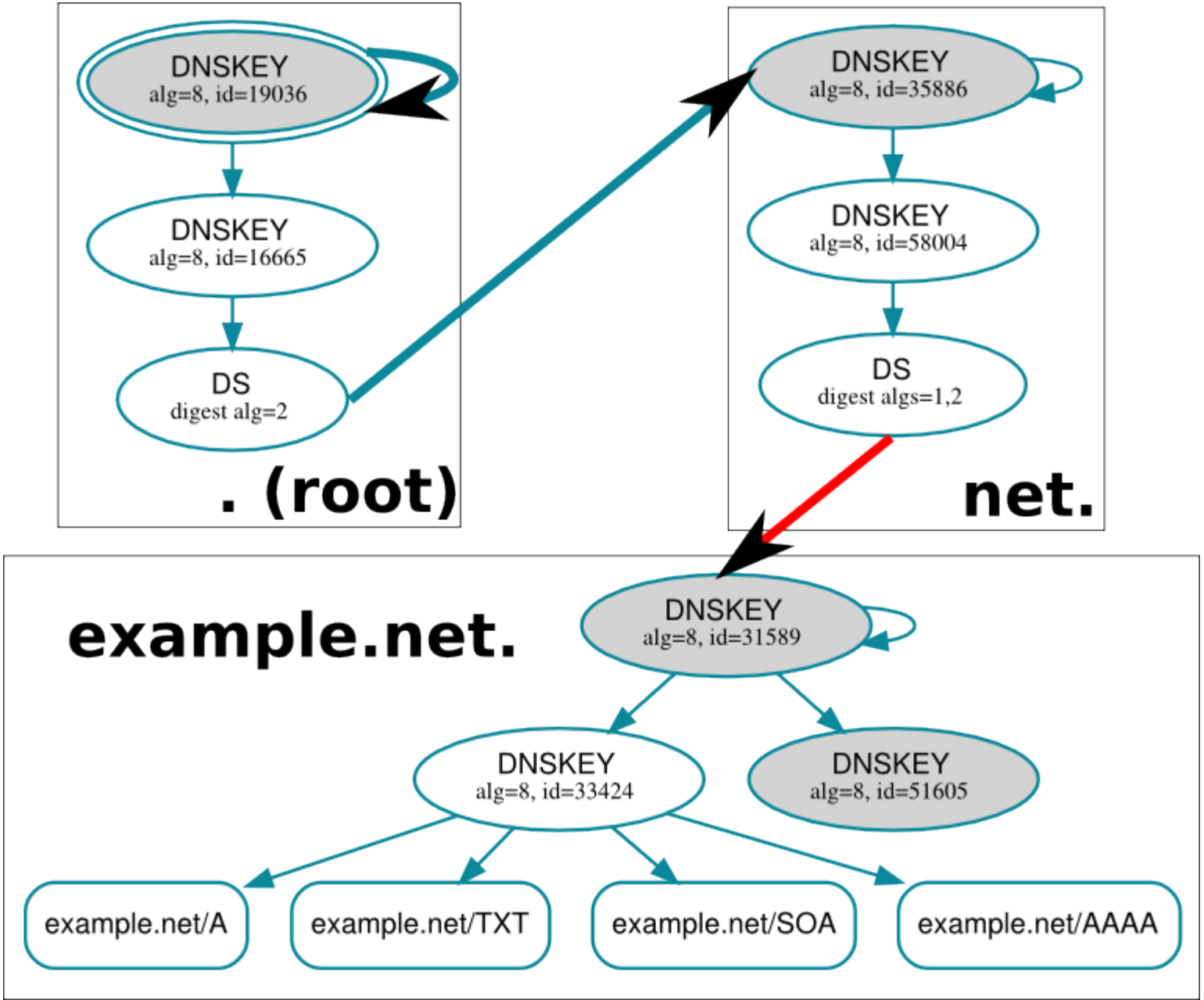
- Do DNSKEY and DS records match?

```
$ dig net. DNSKEY > net.dnskey
$ dnssec-dsfromkey -f net.dnskey net.
net.                    IN      DS      35886 8 1
466A9EDD47858E9E06944FC02B5AE19DBCBA7EC8

net.                    IN      DS      35886 8 2
7862B27F5F516EBE19680444D4CE5E762981931842C465F00236401D8B
D973EE

$ dig net. DS
net.    86400   IN   DS   35886 8 2
7862B27F5F516EBE19680444D4CE5E762981931842C465F00236401D8B
D973EE
```

Chain of trust from root downwards



See dnsviz.net

Result: Hierarchical trust model

- `my.example.net.` can be spoofed **only** by its parents:
  - `example.net.`
  - `net.`
  - DNS root `.`

- Compare situation with X.509 PKI:
- There were **1,482** CA Certificates trustable by Windows or Firefox in 2010
  - See

    `https://www.eff.org/observatory`

Proof of non-existence

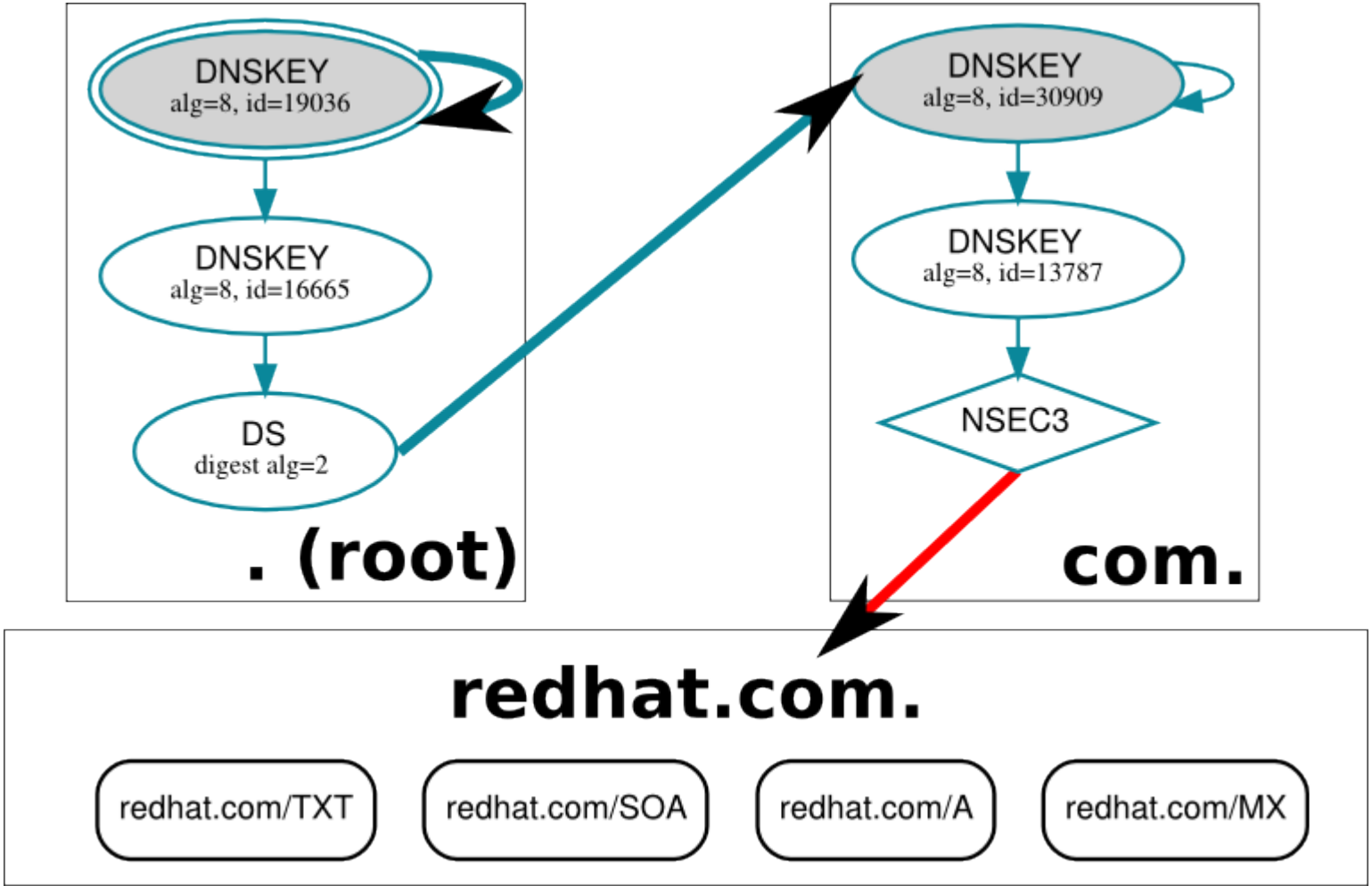
- Signed information that something does not exist in DNS tree:

```
$ dig +dnssec \
hopefully.nonexistent.example.net.

example.net.        3600    IN    NSEC    www.example.net.
A NS SOA TXT AAAA RRSIG NSEC DNSKEY
```

Mixing signed & unsigned zones

- Proof of non-existence of DS record in com. domain



Generation of KEYs

KSK & ZSK

KSK & ZSK

- Generation of KSK

```
$ dnssec-keygen -a RSASHA256 -b 4096 \
    -f KSK ${fasnick}.test.devconf.cz.
```

- Generation of ZSK

```
$ dnssec-keygen -a RSASHA256 \
    -b 1536 ${fasnick}.test.devconf.cz.
```

- In this example use `-r /dev/urandom` for faster key generation.
- Key-pair
  - `K<name>+<alg>+<id>.key`
  - `K<name>+<alg>+<id>.private`

Manual zone signing

with BIND 9

Manual zone signing

- sign the zone file in the **/var/named/dynamic**

```
$ dnssec-signzone -S \
   -o ${fasnick}.test.devconf.cz. ${fasnick}.db
```

- signed zone - **${fasnick}.db.signed**
- Adjust the **/etc/named.conf**

```
zone "${fasnick}.test.devconf.cz." IN {
     type master;
     file "dynamic/${fasnick}.db.signed";
};
```

- Reload BIND: **# systemctl reload named**

Testing the chain of trust

```
$ drill -TD -k /etc/trusted-key.key \
   ${fasnick}.test.devconf.cz. DNSKEY
```

Uploading DS record to parent zone

```
$ dig @localhost <fasnick>.test.devconf.cz \
   DNSKEY > /tmp/your-keys \
   <fasnick>.test.devconf.cz

$ dnssec-dsfromkey -T 10 -f /tmp/your-keys \
   <fasnick>.test.devconf.cz

$ nsupdate -y HMAC-SHA1:keyname:keyvalue
> server testns.devconf.cz
> update add <copy & paste here the DS record>
> send
```

Testing the chain of trust

```
$ drill -S -k /etc/trusted-key.key \
   ${fasnick}.test.devconf.cz.
```

- example output

```
;; Number of trusted keys: 1
;; Chasing: devconf.cz. A

DNSSEC Trust tree:
devconf.cz. (A)
|---devconf.cz. (DNSKEY keytag: 18620 alg: 7 flags: 256)
      |---devconf.cz. (DNSKEY keytag: 4515 alg: 7 flags: 257)
      |---devconf.cz. (DS keytag: 4515 digest type: 1)
      |---cz. (DNSKEY keytag: 12305 alg: 10 flags: 256)
            |---cz. (DNSKEY keytag: 54576 alg: 10 flags: 257)
            |---cz. (DS keytag: 54576 digest type: 2)
                  |---. (DNSKEY keytag: 16665 alg: 8 flags: 256)
                  |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
;; Chase successful
```

Let’s break some signatures

- Edit RRGIG for some particular record
  - edit the **`${fasnick}.db.signed`**
  - change some letter, delete some part, ...
- Reload BIND: **`# systemctl reload named`**
- Retest with **drill** and watch the Chase to fail

Automatic zone signing

with BIND 9

Automatic zone signing

- Adjust the **/etc/named.conf**

```
zone "${fasnick}.test.devconf.cz." IN {
        type master;
        file "dynamic/${fasnick}.db";
        update-policy local;
        auto-dnssec maintain;
        key-directory "dynamic";
};
```

- Change KSK and ZSK keys owner

```
$ chown named /var/named/dynamic/K*.{key,private}
```

- Reload BIND: **# systemctl reload named**

Testing the chain of trust

```
$ drill -S -k /etc/trusted-key.key \
   ${fasnick}.test.devconf.cz.
```

- example output

```
;; Number of trusted keys: 1
;; Chasing: devconf.cz. A

DNSSEC Trust tree:
devconf.cz. (A)
|---devconf.cz. (DNSKEY keytag: 18620 alg: 7 flags: 256)
      |---devconf.cz. (DNSKEY keytag: 4515 alg: 7 flags: 257)
      |---devconf.cz. (DS keytag: 4515 digest type: 1)
      |---cz. (DNSKEY keytag: 12305 alg: 10 flags: 256)
            |---cz. (DNSKEY keytag: 54576 alg: 10 flags: 257)
            |---cz. (DS keytag: 54576 digest type: 2)
                  |---. (DNSKEY keytag: 16665 alg: 8 flags: 256)
                  |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
;; Chase successful
```

Automatic zone signing

with FreeIPA 4.1

FreeIPA: overview

- LDAP, Kerberos, NTP, DNS, X.509 Certificate System, …

  … as an **integrated management solution**
- User interface, APIs etc. for existing implementations: ISC BIND 9
- Automate what can be automated!
- Packages:
  - freeipa-server (Fedora)
  - ipa-server (RHEL)
- `http://www.freeipa.org/page/Demo`

FreeIPA: CLI

- Command line interface for DNSSEC signing

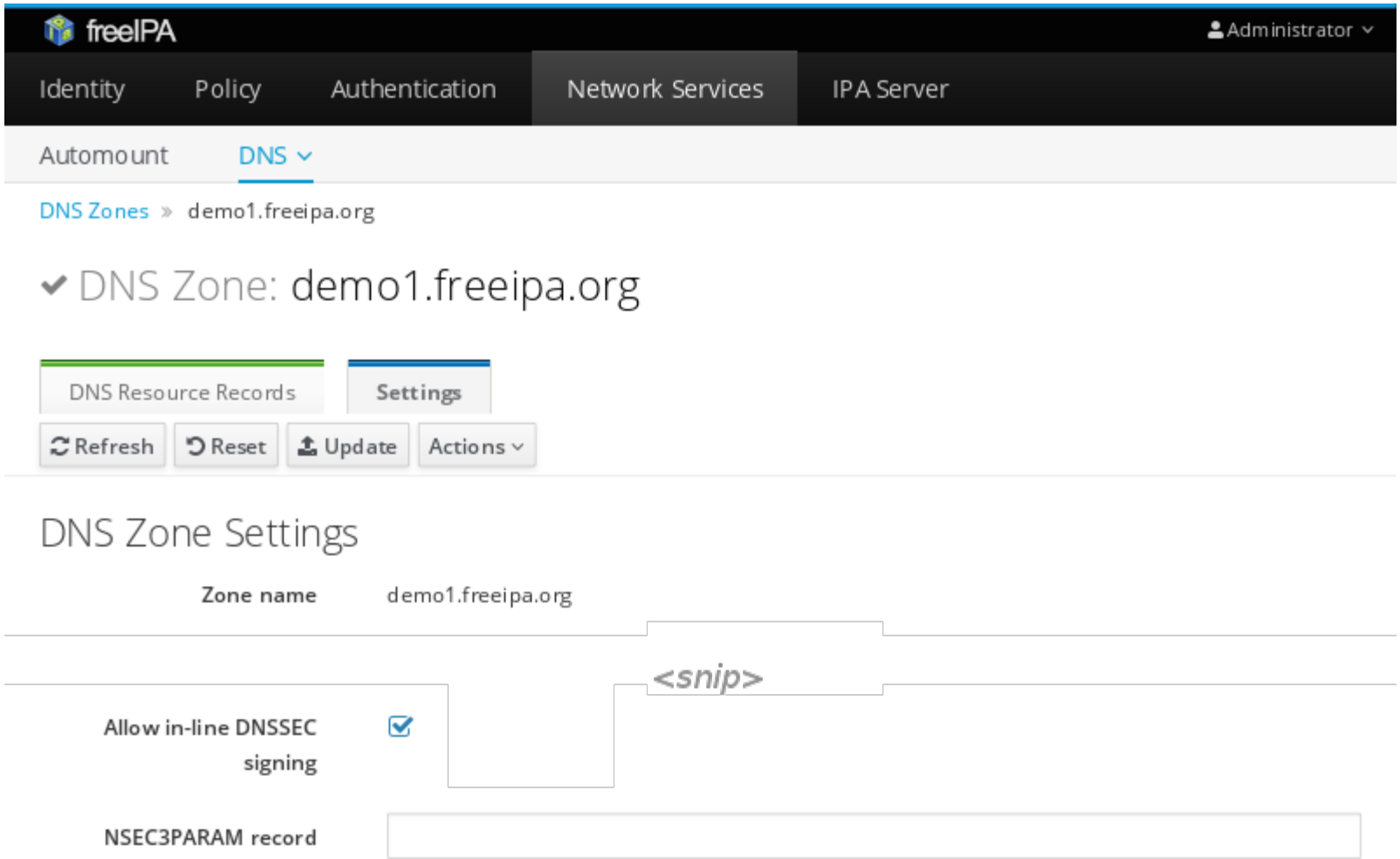
```
$ ipa dnszone-mod example.net. \
    --dnssec=true
```

➔ the zone is signed
➔ keys are automatically rotated

- The only manual step:
  - Upload DS records (hashes of public keys) to parent zone - details depend on DNS registrar

FreeIPA: Web UI

- Web interface for DNSSEC signing



FreeIPA: zone signing how-to

1. Sign the zone:

```
$ ipa dnszone-mod example.net. \
    --dnssec=true
```

2. Upload DS records

```
$ dig @localhost \
    example.net DNSKEY > dnskey
$ dnssec-dsfromkey -f dnskey \
    example.net.
example.net.            IN      DS      35886 8 2
7862B27F5F516EBE19680444D4CE5E762981931842C465F00236
401D8BD973EE
```

DNSSEC on the client

a.k.a. unbound + dnssec-trigger

Testing that you are NOT secured

- Assumption: you are not already running local validating resolver

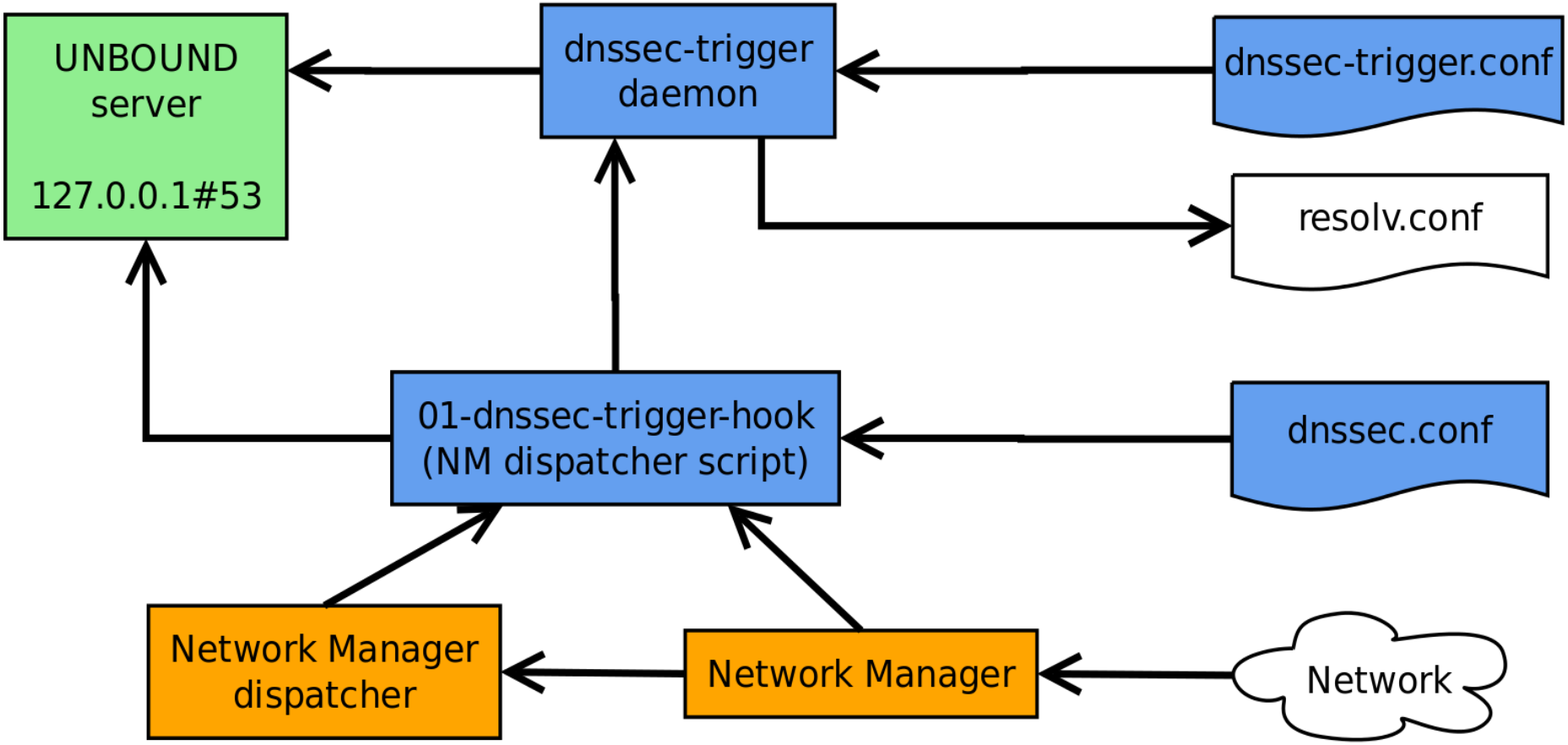
- Try to get deliberately DNSSEC broken sites
  - `$ wget rhybar.cz`
  - `$ wget dnssec-failed.org`

- `Test that chain of trust is broken`
  - `$ drill -S -k /etc/trusted-key.key \ <domain>`

What do you need

- Install packages
  - unbound, dnssec-trigger
- Start and enable the service
  - `# systemctl enable dnssec-triggerd`
  - `# systemctl start dnssec-triggerd`

How it works


UNBOUND
server
127.0.0.1#53
dnssec-trigger
daemon
dnssec-trigger.conf
resolv.conf
01-dnssec-trigger-hook
(NM dispatcher script)
dnssec.conf
Network Manager
dispatcher
Network Manager
Network

Testing that you ARE secured

- Assumption: you are running dnssec-trigger and unbound

- Try to get deliberately DNSSEC broken sites
  - `$ wget rhybar.cz`
  - `$ wget dnssec-failed.org`

- Finally, no one can spoof signed DNS records

Contacts

Petr Špaček pspacek@redhat.com

Tomáš Hozza thozza@redhat.com

Feedback URL

http://devconf.cz/f/140

Other details

- Validators can be locally configured with arbitrary trust anchors
  - E.g. `my.example.net` can be signed **only** by a key you hardcoded into client configuration because you do not trust `example.net` or anybody else!
- RFC 5011 defines trust anchor auto-update
  - key revocation and roll-over
  - resilient up to N-1 key compromises

```
$ cat /etc/named.root.key
managed-keys {
   . initial-key 257 3 8 "VLOyQbSEW0O8gcCjF...";
};
```