Hands-on DNSSEC with DNSViz

Casey Deccio, Verisign Labs

NANOG 66, San Diego

Feb 8, 2016

Preparation

- Demo and exercises available at:
  - http://dnsviz.net/demo/
- Includes links to the following:
  - VirtualBox software
  - VirtualBox demo image
  - Tutorial exercises

Objectives

- Understand the basics of DNS and DNSSEC
- Become familiar with DNS server and analysis tools
  - DiG
  - BIND
  - DNSViz
- Learn how tools might be used to routinely analyze/ monitor your DNS health

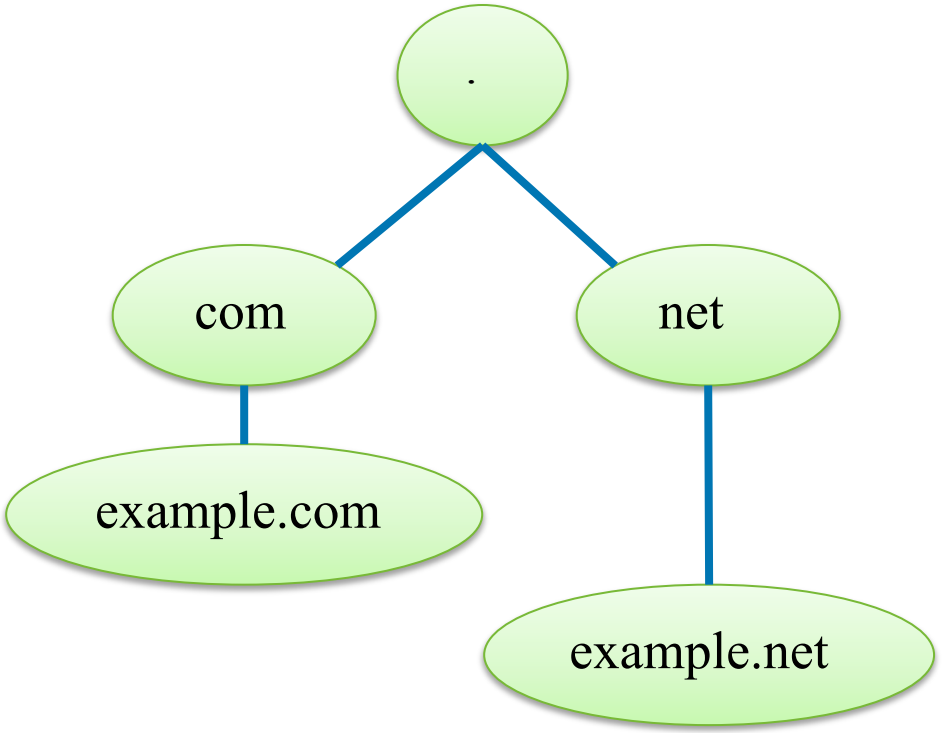
Caveats

- The exercises range from novice-level to advanced.
- Many of the exercises are more to facilitate understanding than efficiency.
- The exercises are be meant for learning DNS/DNSSEC and related tools, but do not cover all details for proper DNS/DNSSEC maintenance.

DNS Overview

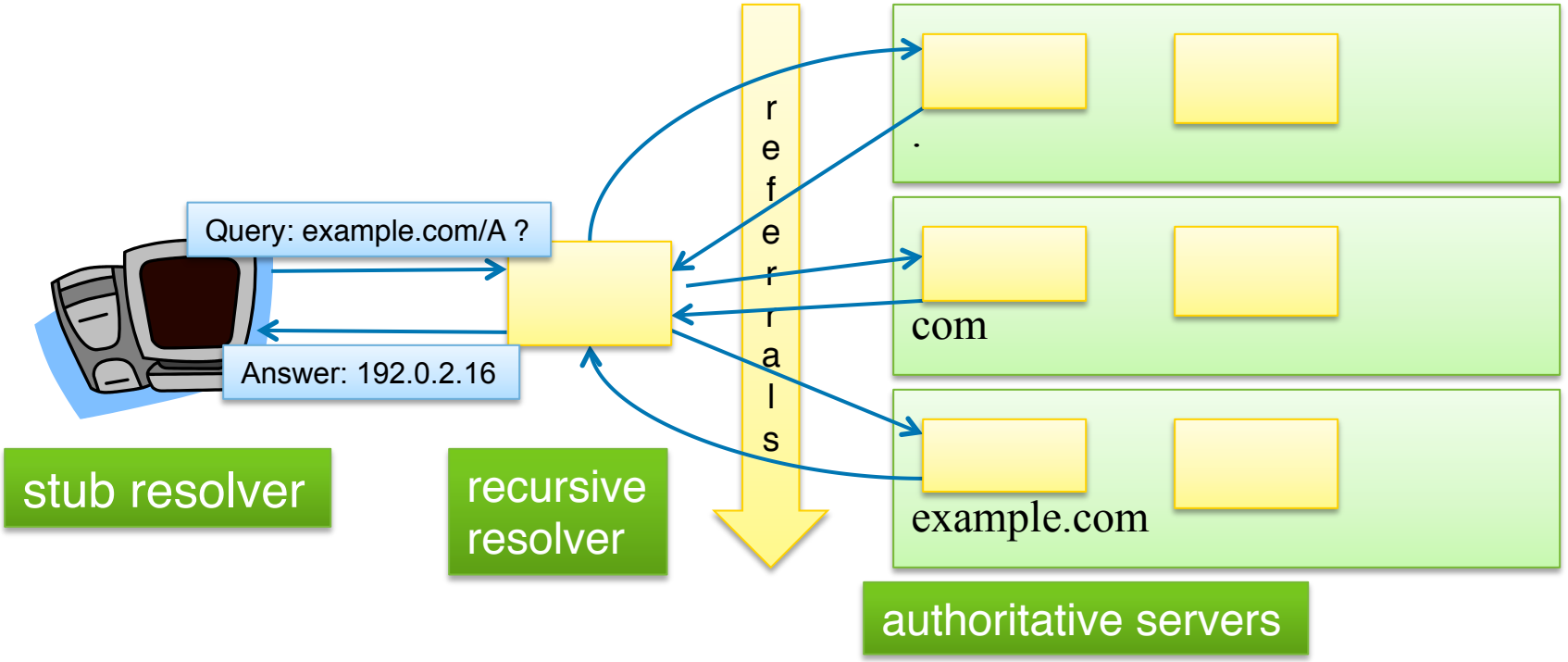
DNS Namespace

- Namespace is organized hierarchically
- DNS **root** is top of namespace
- **Zones** are autonomously managed pieces of DNS namespace
- Subdomain namespace is delegated to child zones



DNS Name Resolution

- ***Resolvers*** query ***authoritative servers***
- Queries begin at root zone, resolvers follow downward referrals
- Resolver stops when it receives authoritative answer



Virtual Environment Initialization

- Unzip dnsviz-demo-v2.zip
- Open dnsviz-demo-v2/dnsviz-demo-v2.vbox

- “Start” VM
- Enlarge screen
- Double-click “Tutorial Exercises” file

- (Exercises 0.1 – 0.2)
  - Open “Terminal Emulator”
  - Change to “demo” directory

Query DNS Servers (1.1 – 1.5)

```
$ dig @a.root-servers.net example.com
```

query a specific server
(rather than querying your
configured resolver)

no record type specified,
so default type
“A” (address) is used

```
$ dig @a.gtld-servers.net example.com
```

```
$ dig @a.iana-servers.net example.com
```

```
$ dig example.com
```

no server is explicitly
designated, so query
goes to local resolver

```
$ dig @a.iana-servers.net foobar.example.com
```

Query a root Server

Terminal - dnsviz@dnsviz-demo: ~/demo

File Edit View Terminal Tabs Help

```
dnsviz@dnsviz-demo:~/demo$ dig @a.root-servers.net example.com

; <<>> DiG 9.9.5-9-Debian <<>> @a.root-servers.net example.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1649
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 16
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;example.com.                   IN      A

;; AUTHORITY SECTION:
com.                    172800  IN      NS      m.gtld-servers.net.
com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      i.gtld-servers.net.
com.                    172800  IN      NS      h.gtld-servers.net.
com.                    172800  IN      NS      g.gtld-servers.net.
com.                    172800  IN      NS      f.gtld-servers.net.
com.                    172800  IN      NS      e.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      a.gtld-servers.net.

;; ADDITIONAL SECTION:
m.gtld-servers.net.     172800  IN      A       192.55.83.30
l.gtld-servers.net.     172800  IN      A       192.41.162.30
```

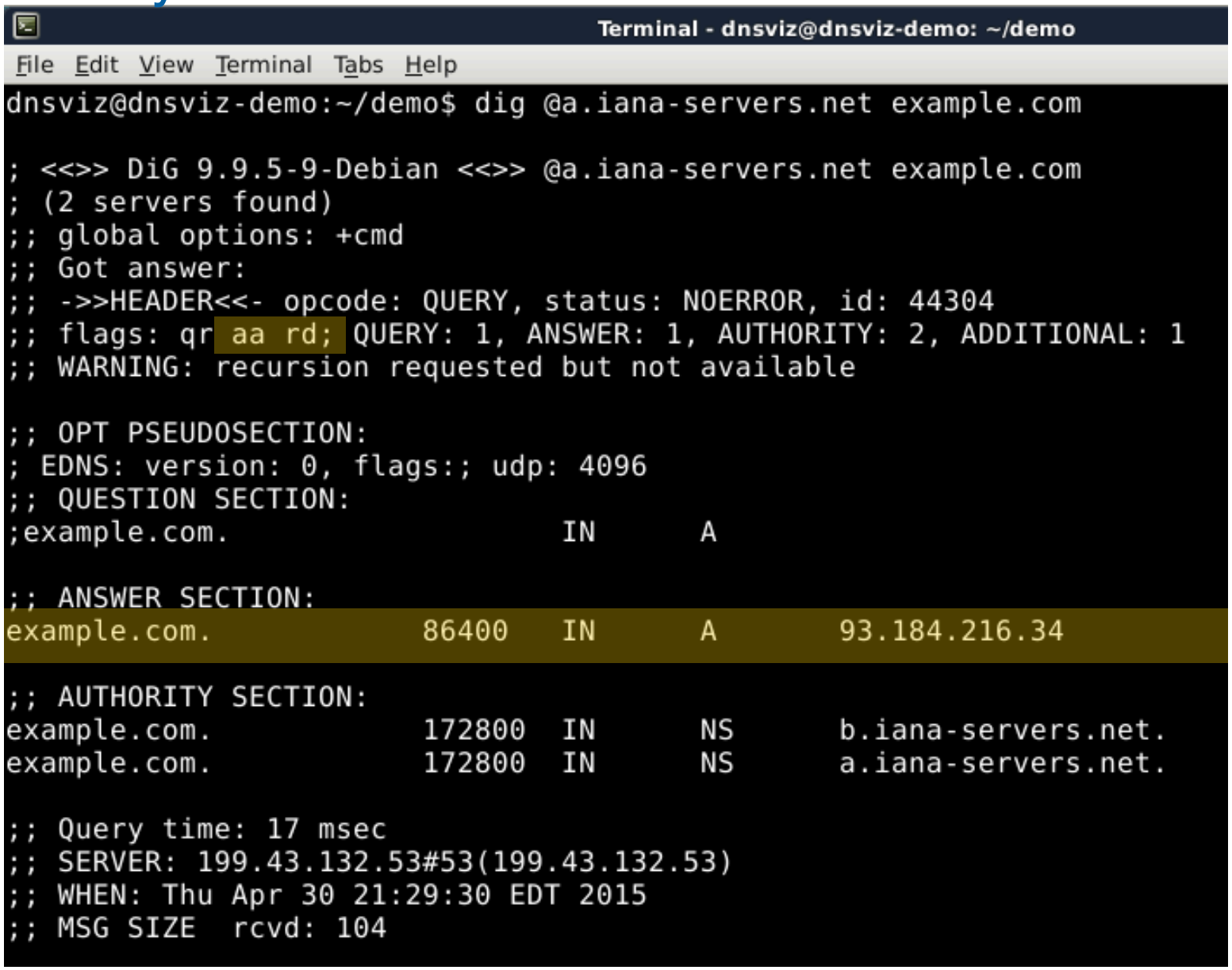
Query a TLD Server

Terminal - dnsviz@dnsviz-demo: ~/demo

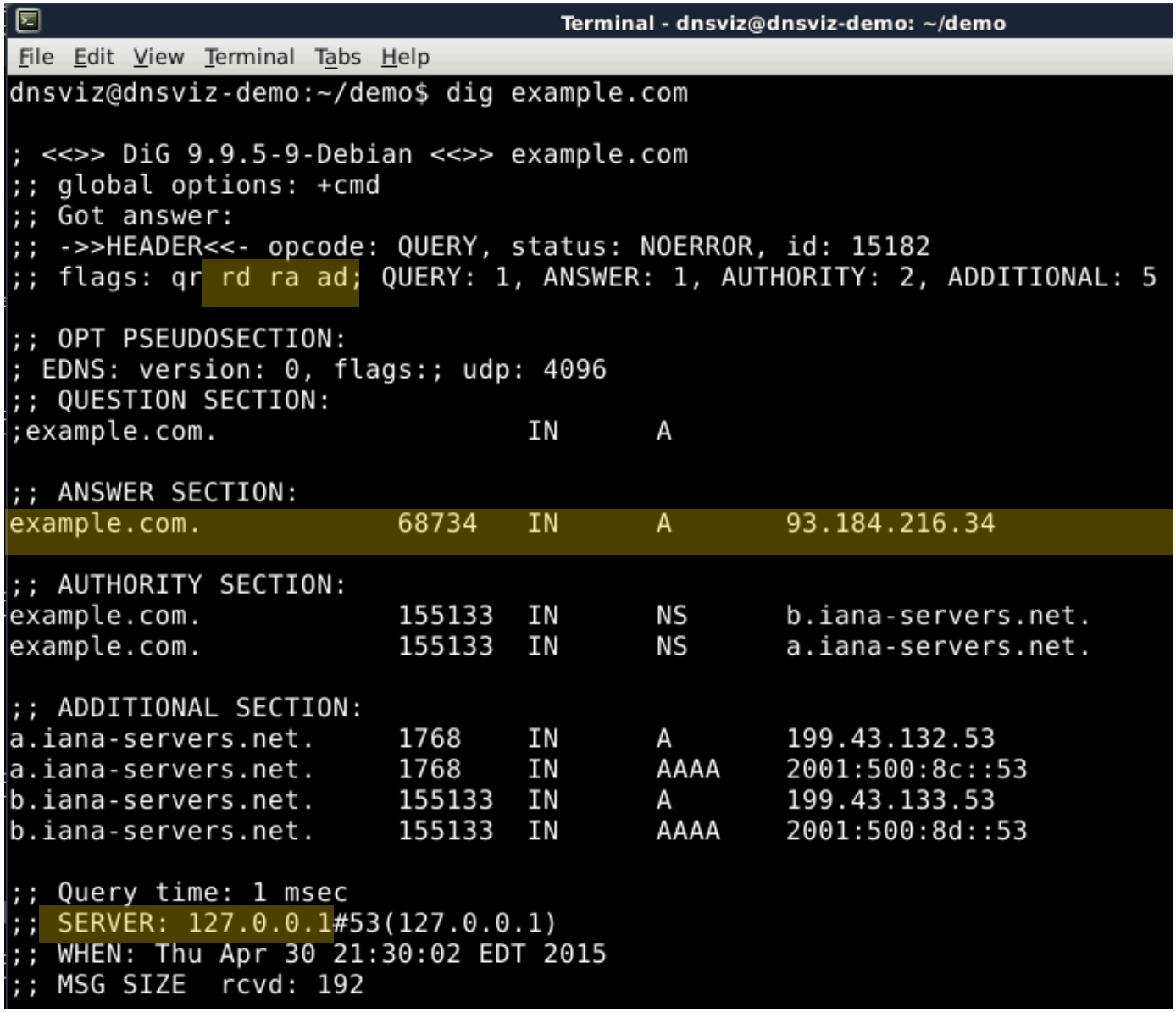
File Edit View Terminal Tabs Help

```
dnsviz@dnsviz-demo:~/demo$ dig @a.gtld-servers.net example.com

; <<>> DiG 9.9.5-9-Debian <<>> @a.gtld-servers.net example.com
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64763
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 5
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;example.com.                   IN      A

;; AUTHORITY SECTION:
example.com.            172800  IN      NS      a.iana-servers.net.
example.com.            172800  IN      NS      b.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     172800  IN      A       199.43.132.53
a.iana-servers.net.     172800  IN      AAAA    2001:500:8c::53
b.iana-servers.net.     172800  IN      A       199.43.133.53
b.iana-servers.net.     172800  IN      AAAA    2001:500:8d::53

;; Query time: 91 msec
;; SERVER: 192.5.6.30#53(192.5.6.30)
;; WHEN: Thu Apr 30 21:27:16 EDT 2015
;; MSG SIZE  rcvd: 176
```

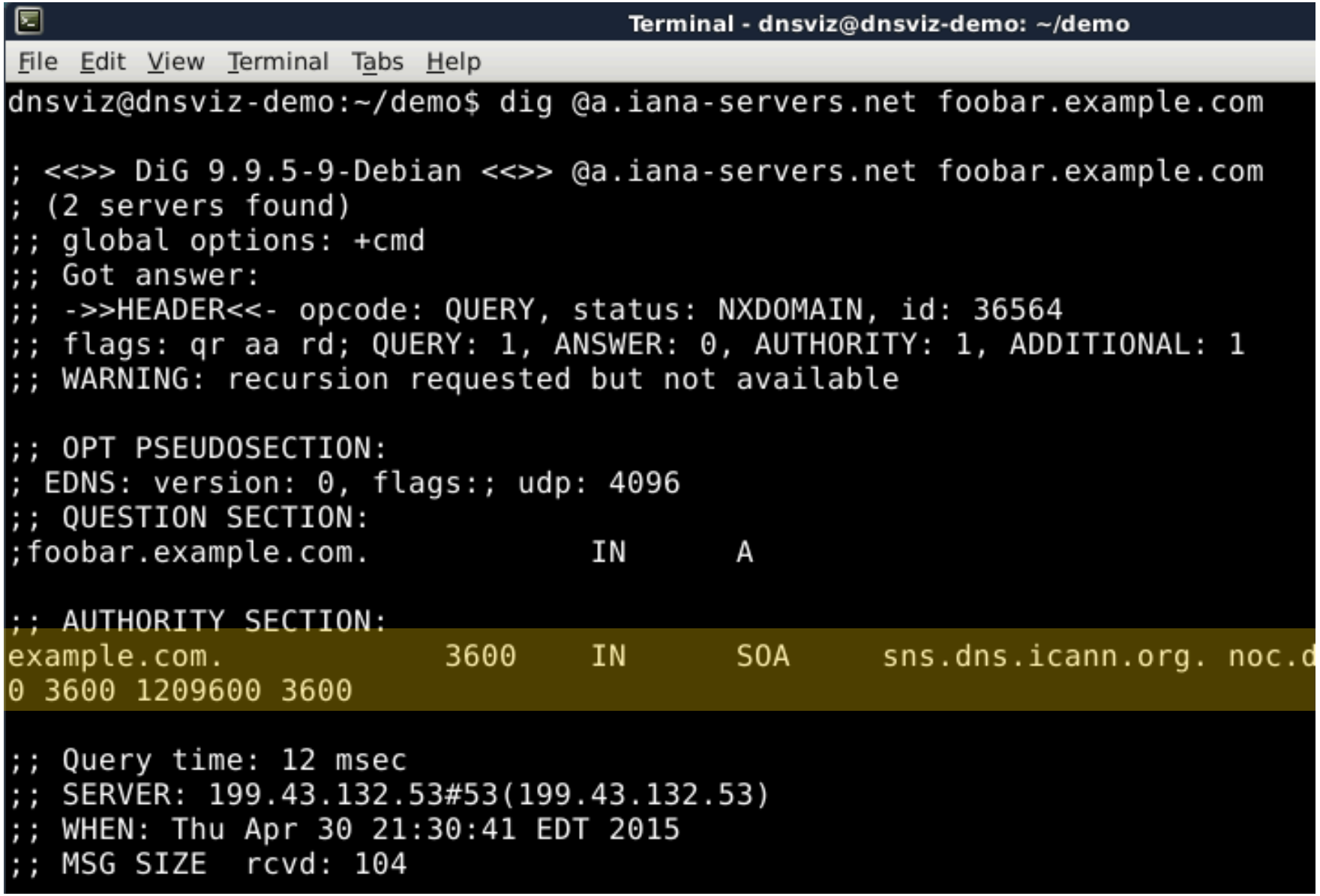
Query an SLD Server



Query Local Recursive Resolver



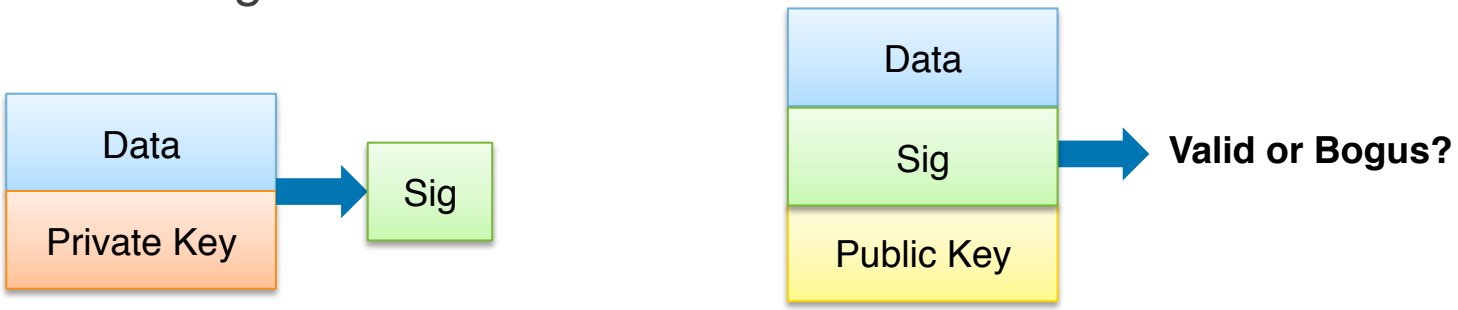
Query for a Non-existent Name



DNSSEC Overview

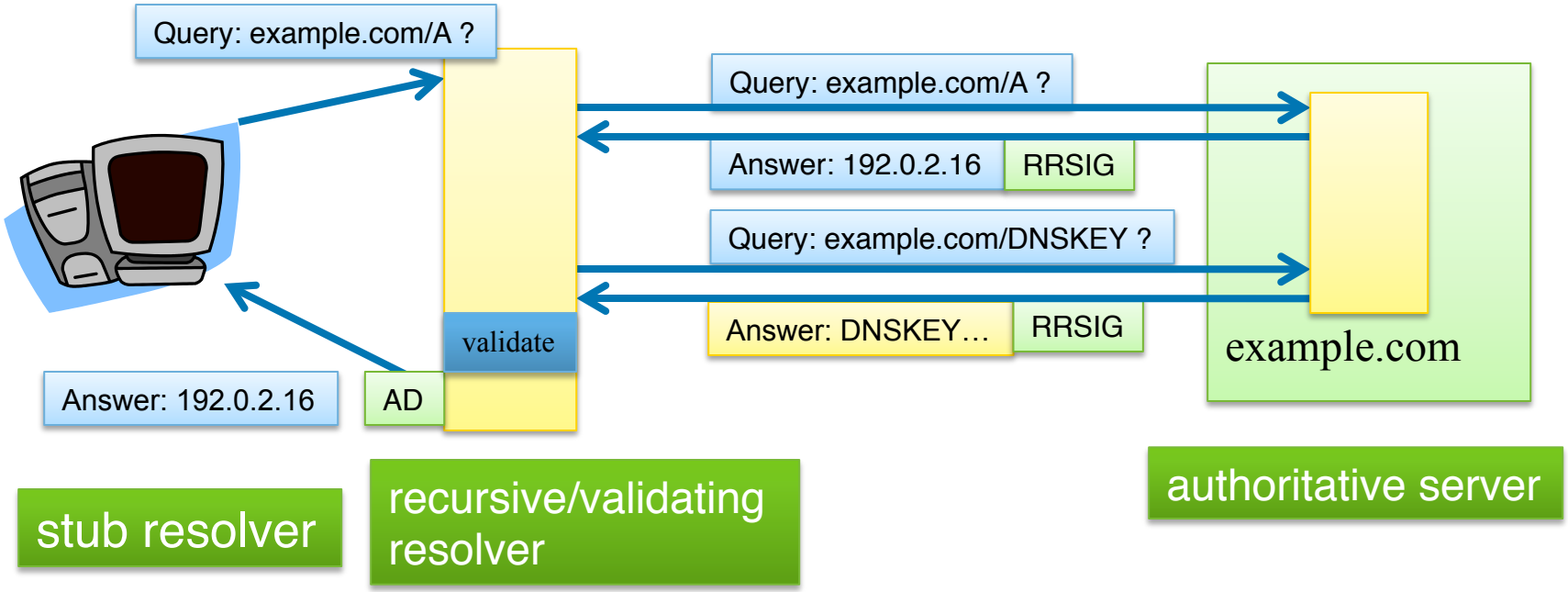
Public Key Cryptography

- Keys
  - **Public** Key – advertised to everyone
  - **Private** Key – kept hidden
- Signatures
  - Made by private key
  - Validated with public key
- Validation
  - Consumer uses public key, message, and signature to validate message



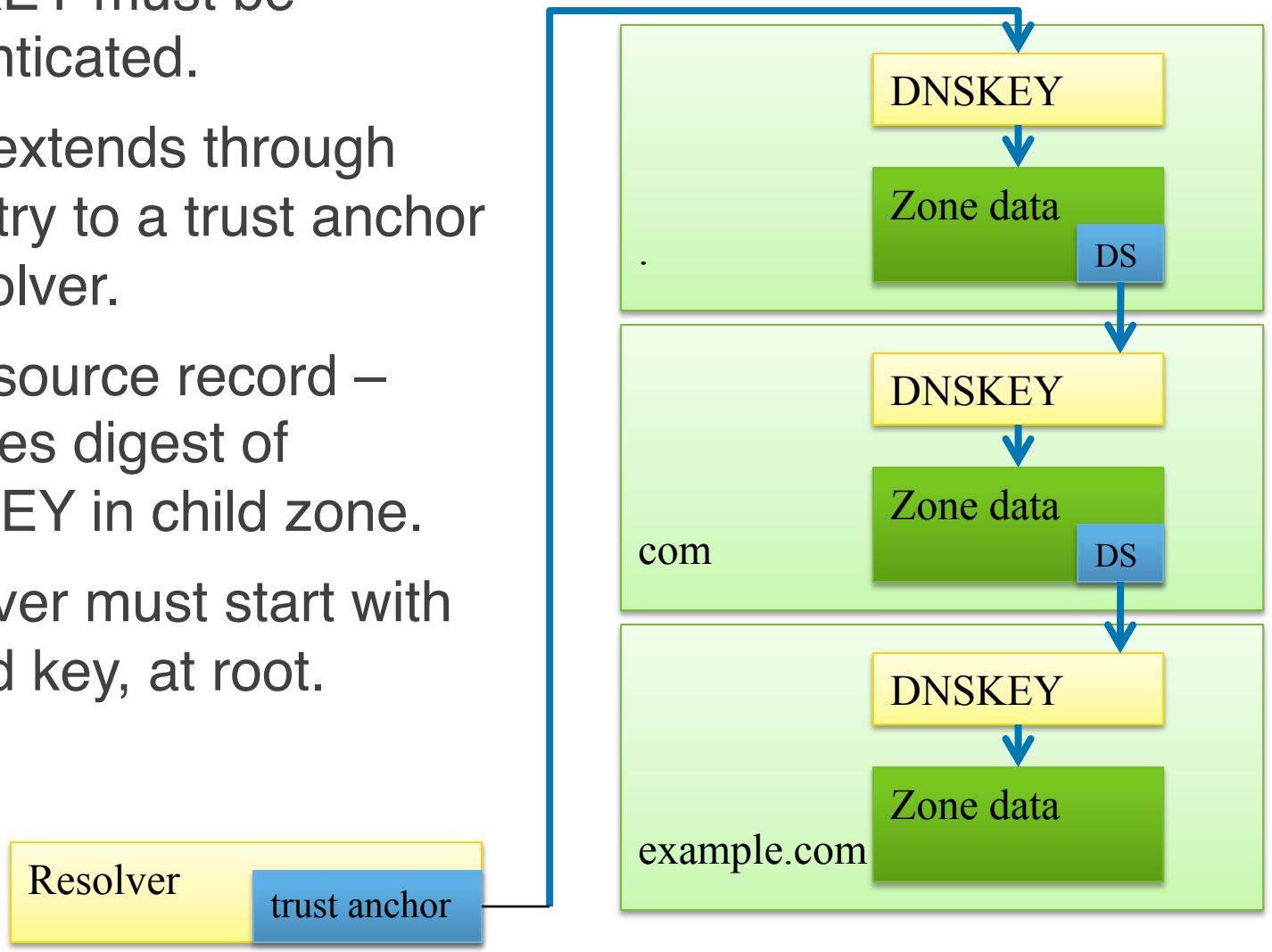
DNS Security Extensions (DNSSEC)

- DNS data signed with private keys
- Signatures (RRSIGs) and public keys (DNSKEYs) published in zone data
- Resolver response
  - If authentic: Authenticated data (AD) bit is set
  - If bogus: SERVFAIL message is returned



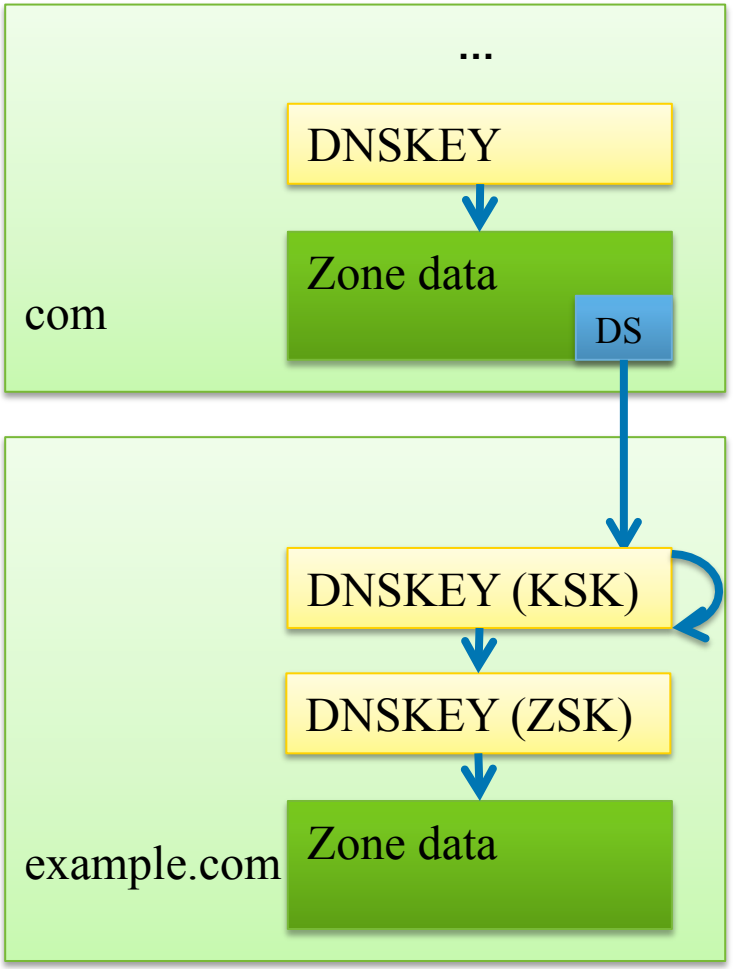
DNSSEC Chain of Trust

- DNSKEY must be authenticated.
- Trust extends through ancestry to a trust anchor at resolver.
- DS resource record – provides digest of DNSKEY in child zone.
- Resolver must start with trusted key, at root.



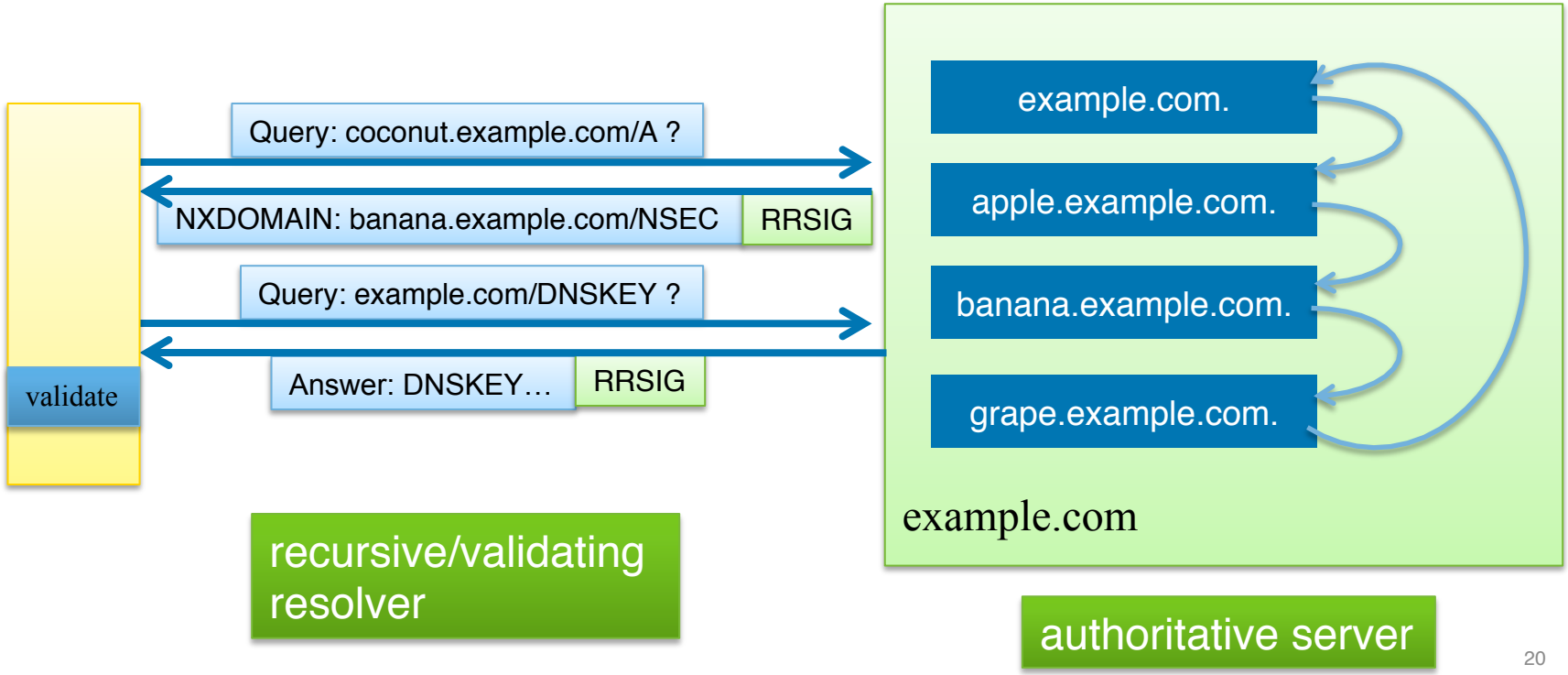
Key Roles – KSK/ZSK

- DNSKEY RRset usually has multiple keys, often with split roles.
- KSK (Key signing key)
  - Signs (only) the DNSKEY RRset.
  - Corresponds to DS records in parent, providing “secure entry point” into zone.
- ZSK (Zone signing key)
  - Signs the rest of the zone.



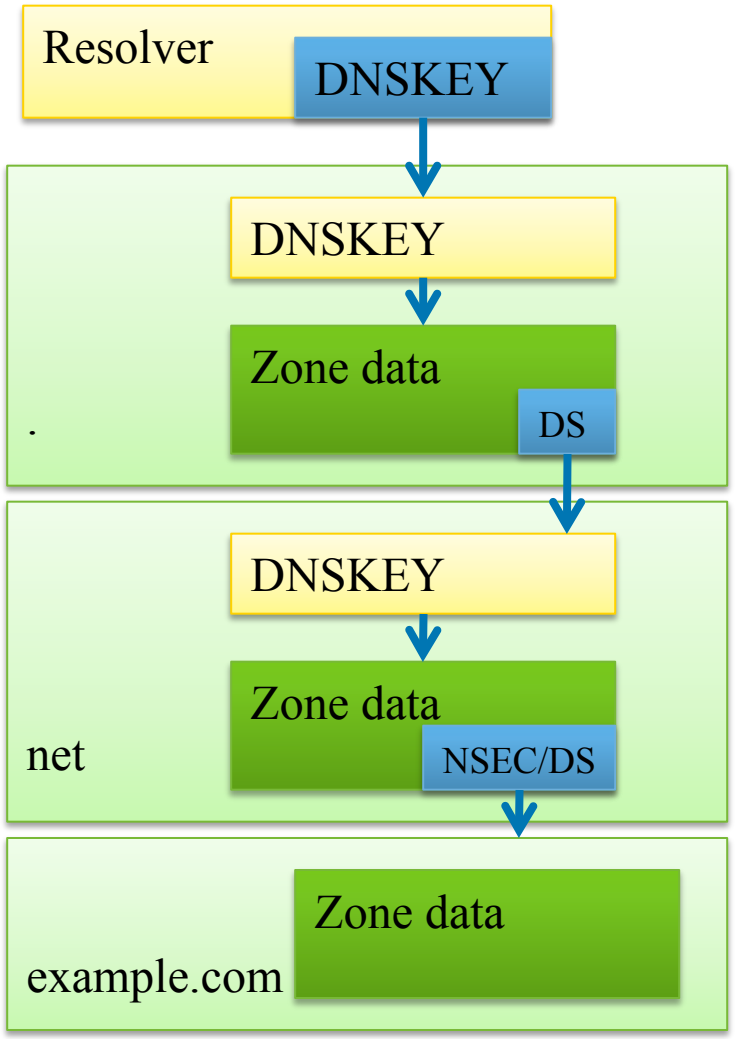
Authenticated Denial of Existence

- How do you prove something doesn't exist?
- "Chain" of names of zone formed using NSEC records.
- NSEC records form comprehensive chain of names (and their record types) in zone in canonical ordering.
- Server uses NSEC records to prove non-existence.



Insecure delegations

- How can DNSSEC be deployed incrementally?
- If child zone is unsigned, resolver must be able to prove it is insecure.
- NSEC resource records provide proof of absence of DS.



Zone Enumeration and NSEC3

- NSEC records allow enumeration of entire zone contents.
- NSEC3 standard introduces *hashed* denial of existence.
  - Joint effort between Verisign, Nominet (.uk), and DENIC (.de).
- Chain is of *hashes* of names, not *names* themselves.

(a hash is the output of a one-way cryptographic function.)

Query for DNSSEC Records (2.1 – 2.5)

```
$ dig +dnssec +multi @a.iana-servers.net example.com
```

include DNSSEC records in response (e.g., RRSIG)

present response in multi-line format with comments (for readability)

query for records of type "DNSKEY" (DNSSEC public key) instead of the default, "A" (address)

```
$ dig +dnssec +multi @a.iana-servers.net example.com DNSKEY
```

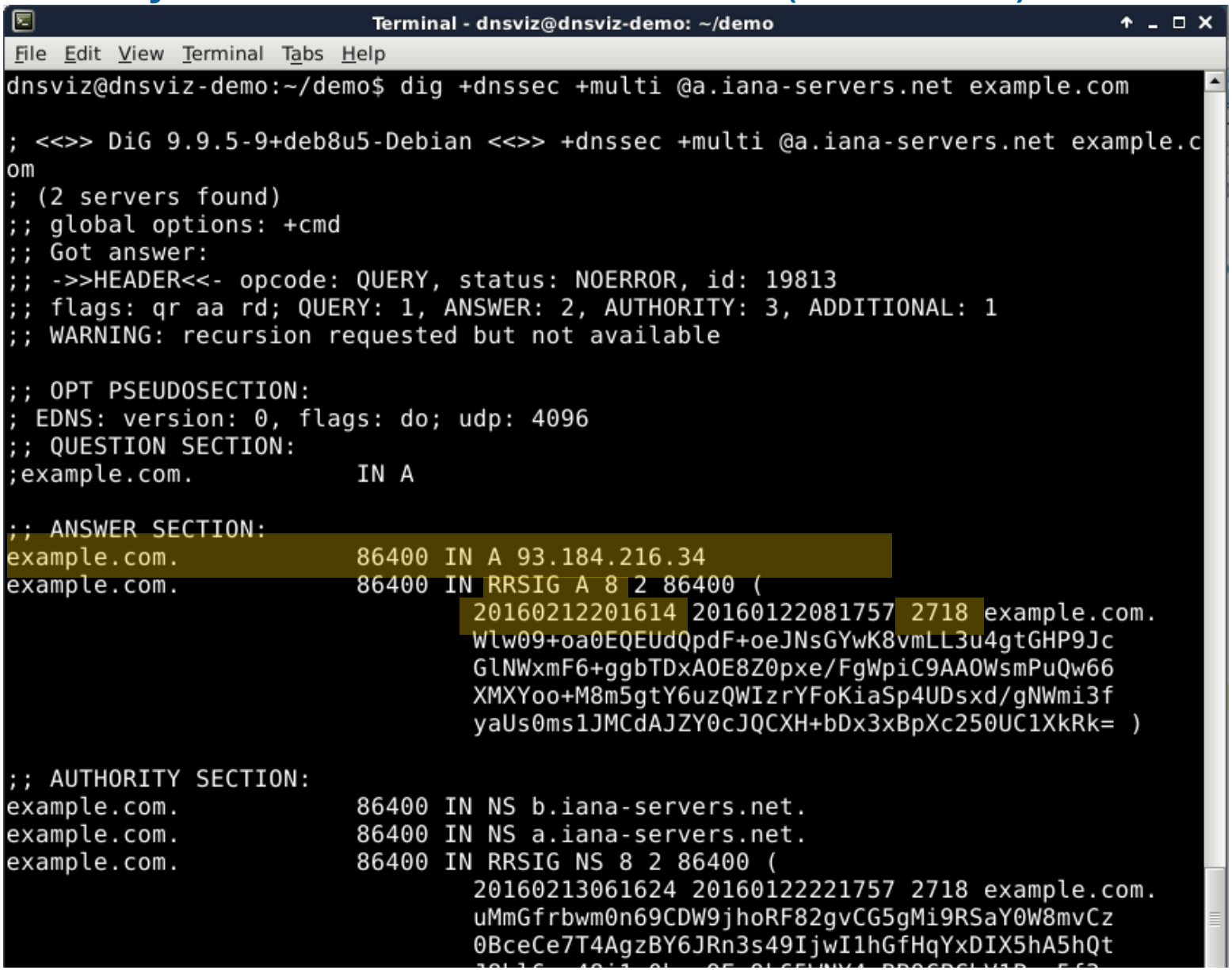
```
$ dig +dnssec +multi @a.gtld-servers.net example.com DS
```

query a "parent" server because we're seeking a DS record

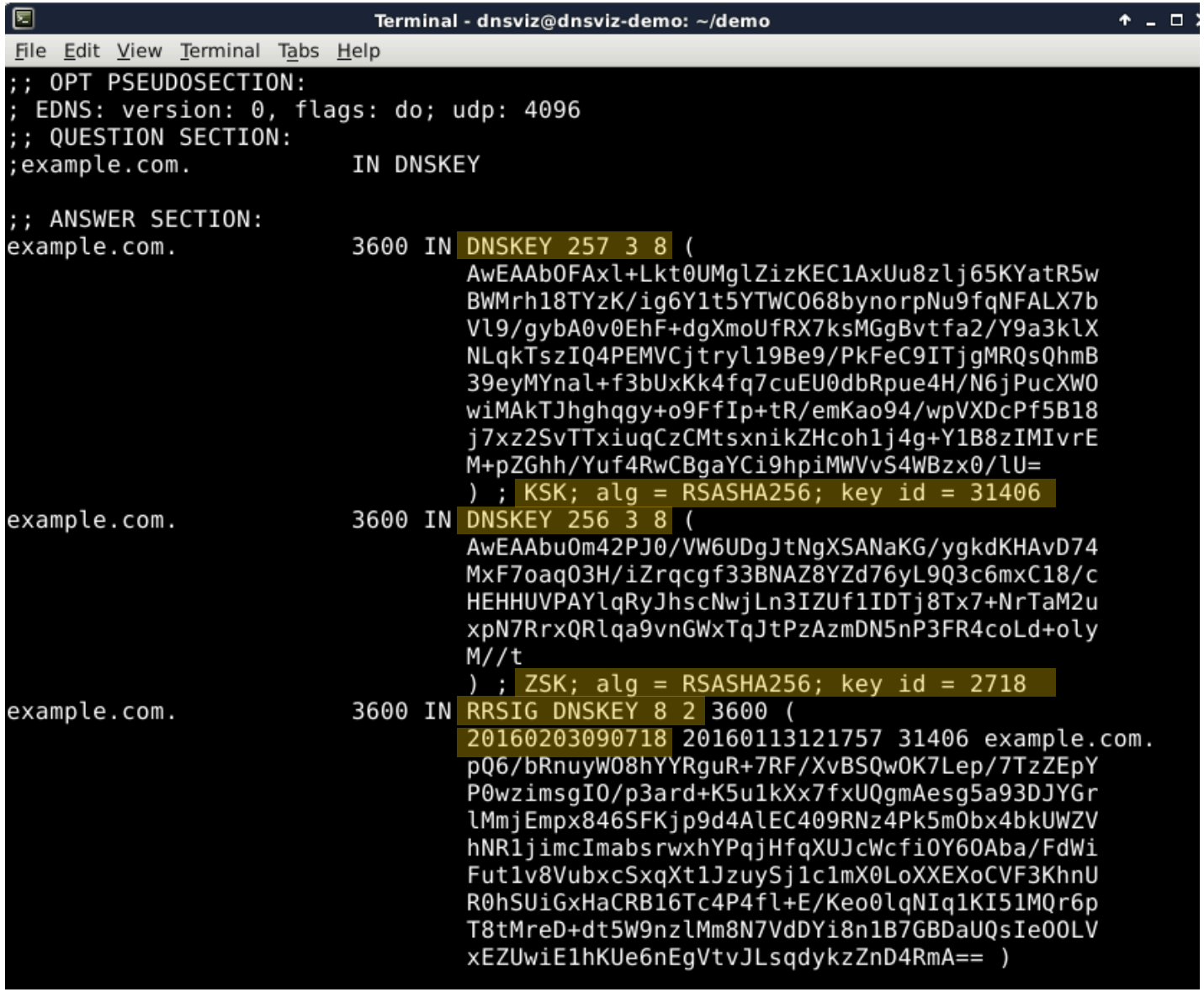
```
$ dig +dnssec +multi example.com
```

```
$ dig +dnssec +multi @a.iana-servers.net foobar.example.com
```

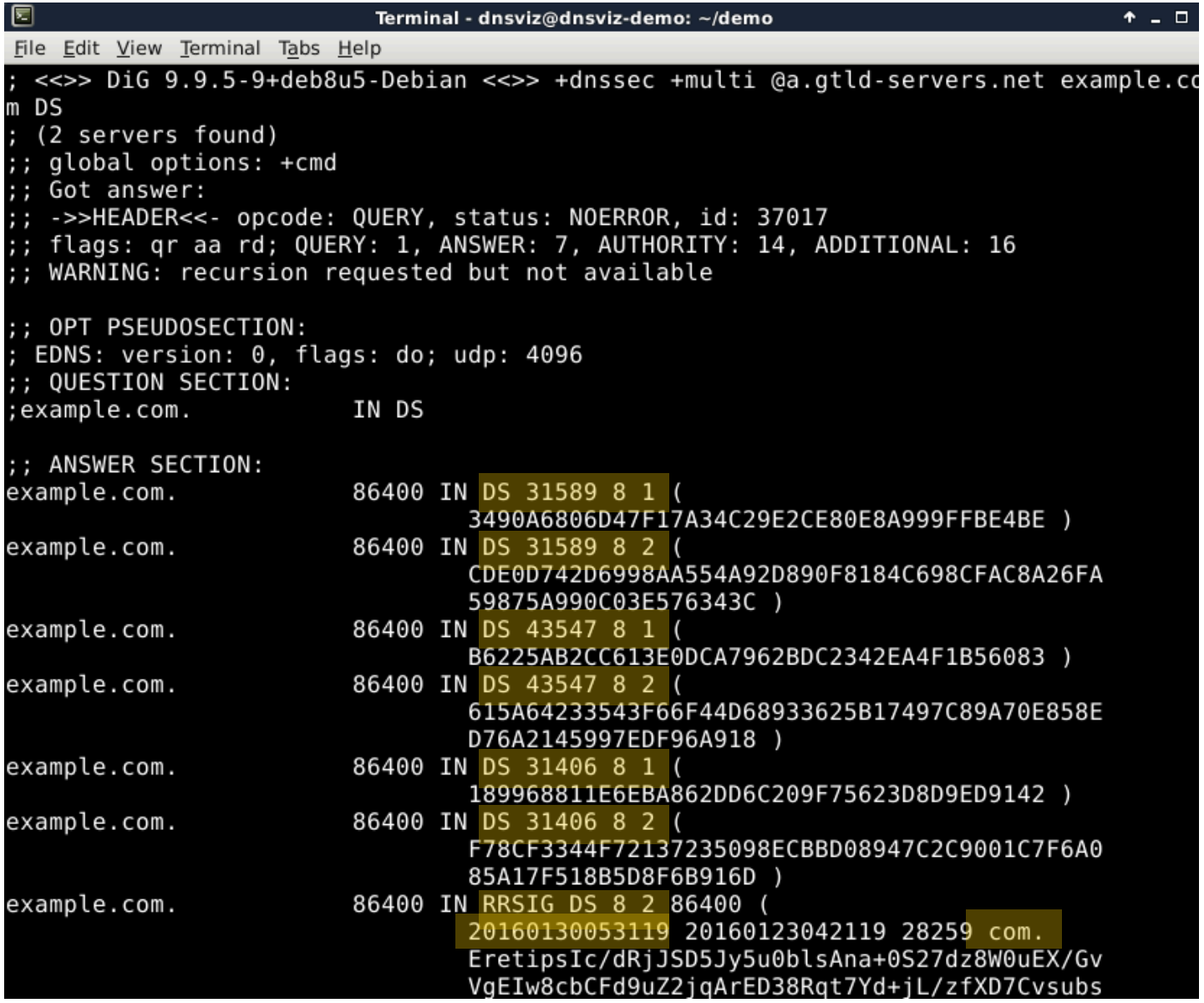
Query for DNSSEC Records (RRSIGs)



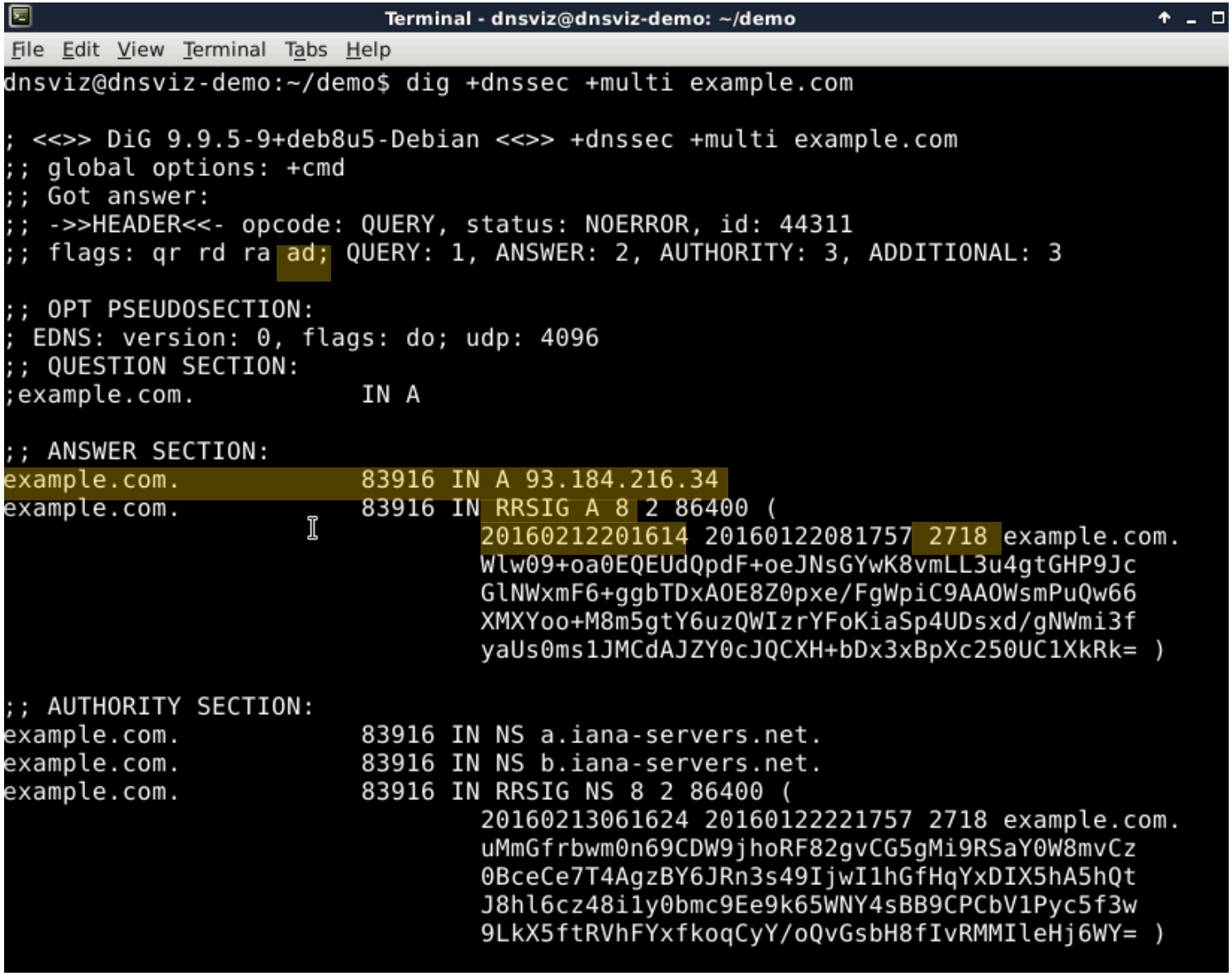
Query for DNSSEC Records (DNSKEY)



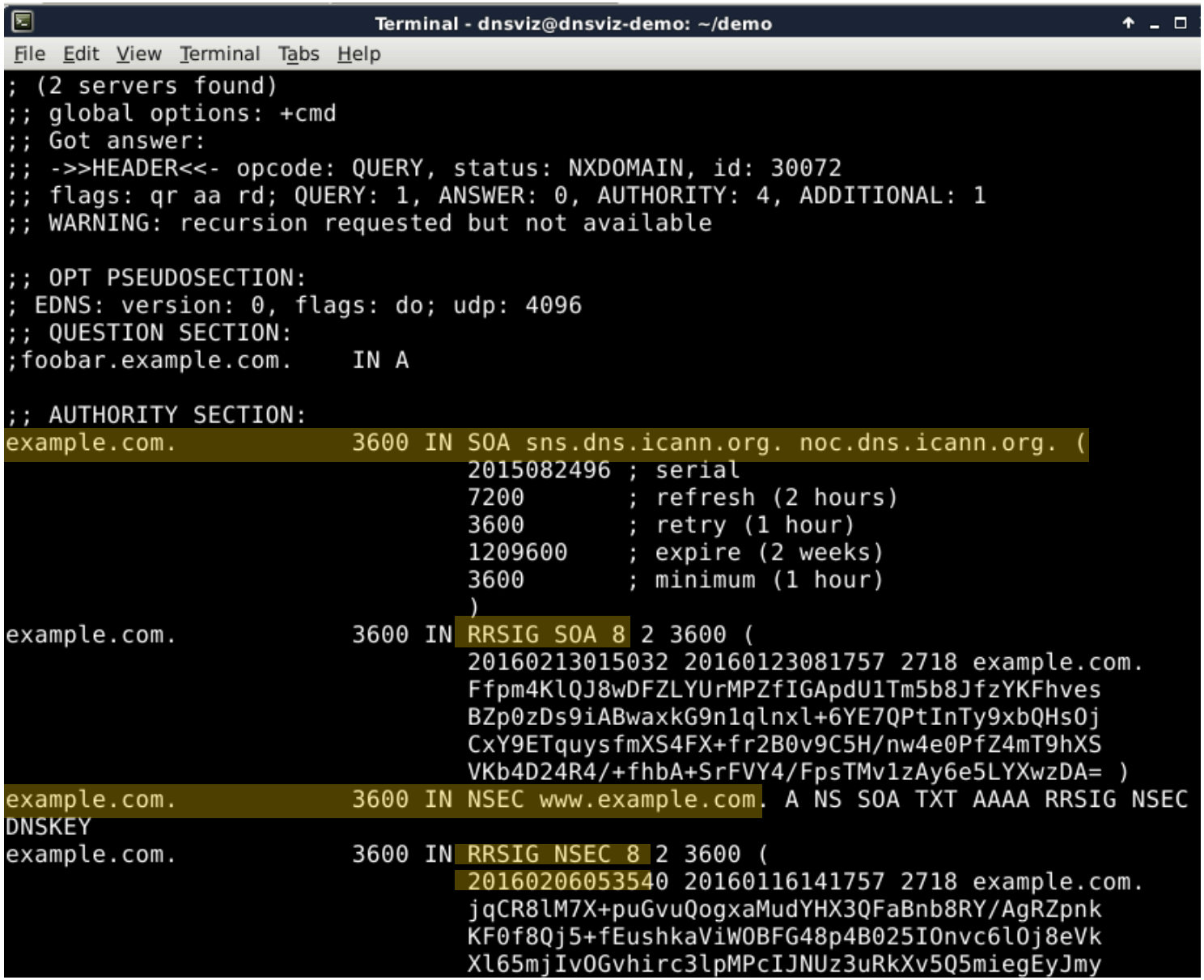
Query for DNSSEC Records (DS)



Query for DNSSEC Records



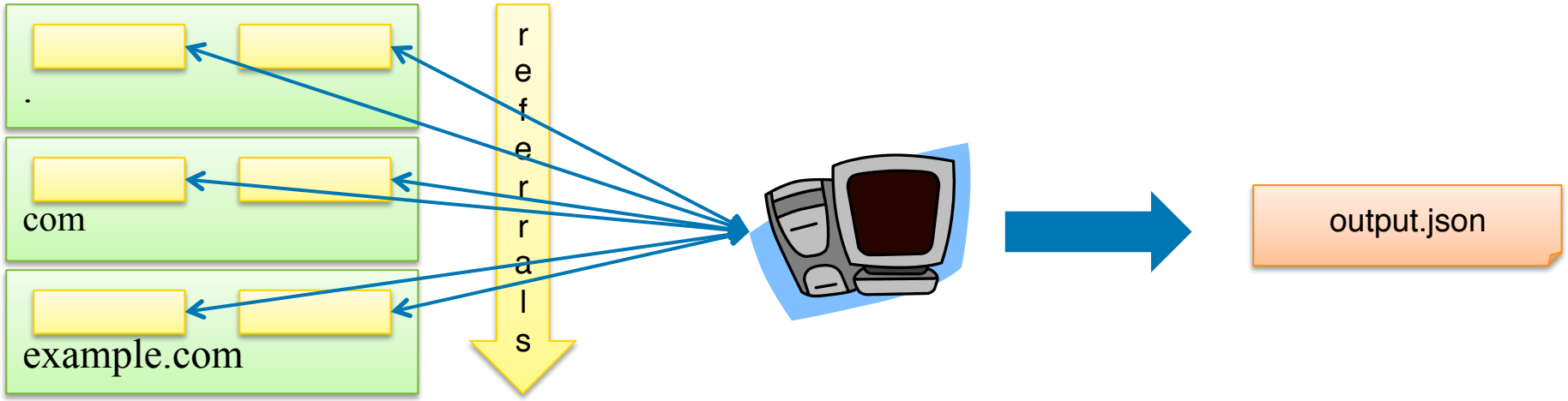
Query For DNSSEC Records (NSEC)



DNSViz

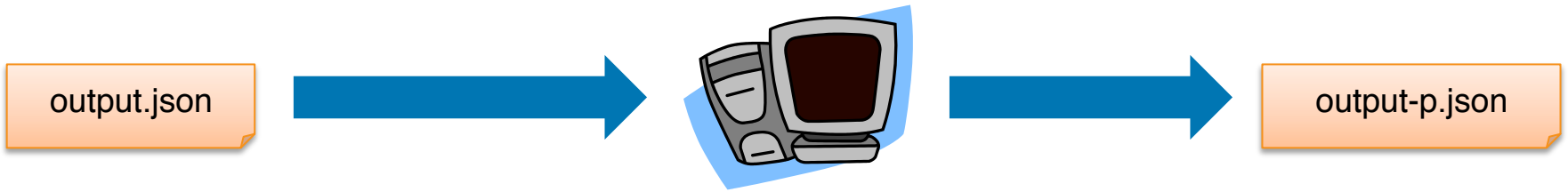
DNS Analysis Using DNSViz (`dnsviz probe` command line)

- Queries issued
  - Referral queries – to learn delegation NS records from parent
  - NS queries – to learn authoritative NS records
  - DNSKEY/DS queries – for building a DNSSEC chain
  - A/AAAA/TXT/MX/SOA queries
  - Diagnostic queries (special handling of errors, etc.)
- All servers queried
  - IPv4/IPv6
  - UDP/TCP



DNS Analysis Using DNSViz (`dnsviz grok` command line)

- Responses analyzed (offline)
  - Responsiveness
    - Query timeouts
    - Network errors
    - EDNS/fragmentation capabilities
  - Consistency
    - Across servers
    - Between DNSKEY/RRSIG
    - Between DNSKEY/DS
- Correctness
  - RRSIG
    - Expiration/inception dates
    - Cryptographic signature
  - DS
    - Cryptographic hash
  - Negative responses
    - NSEC proof correctness
    - SOA record correctness

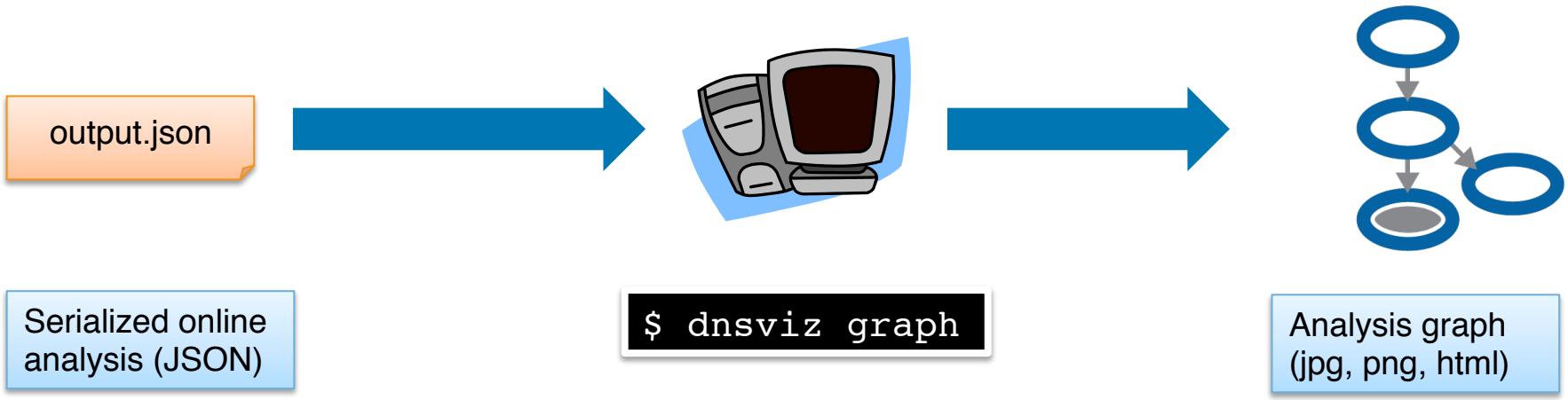


Serialized online analysis (JSON)

Serialized offline analysis (JSON)

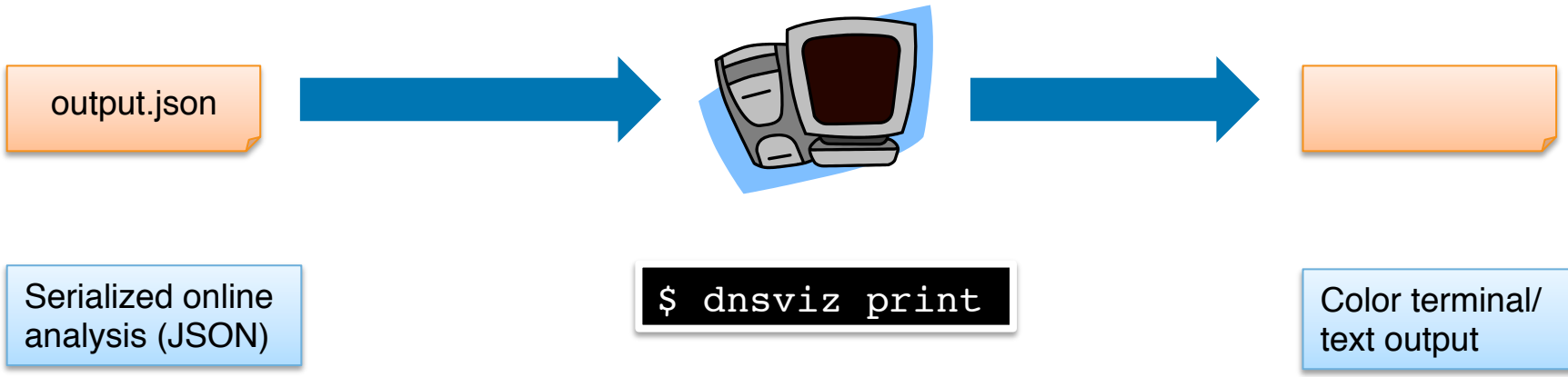
DNS Analysis Using DNSViz (`dnsviz graph` command line)

- Responses analyzed (offline)
  - Responsiveness
    - Query timeouts
    - Network errors
    - EDNS/fragmentation capabilities
  - Consistency
    - Across servers
    - Between DNSKEY/RRSIG
    - Between DNSKEY/DS
- Correctness
  - RRSIG
    - Expiration/inception dates
    - Cryptographic signature
  - DS
    - Cryptographic hash
  - Negative responses
    - NSEC proof correctness
    - SOA record correctness



DNS Analysis Using DNSViz (`dnsviz print` command line)

- Responses analyzed (offline)
  - Responsiveness
    - Query timeouts
    - Network errors
    - EDNS/fragmentation capabilities
  - Consistency
    - Across servers
    - Between DNSKEY/RRSIG
    - Between DNSKEY/DS
- Correctness
  - RRSIG
    - Expiration/inception dates
    - Cryptographic signature
  - DS
    - Cryptographic hash
  - Negative responses
    - NSEC proof correctness
    - SOA record correctness



Analyze Using `dnsviz probe` (3.1 – 3.2)

Issue diagnostic queries to authoritative servers, rather than recursive servers

```
$ dnsviz probe -A -a . -p example.com > example.com.json
```

follow referrals from root (".") to analyze name

make the output "pretty" (for readability)

store analysis in file called "example.com.json"

```
$ medit example.com.json &
```

Analyze Using `dnsviz grok` (3.3 – 3.4)

make the output
“pretty” (for readability)

read analysis from
“example.com.json”

```
$ dnsviz grok -p < example.com.json > example.com-p.json
```

store analysis in file called
“example.com-p.json”

```
$ medit example.com-p.json
```

Analyze Using `dnsviz grok` (3.5 – 3.6)

show only information that is of priority “info” or higher

```
$ dnsviz grok -l info -p < example.com.json \
                  > example.com-p1.json
```

```
$ medit example.com-p1.json
```

Analyze Using `dnsviz grok` (3.7)

show only information that is of priority “error” or higher

display output (if any) to screen, instead of redirecting to file

```
$ dnsviz grok -l error -p < example.com.json
```

Analyze Using `dnsviz graph` (3.8 – 3.11)

output interactive HTML format

Don't use any trust anchor

```
$ dnsviz graph -Thtml -t /dev/null < example.com.json \
                      > example.com.html
```

```
$ firefox example.com.html &
```

```
$ dnsviz graph -Thtml -t tk.txt < example.com.json \
                      > example.com.html
```

anchor trust with root KSK

```
$ firefox example.com.html &
```

Analyze Using `dnsviz print` (3.12 – 3.13)

Don't use any trust anchor

```
$ dnsviz print -t /dev/null < example.com.json
```

```
$ dnsviz print -t tk.txt < example.com.json
```

anchor trust with root KSK

View `dnsviz probe` Output

example.com.json


```
{
    ".": {
        "type": "authoritative",
        "stub": false,
        "analysis_start": "2016-01-26 14:54:55 UTC",
        "analysis_end": "2016-01-26 14:54:58 UTC",
        "clients_ipv4": [
            "10.0.2.15"
        ],
        "clients_ipv6": [],
        "referral_rdtype": "NS",
        "explicit_delegation": false,
        "auth_ns_ip_mapping": {
            "a.root-servers.net.": [
                "198.41.0.4",
                "2001:503:ba3e::2:30"
            ],
            "b.root-servers.net.": [
                "192.228.79.201",
                "2001:500:84::b"
            ],
            "c.root-servers.net.": [
                "192.33.4.12",
                "2001:500:2::c"
            ],
            "d.root-servers.net.": [
                "199.7.91.13",
                "2001:500:2d::d"
            ],
            "e.root-servers.net.": [
                "192.203.230.10"
            ],
            "f.root-servers.net.": [
                "192.5.5.241",
                "2001:500:2f::f"
```

View `dnsviz probe` Output

example.com.json


```
            ]
        },
        "queries": [
            {
                "qname": ".",
                "qclass": "IN",
                "qtype": "NS",
                "options": {
                    "flags": 0,
                    "edns_version": 0,
                    "edns_max_udp_payload": 4096,
                    "edns_flags": 32768,
                    "edns_options": [],
                    "tcp": false
                },
                "responses": {
                    "192.5.5.241": {
                        "10.0.2.15": {
                            "message": "DPyEAAABAA4AAAAZAAACAAEAAAIA
                            "msg_size": 913,
                            "time_elapsed": 86,
                            "history": []
                        }
                    },
                    "192.33.4.12": {
                        "10.0.2.15": {
                            "message": "qw6EAAABAA4AAAAZAAACAAEAAAIA
                            "msg_size": 913,
                            "time_elapsed": 32,
                            "history": []
                        }
                    },
                    "192.36.148.17": {
                        "10.0.2.15": {
                            "message": "KIOEAAABAA4AAAAZAAACAAEAAAIA
```

View `dnsviz probe` Output

example.com.json


```
                }
            }
        ]
    },
    "example.com.": {
        "type": "authoritative",
        "stub": false,
        "analysis_start": "2016-01-26 14:54:59 UTC",
        "analysis_end": "2016-01-26 14:55:01 UTC",
        "clients_ipv4": [
            "10.0.2.15"
        ],
        "clients_ipv6": [],
        "parent": "com.",
        "referral_rdtype": "NS",
        "explicit_delegation": false,
        "nxdomain_name": "rph3tzkbls.example.com.",
        "nxdomain_rdtype": "A",
        "nxrrset_name": "example.com.",
        "nxrrset_rdtype": "CNAME",
        "auth_ns_ip_mapping": {
            "a.iana-servers.net.": [
                "199.43.132.53",
                "2001:500:8c::53"
            ],
            "b.iana-servers.net.": [
                "199.43.133.53",
                "2001:500:8d::53"
            ]
        },
        "queries": [
            {
                "qname": "example.com.",
                "qclass": "IN",
                "qtype": "A",
```

View `dnsviz grok` Output

example.com-p.json


```
{
    ".": {
        "status": "NOERROR",
        "queries": {
            "./IN/DNSKEY": {
                "answer": [
                    {
                        "id": "./IN/DNSKEY",
                        "description": "RRset for ./DNSKEY",
                        "name": ".",
                        "ttl": 172800,
                        "type": "DNSKEY",
                        "rdata": [
                            "256 3 8 AwEAAbr/RV0stAWYbmKOldjShp4AOQGO yY3
                            "257 3 8 AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29 eux
                        ],
                        "servers": [
                            "192.5.5.241",
                            "192.33.4.12",
                            "192.36.148.17",
                            "192.58.128.30",
                            "192.112.36.4",
                            "192.203.230.10",
                            "192.228.79.201",
                            "193.0.14.129",
                            "198.41.0.4",
                            "198.97.190.53",
                            "199.7.83.42",
                            "199.7.91.13",
                            "202.12.27.33"
                        ],
                        "query_options": [
                            "UDP_0_EDNS0_32768_4096"
                        ],
                        "rrsig": [
```

View `dnsviz grok` Output

example.com-p.json


```
            }
        },
        "dnskey": [
            {
                "id": "8/19036",
                "description": "DNSKEY for . (algorithm 8 (RSA/S
                "flags": 257,
                "protocol": 3,
                "algorithm": 8,
                "key": "AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhV\
                "ttl": 172800,
                "key_length": 2048,
                "key_tag": 19036,
                "servers": [
                    "192.5.5.241",
                    "192.33.4.12",
                    "192.36.148.17",
                    "192.58.128.30",
                    "192.112.36.4",
                    "192.203.230.10",
                    "192.228.79.201",
                    "193.0.14.129",
                    "198.41.0.4",
                    "198.97.190.53",
                    "199.7.83.42",
                    "199.7.91.13",
                    "202.12.27.33"
                ],
                "query_options": [
                    "UDP_0_EDNS0_32768_4096"
                ]
            },
            {
                "id": "8/54549",
                "description": "DNSKEY for . (algorithm 8 (RSA/S
```

View `dnsviz grok` Output

example.com-p.json


```
},
"rph3tzkbls.example.com./IN/A": {
    "nxdomain": [
        {
            "id": "rph3tzkbls.example.com./IN/A",
            "proof": [
                {
                    "id": "NSEC",
                    "description": "NSEC record(s) proving the non-existen
                    "nsec": [
                        {
                            "id": "example.com./IN/NSEC",
                            "description": "RRset for example.com/NSEC",
                            "name": "example.com.",
                            "ttl": 3600,
                            "type": "NSEC",
                            "rdata": [
                                "www.example.com. A NS SOA TXT AAAA RRSIG
                            ],
                            "servers": [
                                "199.43.132.53",
                                "199.43.133.53"
                            ],
                            "query_options": [
                                "UDP_0_EDNS0_32768_4096"
                            ],
                            "rrsig": [
                                {
                                    "id": "example.com./8/2718",
                                    "description": "RRSIG covering example
                                    "signer": "example.com.",
                                    "algorithm": 8,
                                    "key_tag": 2718,
                                    "original_ttl": 3600,
                                    "labels": 2,
                                    "inception": "2016 01 16 14:17:57 UTC"
```

View `dnsviz grok` Output

example.com-p1.json

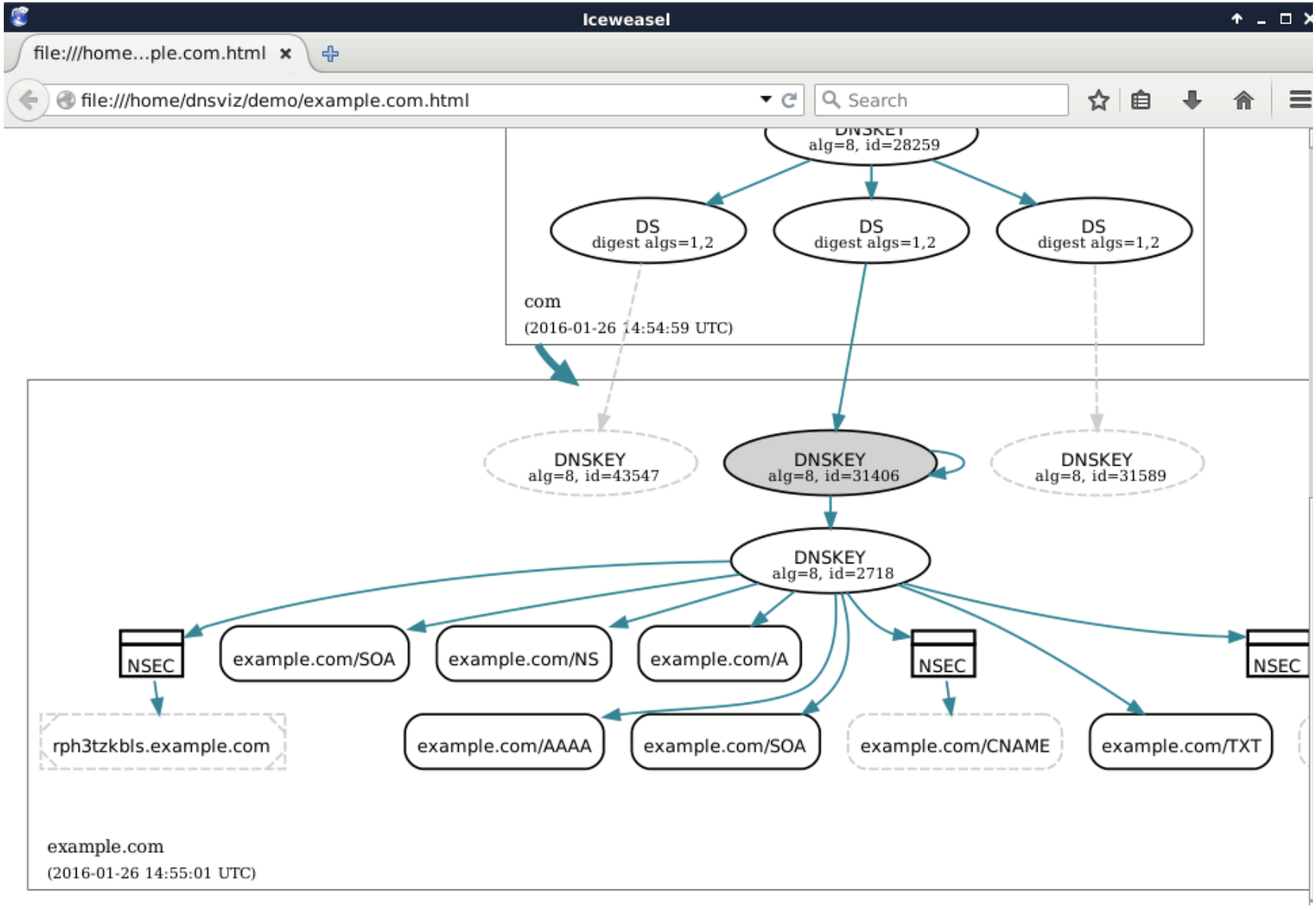

```
{
    ".": {
        "status": "NOERROR",
        "queries": {
            "./IN/DNSKEY": {
                "answer": [
                    {
                        "id": "./IN/DNSKEY",
                        "rrsig": [
                            {
                                "id": "./8/19036",
                                "status": "VALID",
                                "servers": [
                                    "192.5.5.241",
                                    "192.33.4.12",
                                    "192.36.148.17",
                                    "192.58.128.30",
                                    "192.112.36.4",
                                    "192.203.230.10",
                                    "192.228.79.201",
                                    "193.0.14.129",
                                    "198.41.0.4",
                                    "198.97.190.53",
                                    "199.7.83.42",
                                    "199.7.91.13",
                                    "202.12.27.33"
                                ],
                                "query_options": [
                                    "UDP_0_EDNS0_32768_4096"
                                ]
                            }
                        ],
                        "servers": [
                            "192.5.5.241",
                            "192.33.4.12",
                            "192.36.148.17",
```

View `dnsviz grok` Output

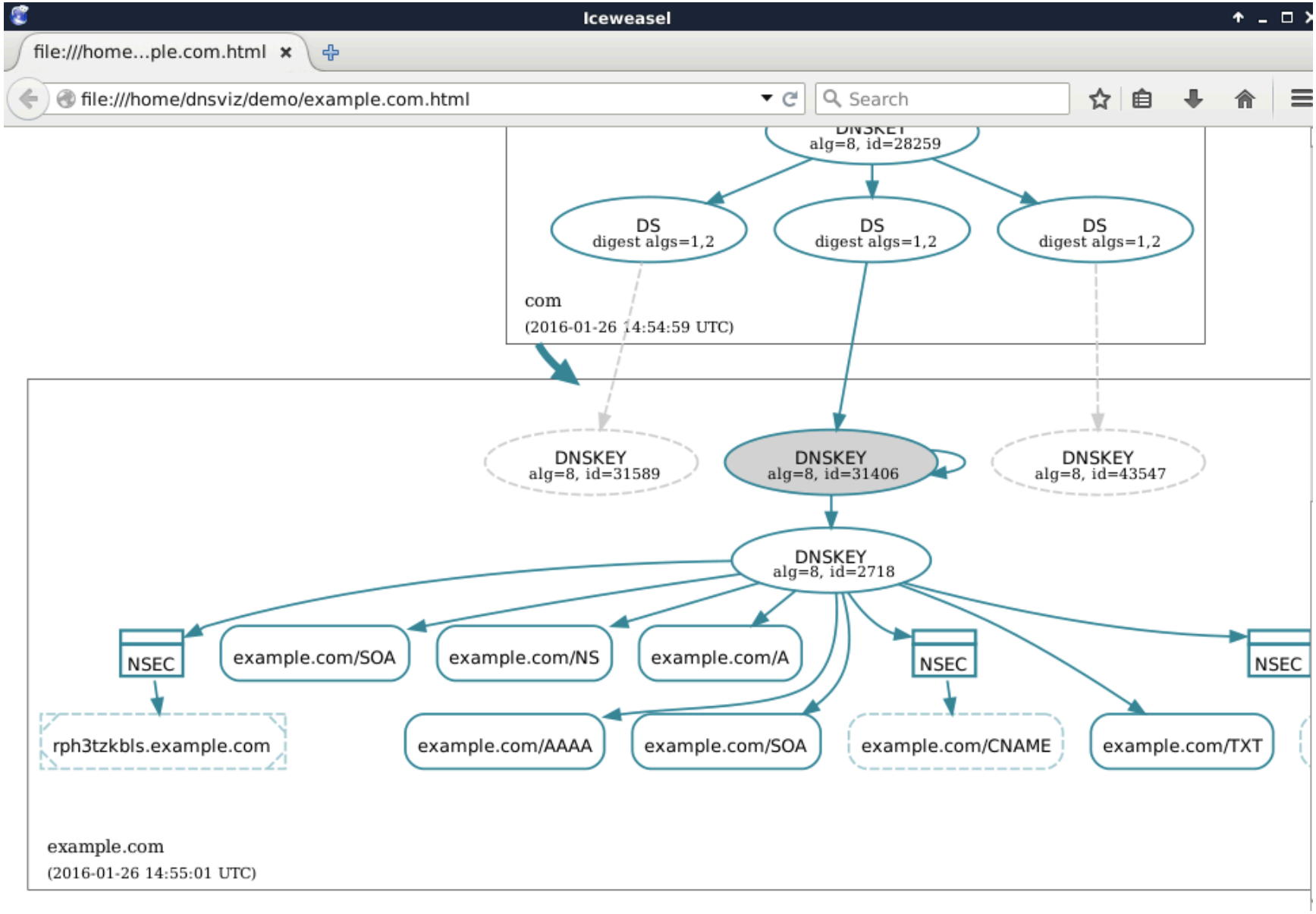
example.com-p1.json


```
                    "UDP_0_EDNS0_32768_4096"
                ]
            }
        ],
        "delegation": {
            "ds": [
                {
                    "id": "8/30909/2",
                    "status": "VALID",
                    "servers": [
                        "192.5.5.241",
                        "192.33.4.12",
                        "192.36.148.17",
                        "192.58.128.30",
                        "192.112.36.4",
                        "192.203.230.10",
                        "192.228.79.201",
                        "193.0.14.129",
                        "198.41.0.4",
                        "198.97.190.53",
                        "199.7.83.42",
                        "199.7.91.13",
                        "202.12.27.33"
                    ],
                    "query_options": [
                        "UDP_0_EDNS0_32768_4096"
                    ]
                }
            ],
            "status": "SECURE"
        }
    },
    "example.com.": {
        "status": "NOERROR",
        "queries": {
            "example.com./IN/A": {
```

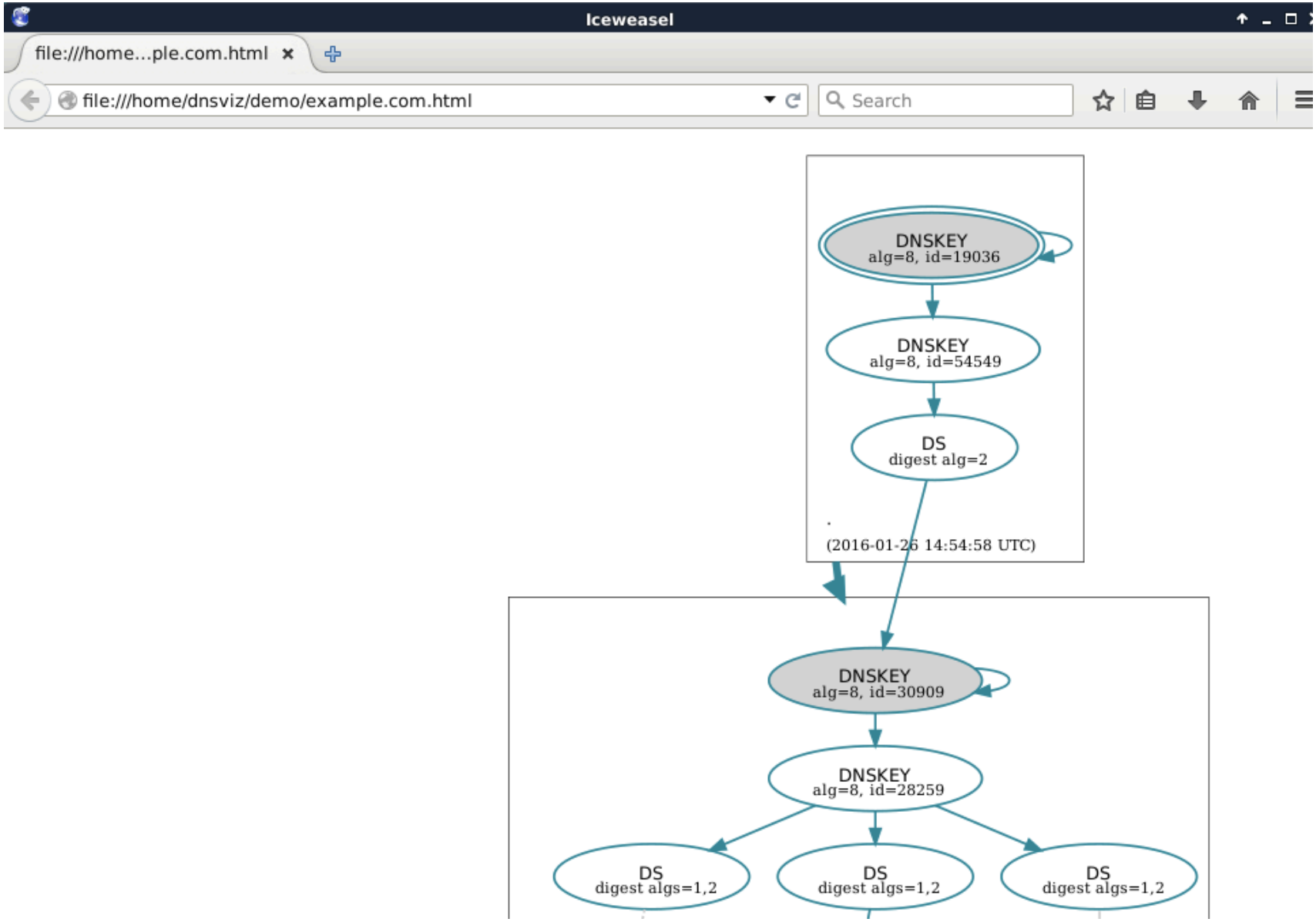
View `dnsviz graph` Output



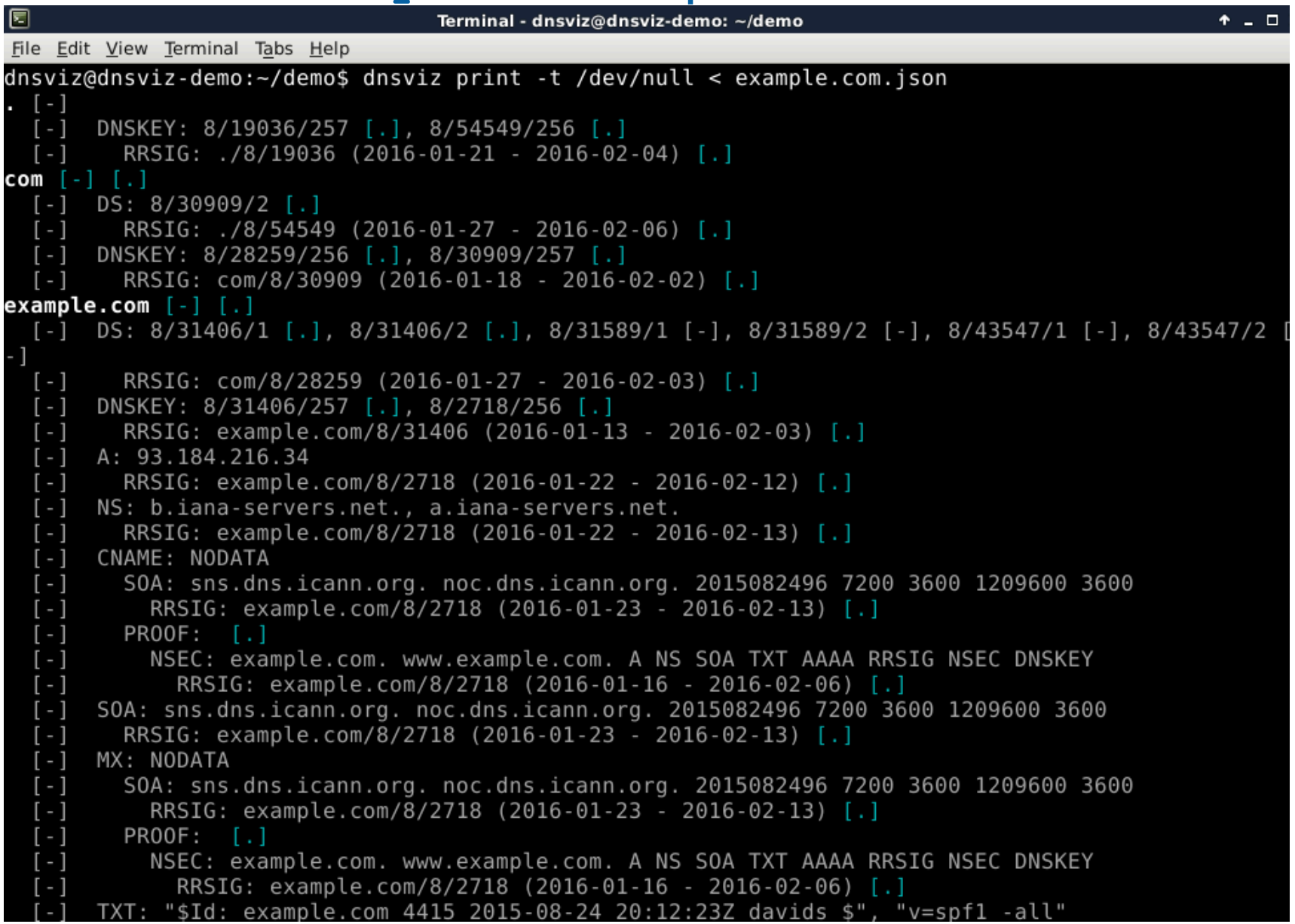
View `dnsviz graph` Output



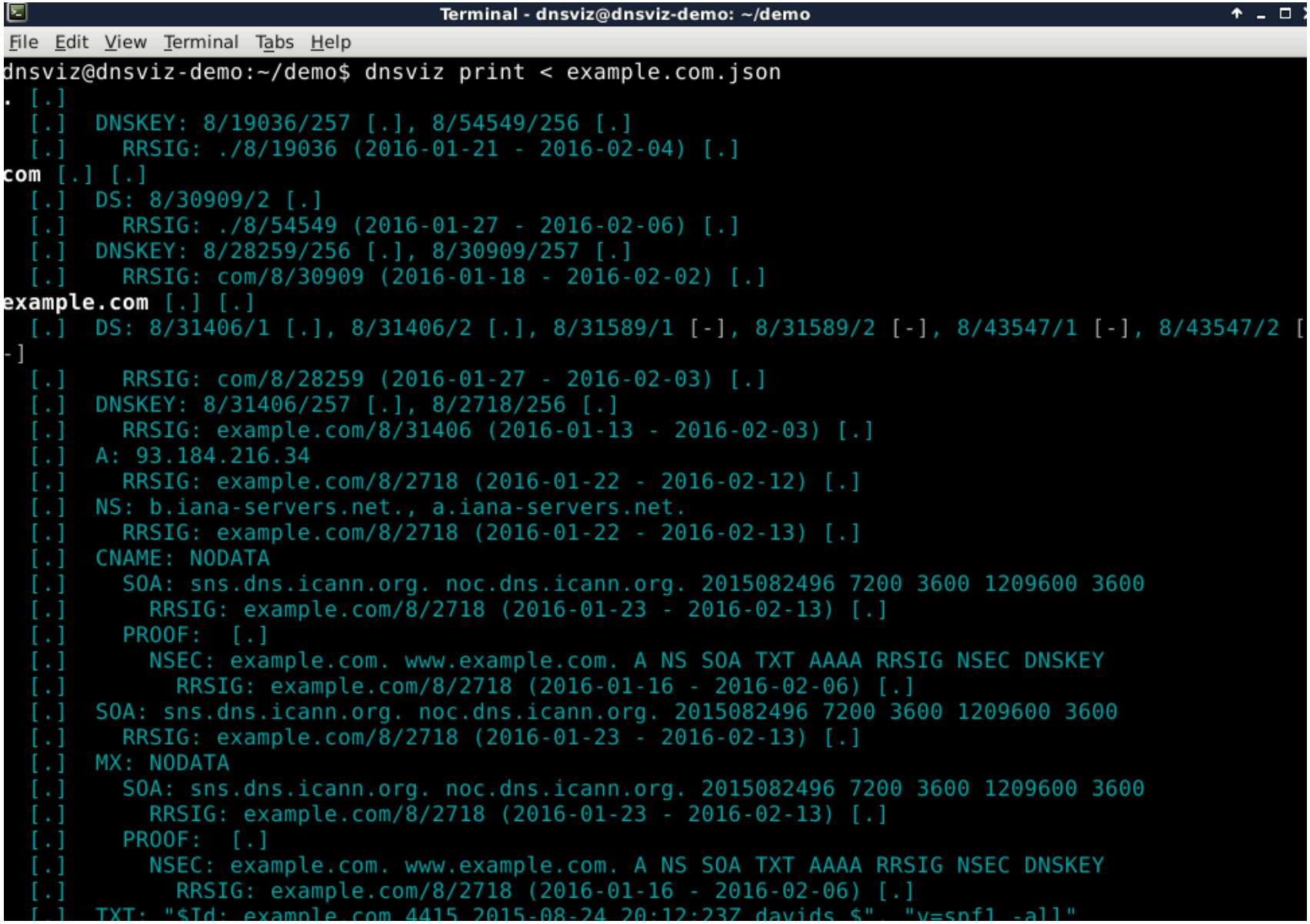
View `dnsviz graph` Output



View `dnsviz print` Output



View `dnsviz print` Output

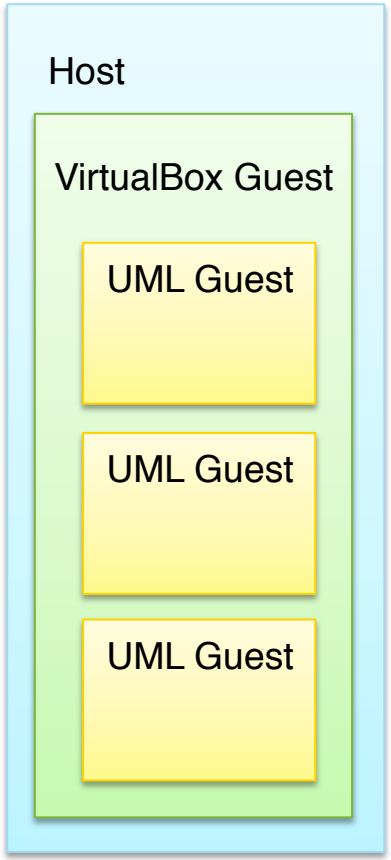


Signing a DNS Zone

Setup Virtual DNS Environment (4.1 – 4.2)

```
$ ./start_all
```

(Wait for all three consoles to come up)



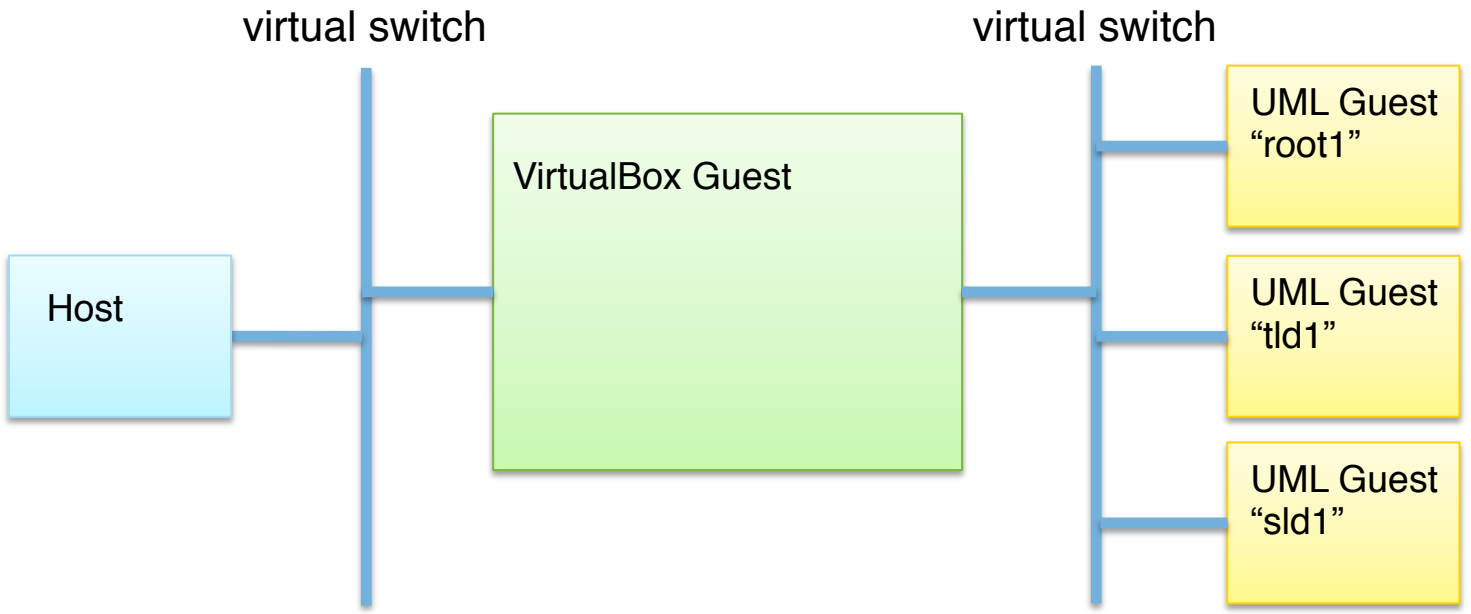
Change directory for all three consoles: root, tld1, sld1

```
$ cd /etc/bind
```

Setup Virtual DNS Environment (4.3)

```
$ ./dns_change_root local
```

(point DNS root hints and trusted keys to internal root server)



Analyze example.com in Local Environment (4.4 – 4.6)

Specify addresses for alternate (local) root servers

Pipe results directly to `dnsviz graph`, rather than redirecting to file

```
$ dnsviz probe -A -a .
        -x .:root1=192.168.213.10,root1=fd02:f00d:8::10 example.com | \
        dnsviz graph -Thtml -O -t tk-local.txt
```

Output analysis to file named "example.com.html"

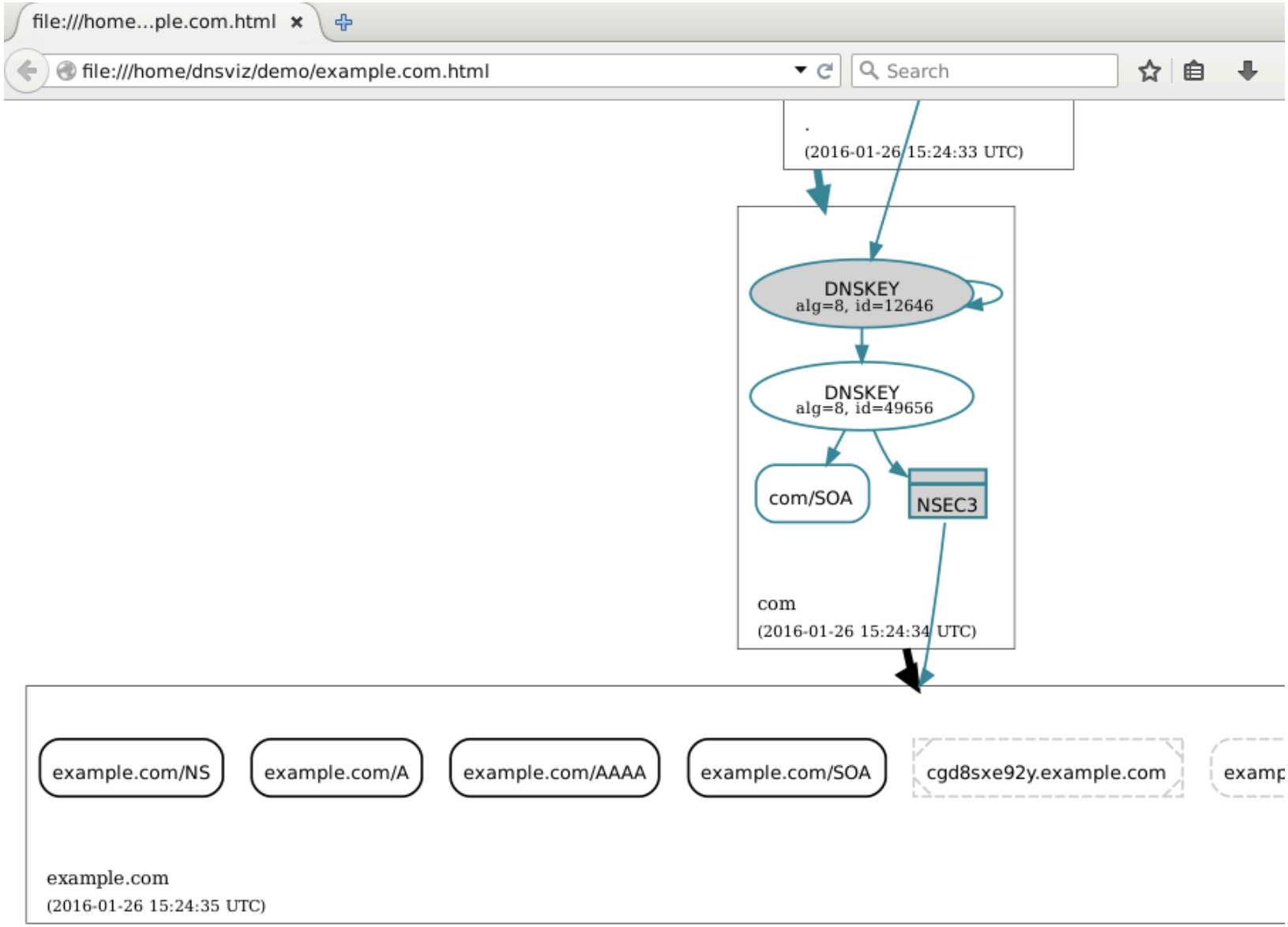
Use local trust anchor, rather than the one for the public root

```
$ ./dnsviz_analyze example.com
```

(script included for simplification)

```
$ firefox example.com.html &
```

View `dnsviz graph` Output



Add Records to example.com Zone (5.1 – 5.4)

- Add A records for names “a”, “c”, and “e” (on **sld1**)

(hint: see existing record for “www”)

```
# nano db.example.com
```

or

```
# vi db.example.com
```

- Check zone

```
# named-checkzone example.com db.example.com
```

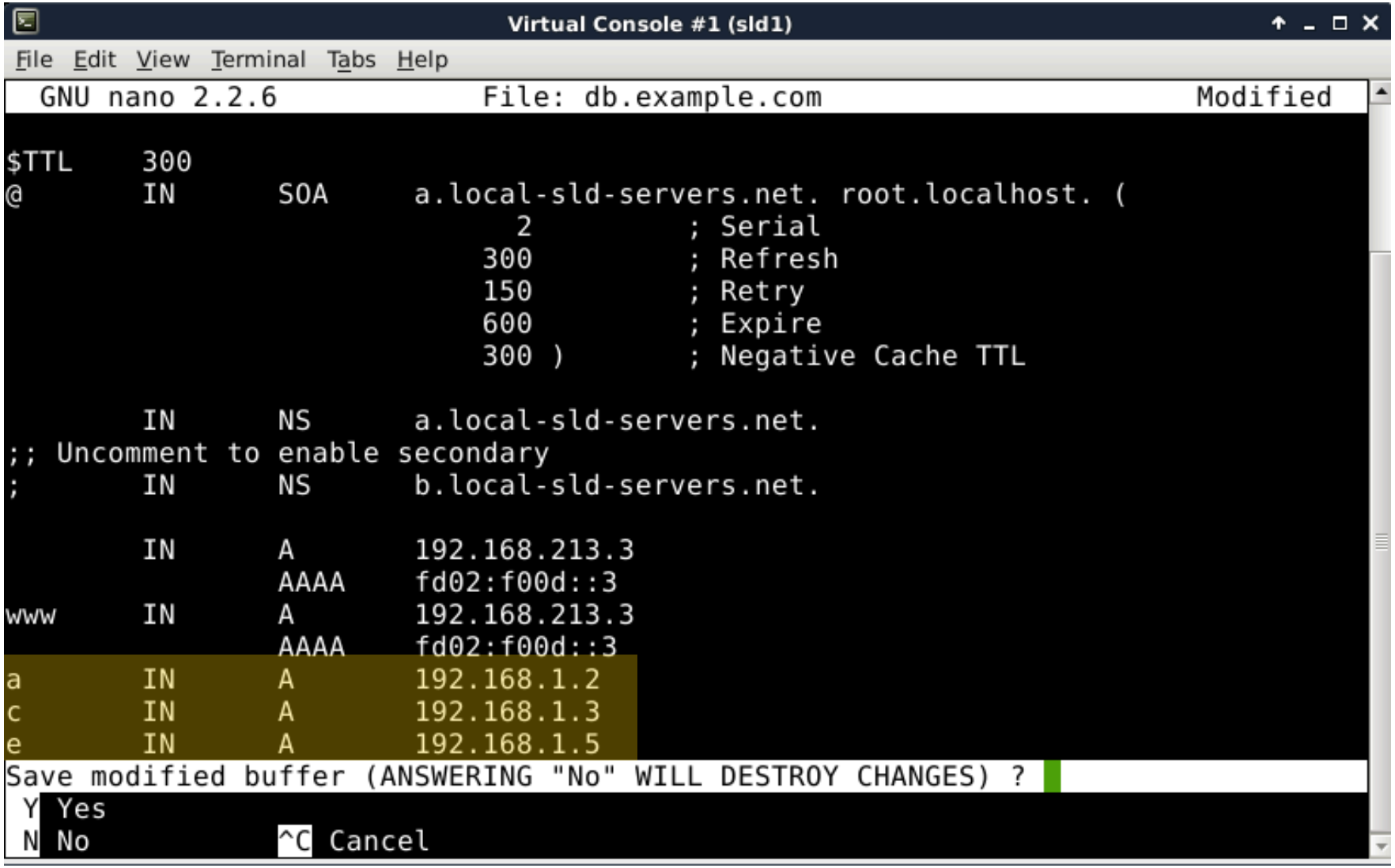
- Reload zone

```
# service bind9 reload
```

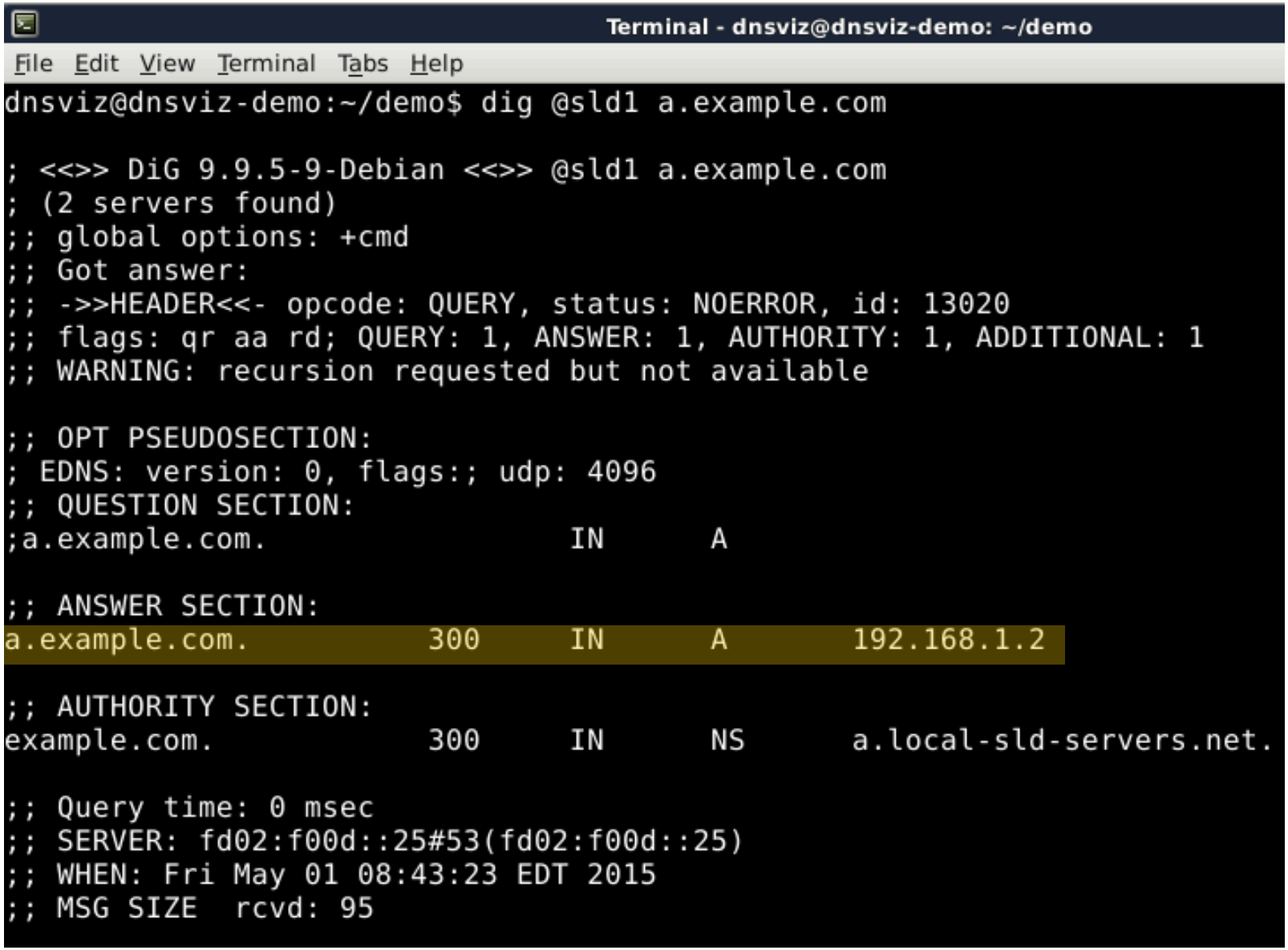
- Check that record shows up (query from VirtualBox guest)

```
$ dig @sld1 a.example.com
```

Add Records to example.com Zone



Add Records to example.com Zone



Create DNSSEC Keys for example.com Zone (6.1 – 6.3)

(on **sld1**)

Set the "SEP" bit for this DNSKEY

Use algorithm RSASHA256 for signing

Create a 2048-bit key

```
# KSK=`dnssec-keygen -n ZONE -f KSK -a RSASHA256 -b 2048 \
      -r /dev/urandom example.com`
```

No "SEP" bit here

Create a 1024-bit key

```
# ZSK=`dnssec-keygen -n ZONE -a RSASHA256 -b 1024 \
      -r /dev/urandom example.com`
```

```
# ls $KSK* $ZSK*
```

Add DNSKEY Records to example.com Zone (6.4 – 6.9)

- Look at DNSKEY records (on **sld1**):

```
# cat $KSK.key $ZSK.key
```

- Add DNSKEY records to zone

```
# cat $KSK.key $ZSK.key >> db.example.com
```

- Reload zone

```
# service bind9 reload
```

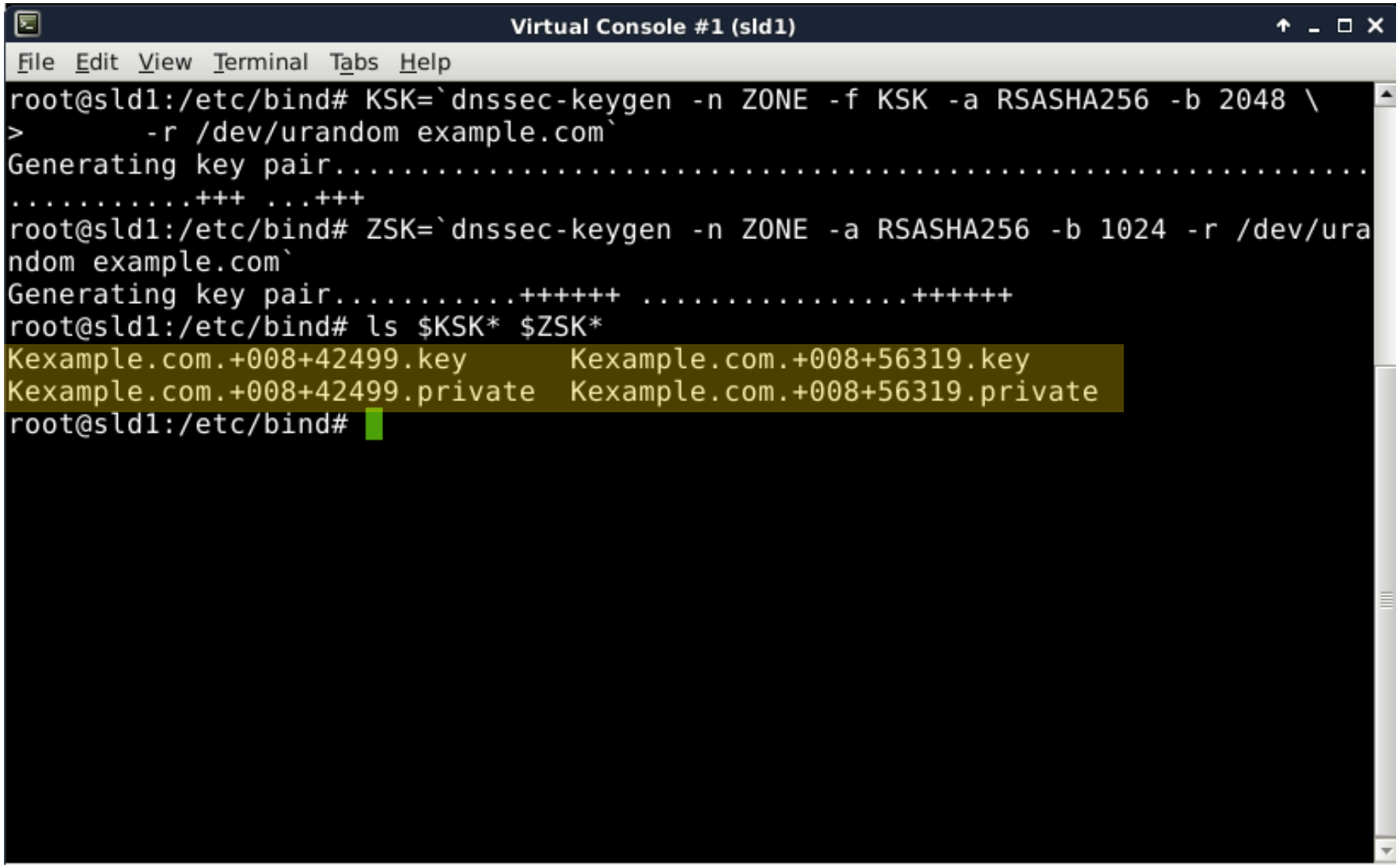
- Re-analyze

```
$ ./dnsviz_analyze example.com
```

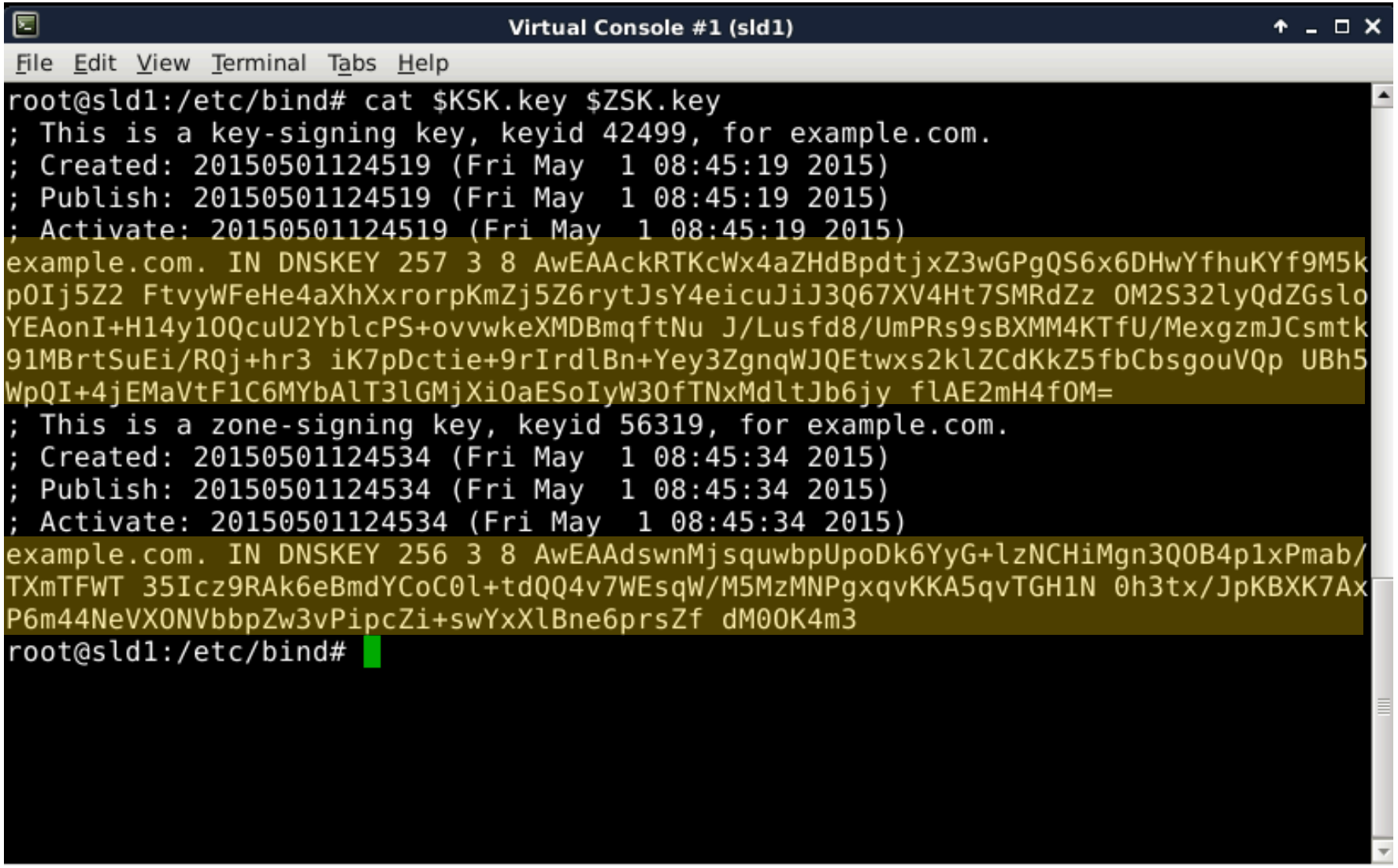
```
$ firefox example.com.html &
```

```
$ dig +noall +comment +ad example.com
```

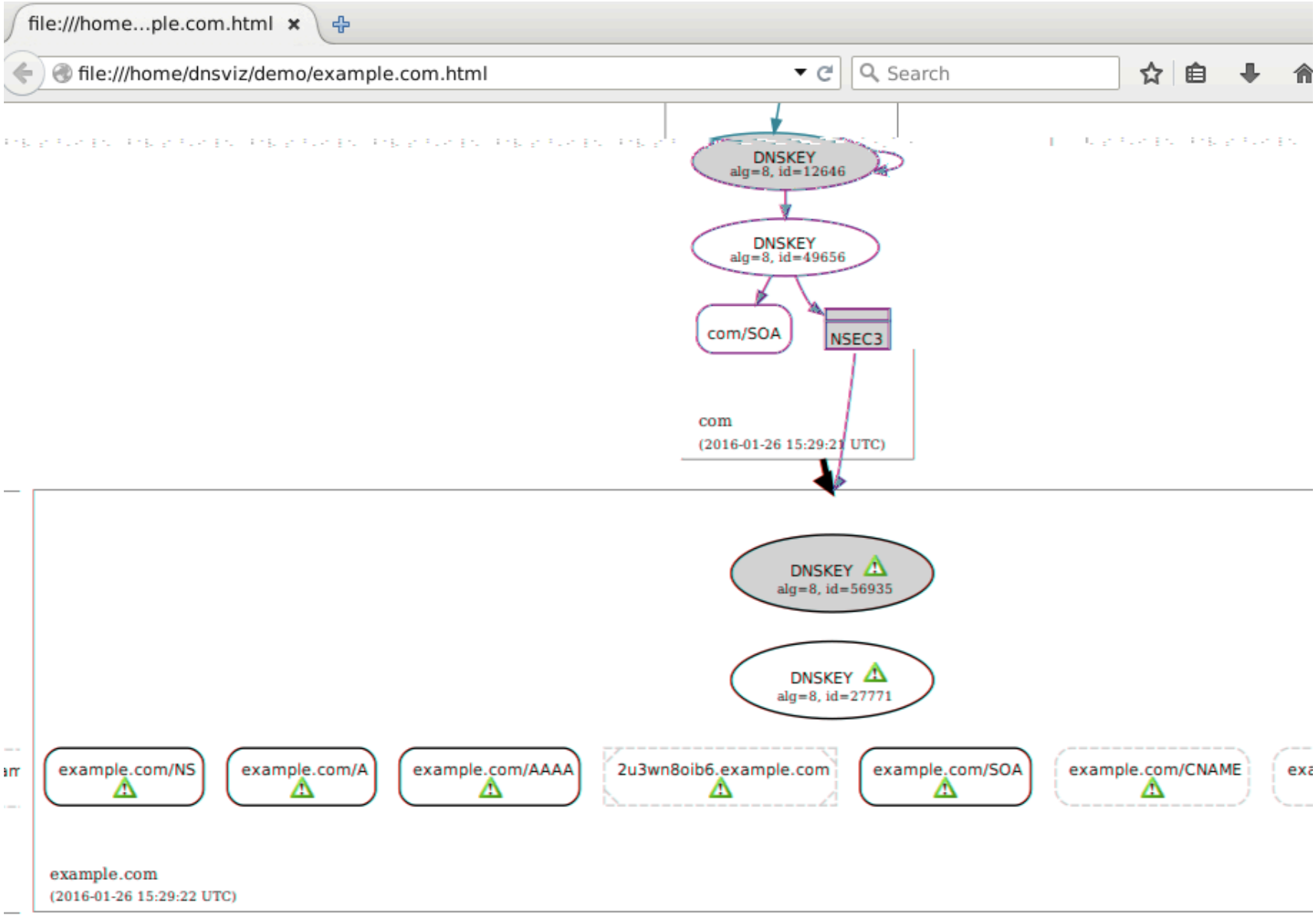
Create DNSSEC keys for example.com



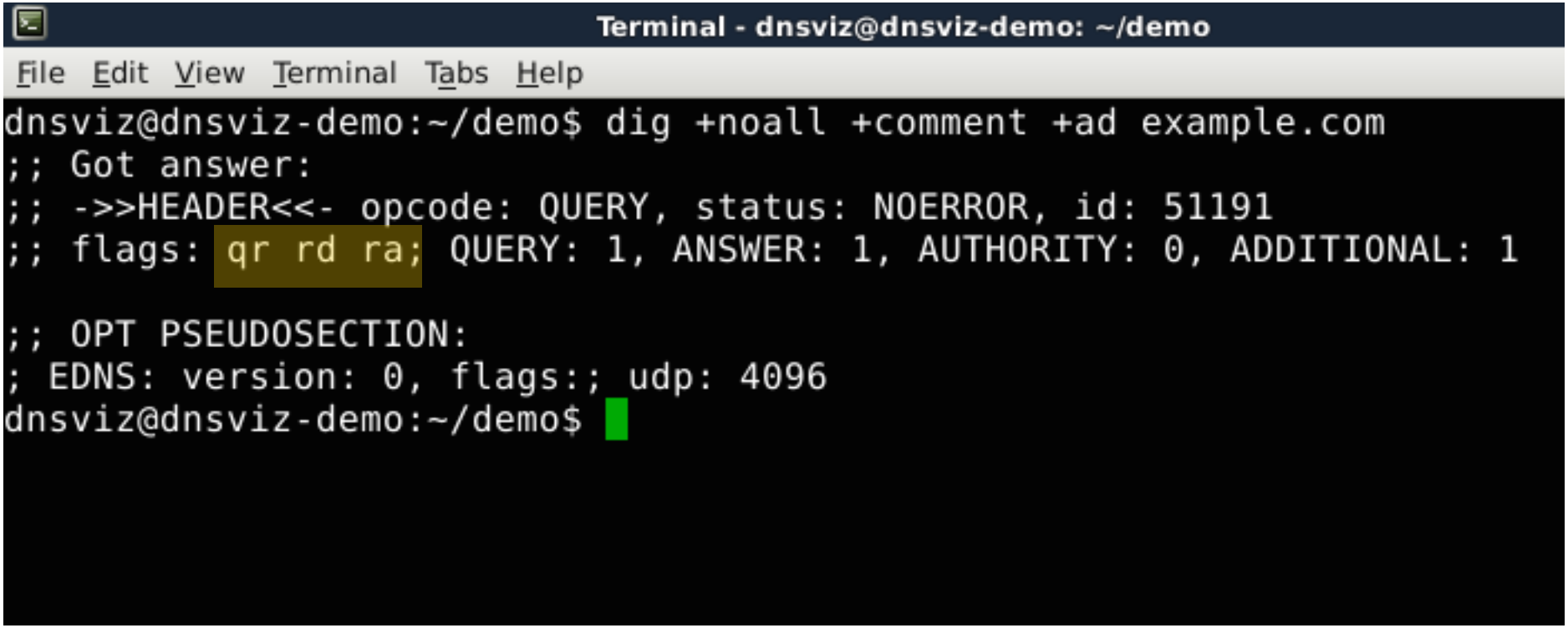
Create DNSSEC keys for example.com



View `dnsviz graph` Output: DNSKEYs with no RRSIGs



View `dig` Output: no AD bit



Sign Records in example.com Zone (7.1 – 7.4)

- Sign zone (**sld1**)

Use pseudo-random entropy source (**not for production use**)

```
# dnssec-signzone -r /dev/urandom \
    -k $KSK -o example.com db.example.com $ZSK
```

Sign only DNSKEY records with this key

Sign entire zone with this key

- Point named.conf to signed zone file

```
# sed -i -e 's:/db.example.com:&.signed:' named.conf.local
```

- Reload zone

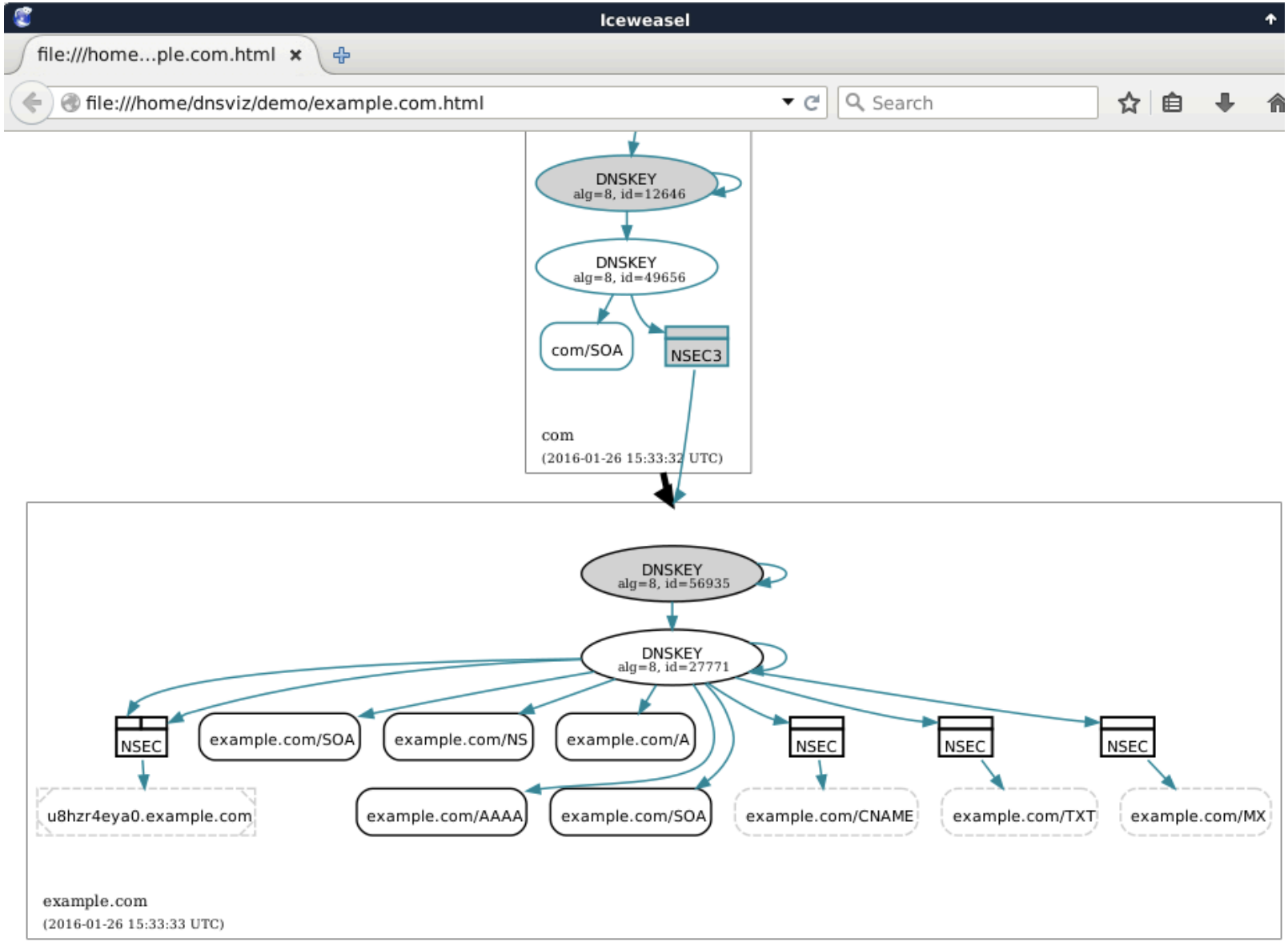
```
# service bind9 reload
```

```
$ ./dnsviz_analyze example.com
```

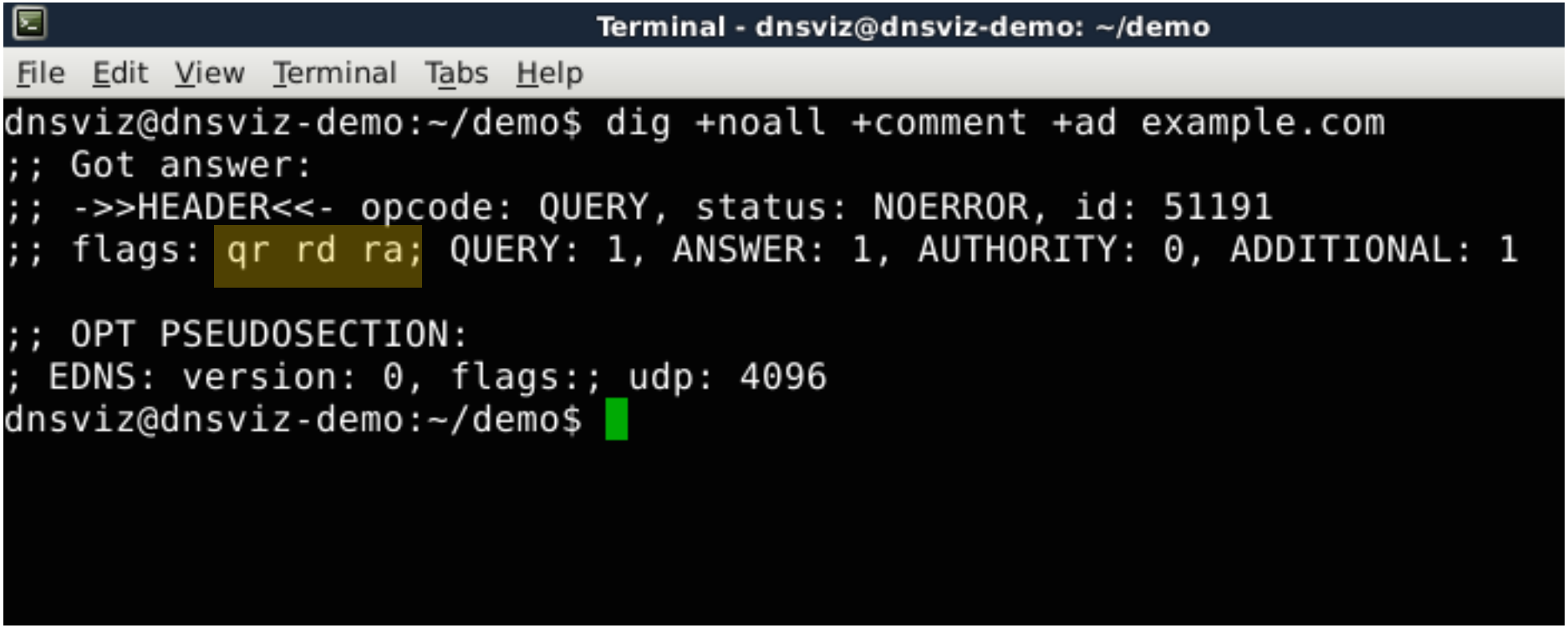
```
$ firefox example.com.html &
```

```
$ dig +noall +comment +ad example.com
```

View `dnsviz graph` Output: Signed example.com Zone



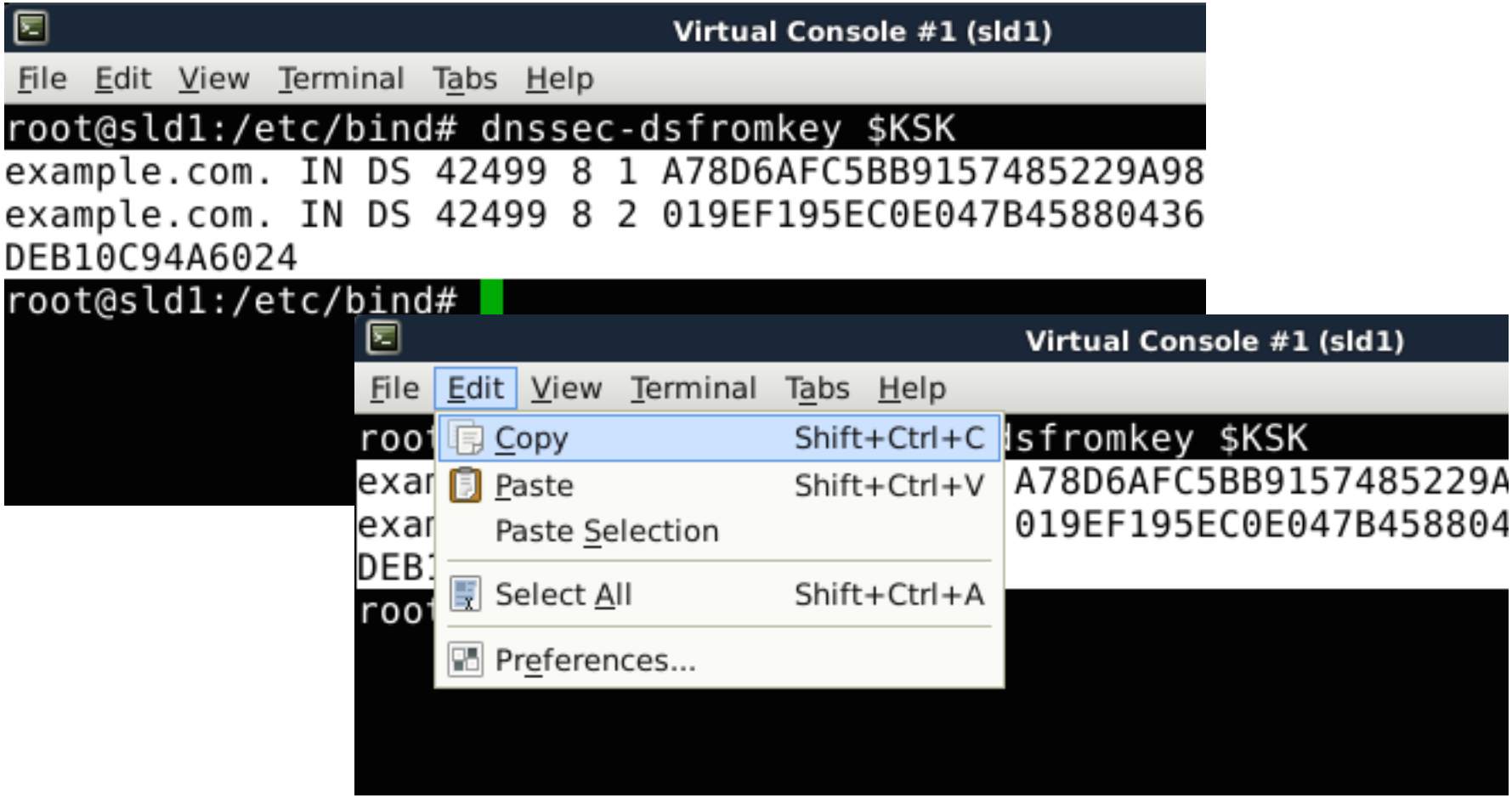
View `dig` Output: no AD bit



Generate DS Records for example.com (8.1 – 8.2)

- Create/copy DS records (on **sld1**)

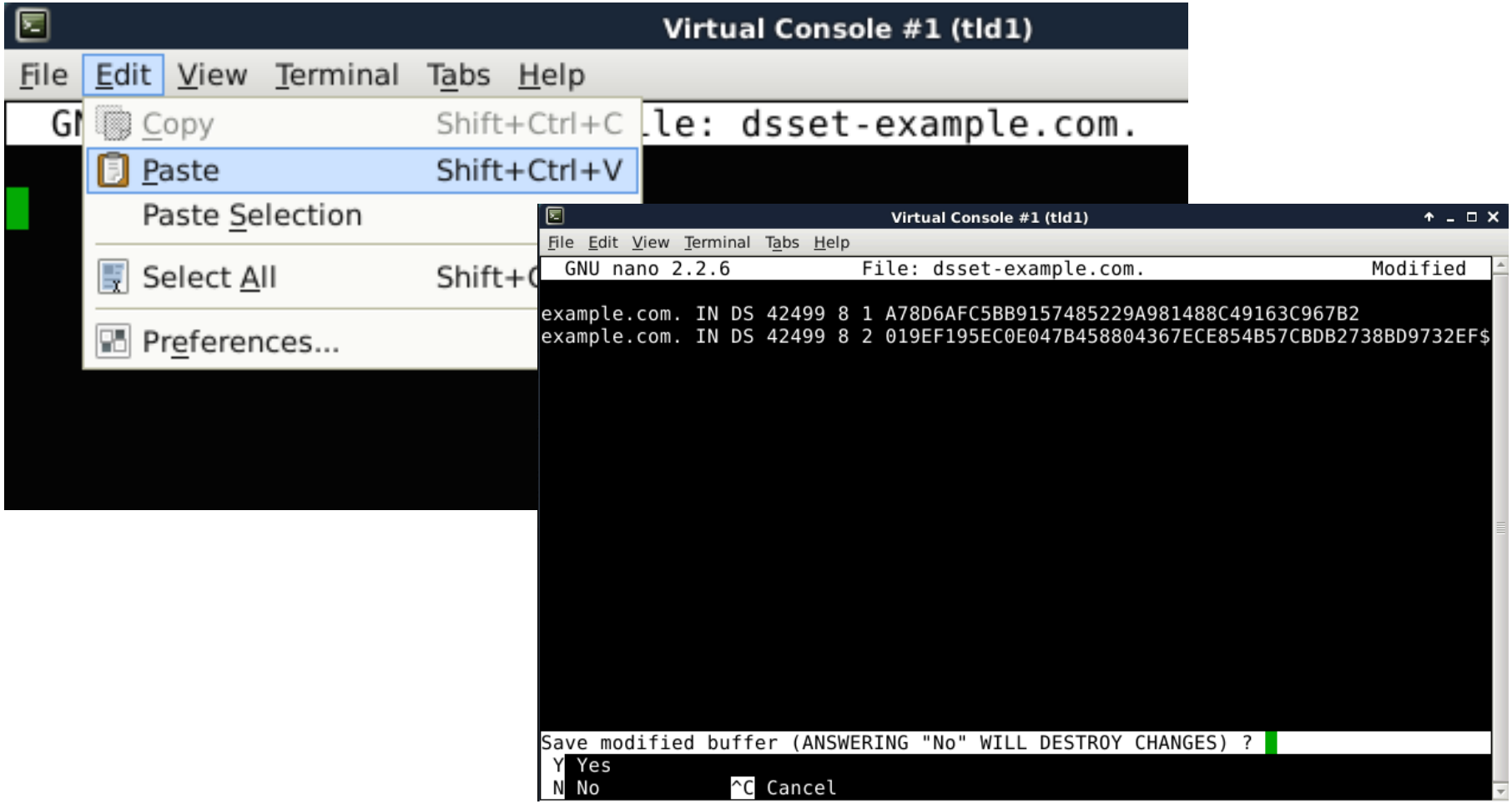
```
# dnssec-dsfromkey $KSK
```



Add DS Records for example.com (8.3a – 8.3c)

- Add DS records to “example” zone (on **tld1**)

```
# nano dsset-example.com.
```



Sign Records in “example.com” Zone (8.4 – 8.5)

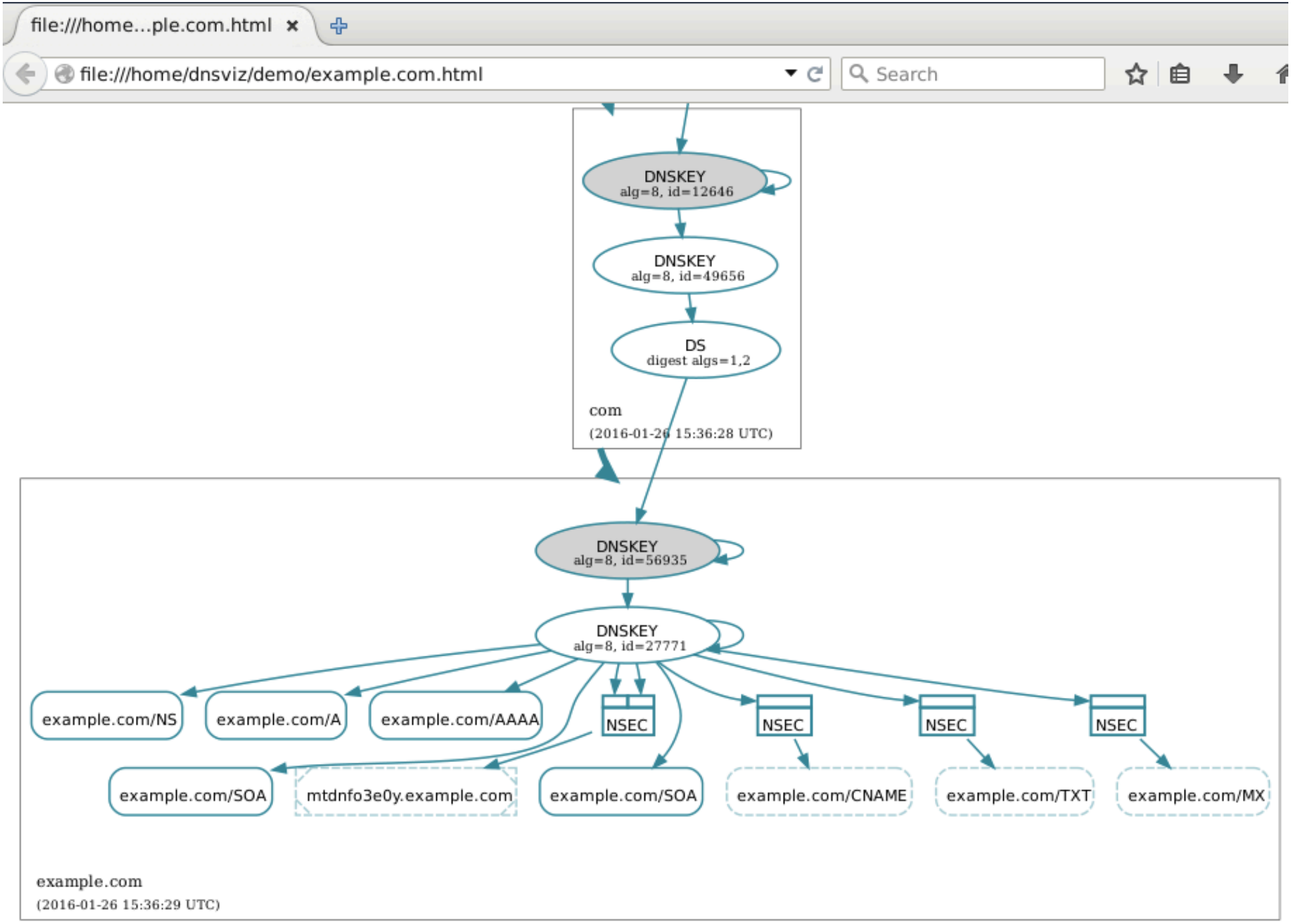
- Sign zone (on **tld1**)

```
# ./resign_tld
```

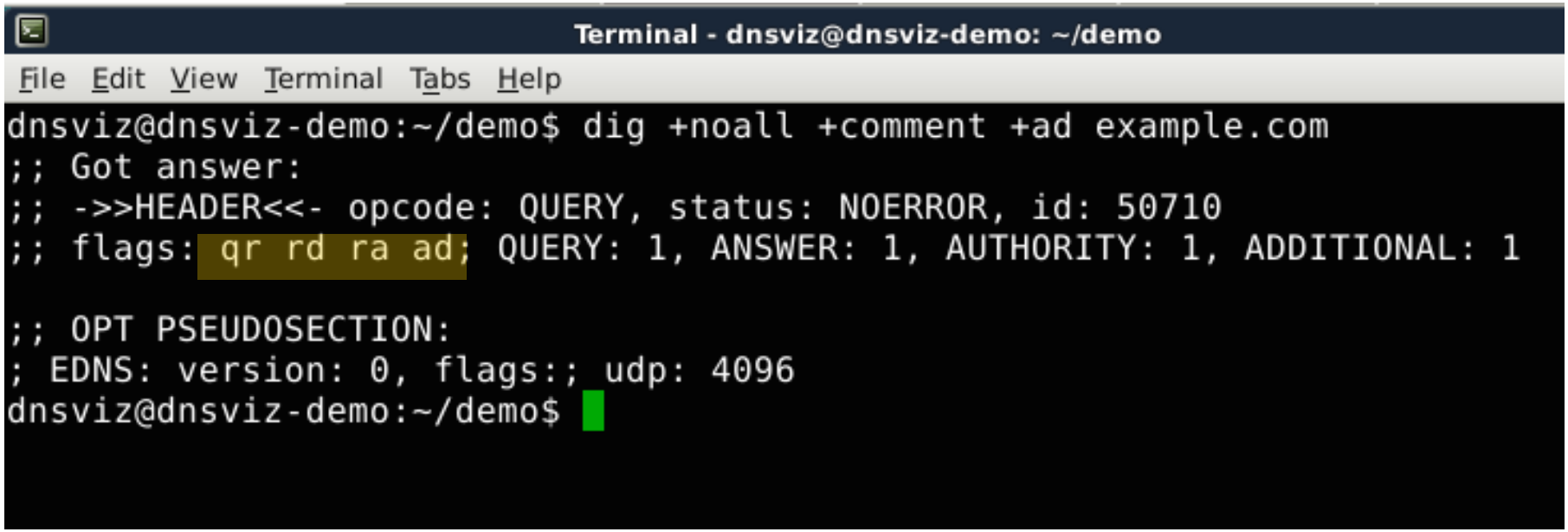
```
$ ./dnsviz_analyze example.com
```

```
$ firefox example.com.html &
```

```
$ dig +noall +comment +ad example.com
```


file:///home...ple.com.html
file:///home/dnsviz/demo/example.com.html
Search
DNSKEY
alg=8, id=12646
DNSKEY
alg=8, id=49656
DS
digest algs=1,2
com
(2016-01-26 15:36:28 UTC)
DNSKEY
alg=8, id=56935
DNSKEY
alg=8, id=27771
example.com/NS
example.com/A
example.com/AAAA
NSEC
NSEC
NSEC
NSEC
example.com/SOA
mtdnfo3e0y.example.com
example.com/SOA
example.com/CNAME
example.com/TXT
example.com/MX
example.com
(2016-01-26 15:36:29 UTC)

View `dig` Output: AD bit



Fun with DNSViz

Use KSK to Only Sign DNSKEY RRset (9.1 – 9.3)

Don't sign zone data with KSK

```
# dnssec-signzone -x -r /dev/urandom \
    -k $KSK -o example.com db.example.com $ZSK
```

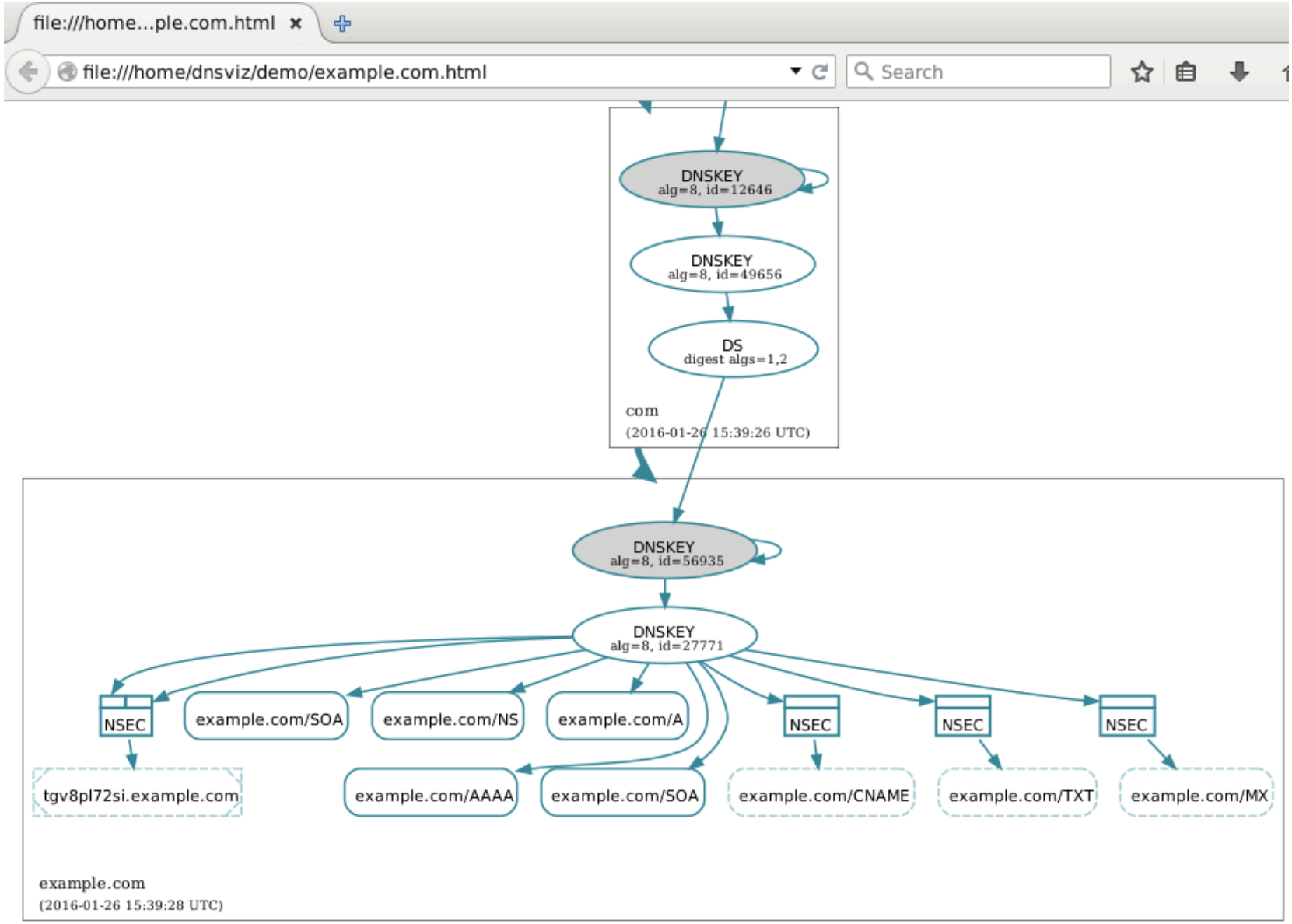
```
# service bind9 reload
```

```
$ ./dnsviz_analyze example.com
```

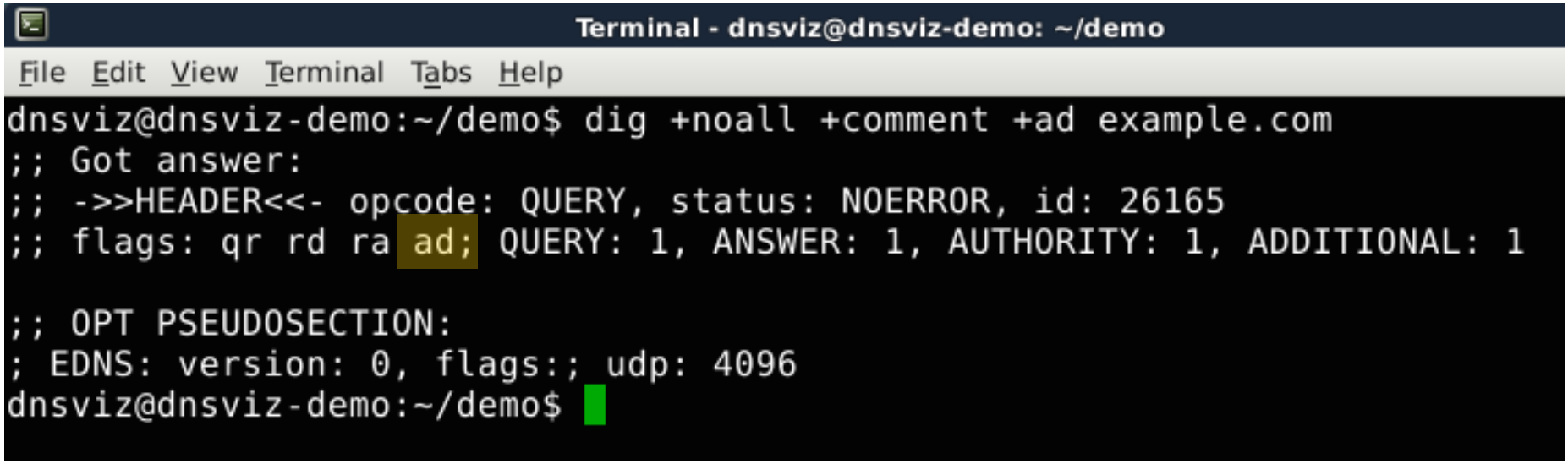
```
$ firefox example.com.html &
```

```
$ dig +noall +comment +ad example.com
```

View `dnsviz graph` Output: KSK-only



View `dig` Output: AD bit



Add New KSK to example.com Zone (9.4 – 9.8)

- Generate new KSK:

```
# NEWKSK=`dnssec-keygen -n ZONE -f KSK -a RSASHA256 -b 2048 \
      -r /dev/urandom example.com`
```

```
# cat $NEWKSK.key >> db.example.com
```

- Re-sign zone:

```
# dnssec-signzone -x -r /dev/urandom \
    -k $KSK -o example.com db.example.com $ZSK
```

- Reload zone

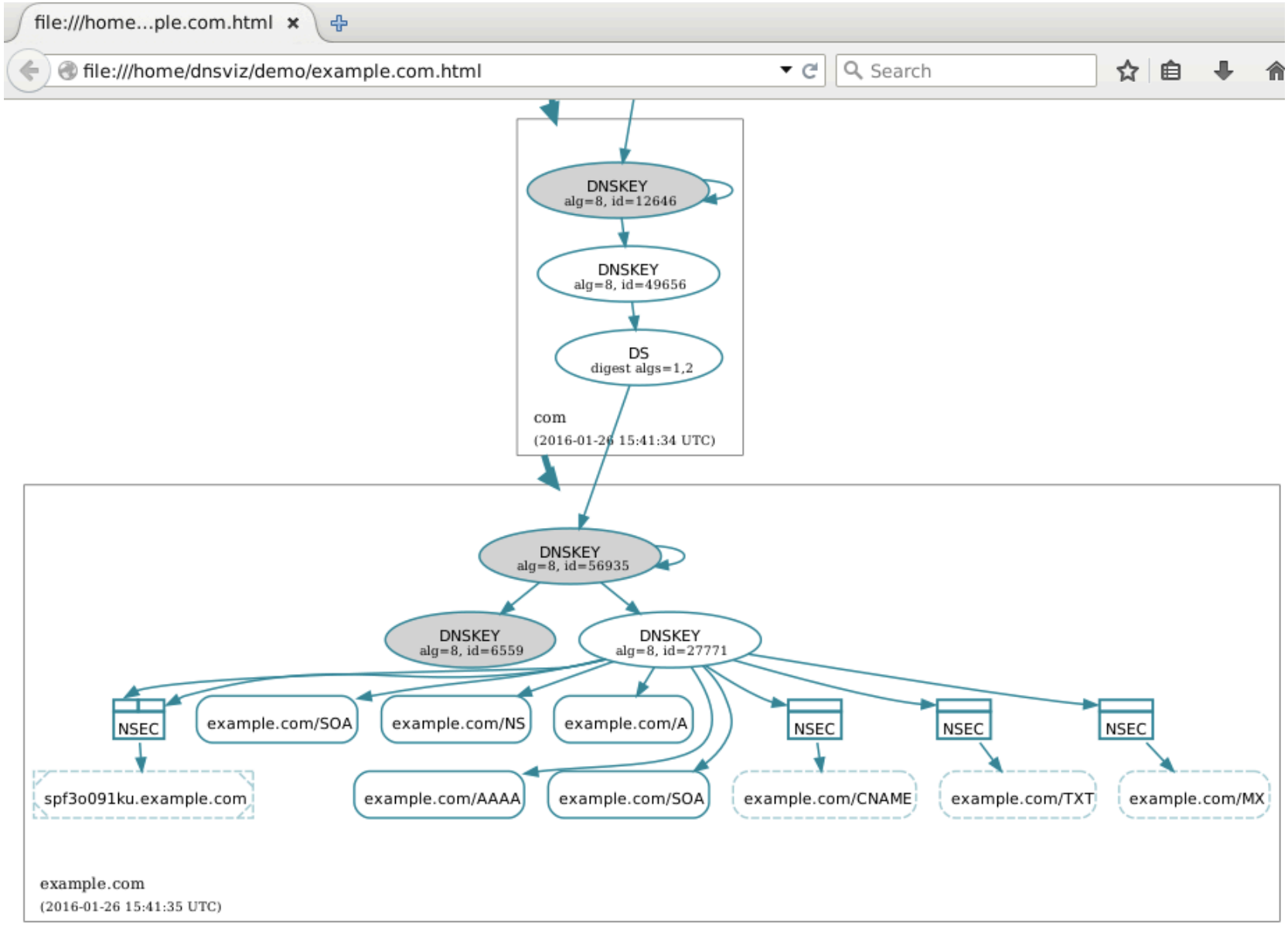
```
# service bind9 reload
```

```
$ ./dnsviz_analyze example.com
```

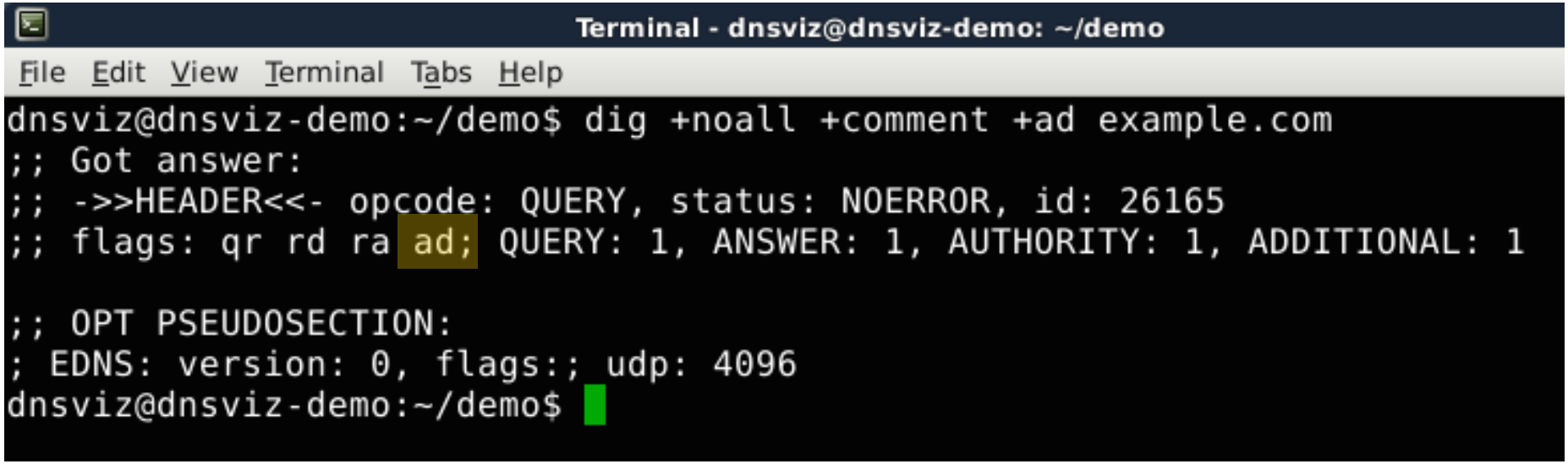
```
$ firefox example.com.html &
```

```
$ dig +noall +comment +ad example.com
```

View `dnsviz graph` Output: Standby KSK



View `dig` Output: AD bit



Add New KSK to example.com Zone (9.9 – 9.11)

- Re-sign zone with two KSKs:

```
# dnssec-signzone -x -r /dev/urandom \
    -k $KSK -k $NEWKSK -o example.com db.example.com $ZSK
```

- Reload zone

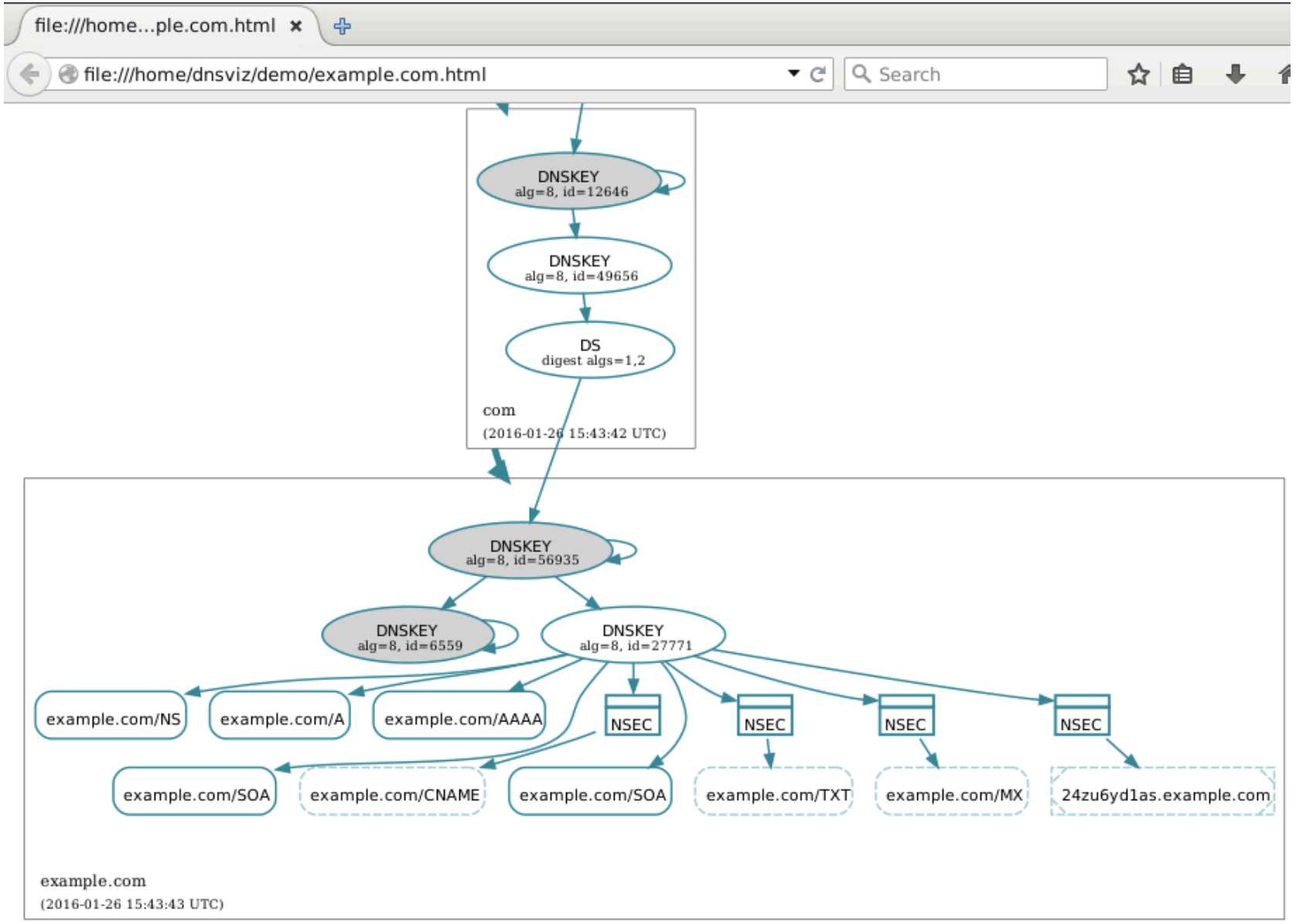
```
# service bind9 reload
```

```
$ ./dnsviz_analyze example.com
```

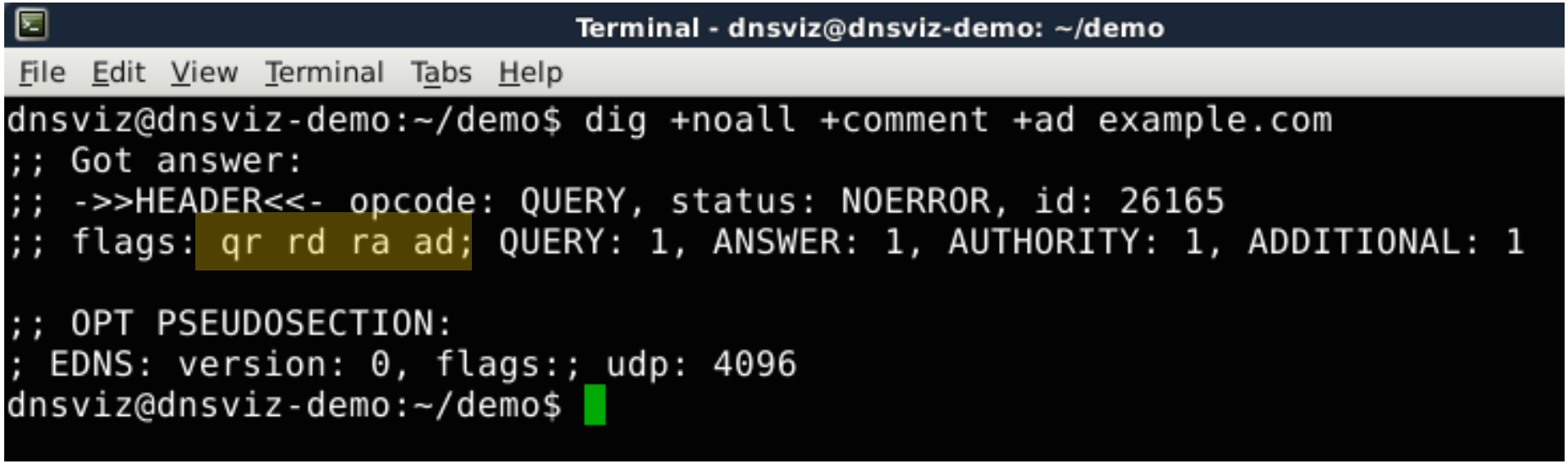
```
$ firefox example.com.html &
```

```
$ dig +noall +comment +ad example.com
```

View `dnsviz graph` Output: Multiple KSKs



View `dig` Output: AD bit



Change KSK for example.com Zone (9.12 – 9.14)

- Sign with only the second KSK:

```
# dnssec-signzone -x -r /dev/urandom \
    -k $NEWKSK -o example.com db.example.com $ZSK
```

- Reload zone

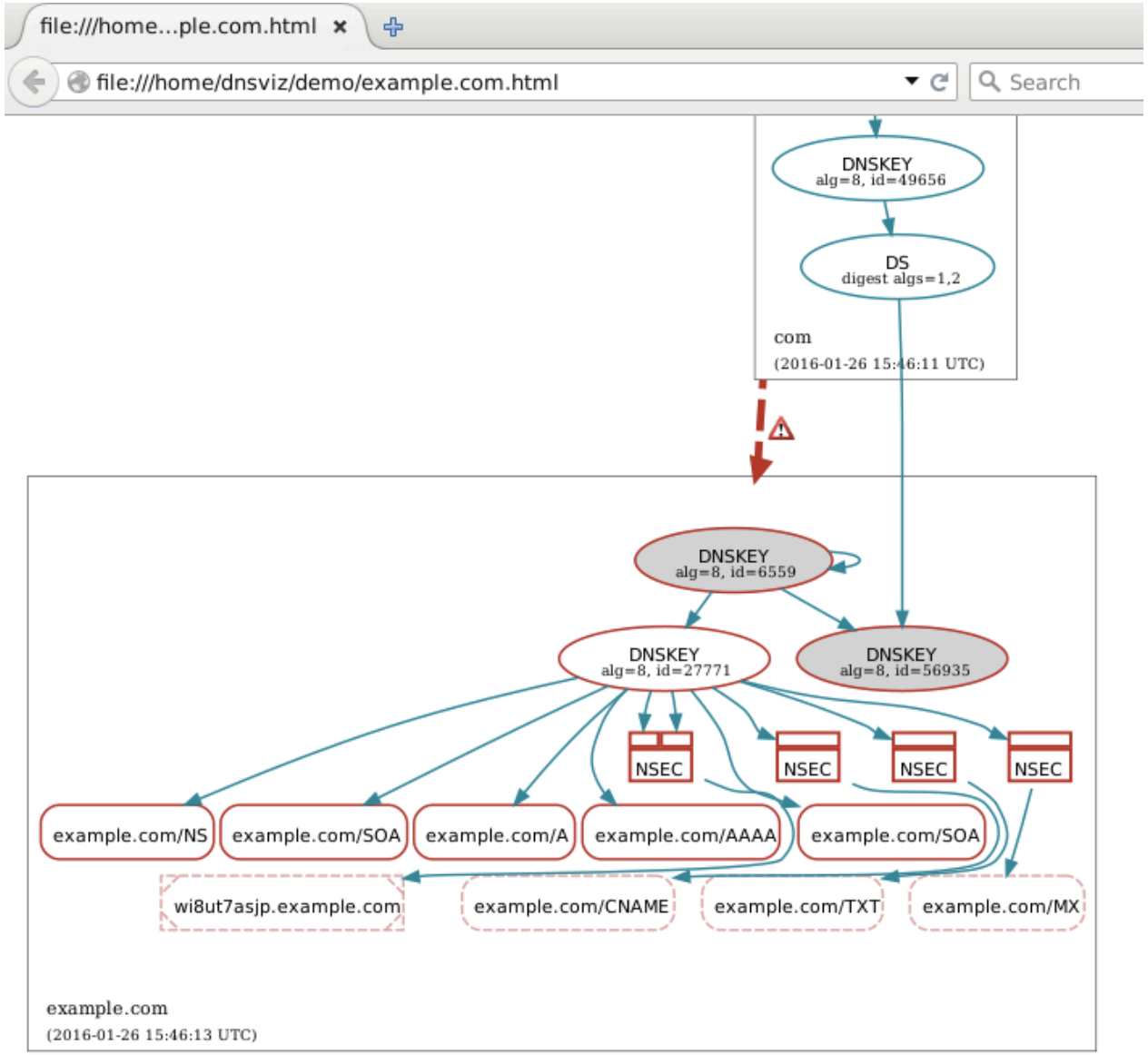
```
# service bind9 reload
```

```
$ ./dnsviz_analyze example.com
```

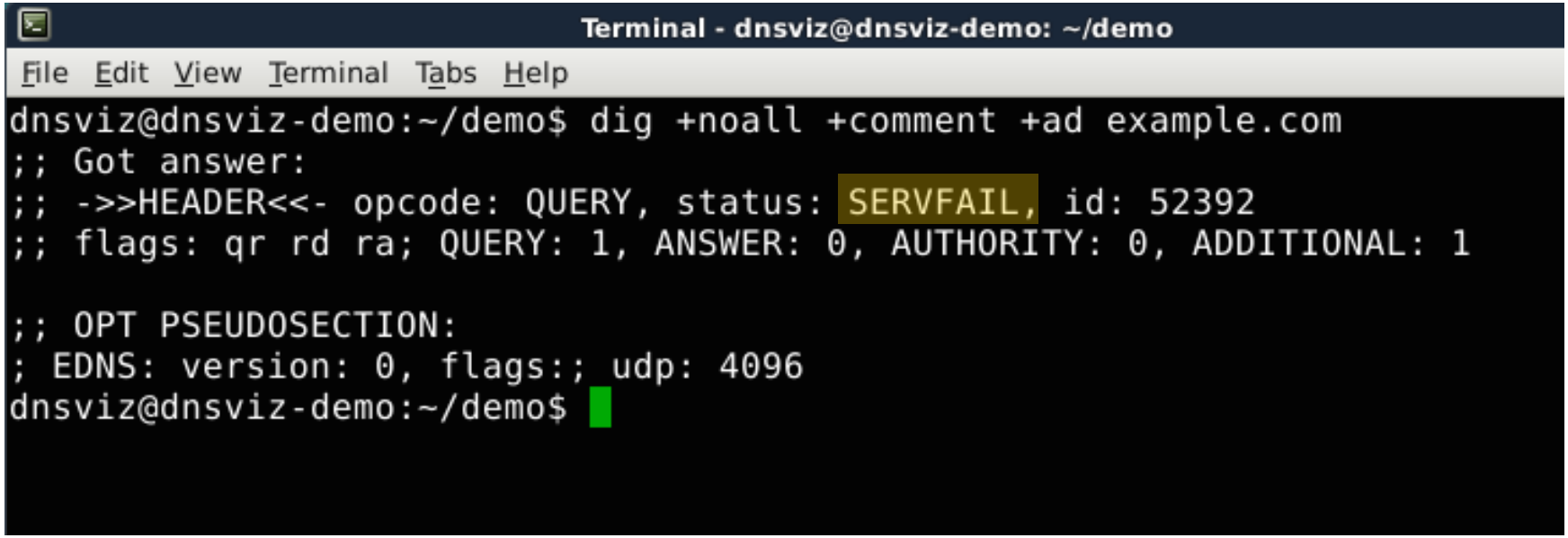
```
$ firefox example.com.html &
```

```
$ dig +noall +comment +ad example.com
```

View `dnsviz graph` Output: DS Mismatch



View `dig` Output: SERVFAIL



Tamper with Record Content (9.15 – 9.17)

- Change SOA record:

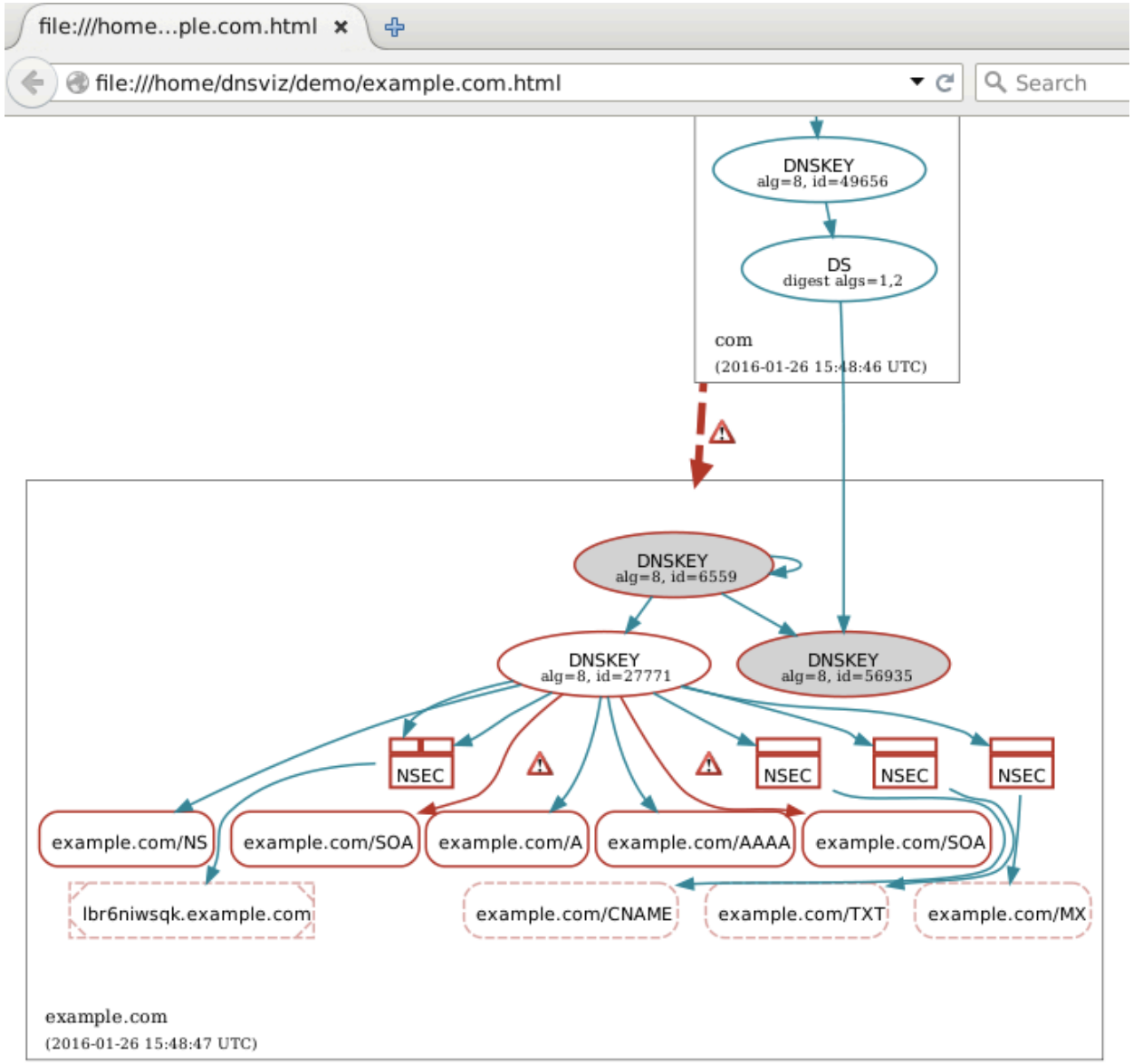
```
# sed -i -e 's/root.localhost/root1.localhost/' \
    db.example.com.signed
```

```
# service bind9 reload
```

```
$ ./dnsviz_analyze example.com
```

```
$ firefox example.com.html &
```

View `dnsviz graph` Output: Invalid Signatures



Change RRSIG Expiration (9.18 – 9.21)

- Set the RRSIG expiration explicitly to 1 second from “now”

```
# dnssec-signzone -x -e now+1 -r /dev/urandom \
    -k $NEWKSK -o example.com db.example.com $ZSK
```

- Manipulate (again) SOA record

```
# sed -i -e ‘s/root.localhost/root1.localhost/’ \
    db.example.com.signed
```

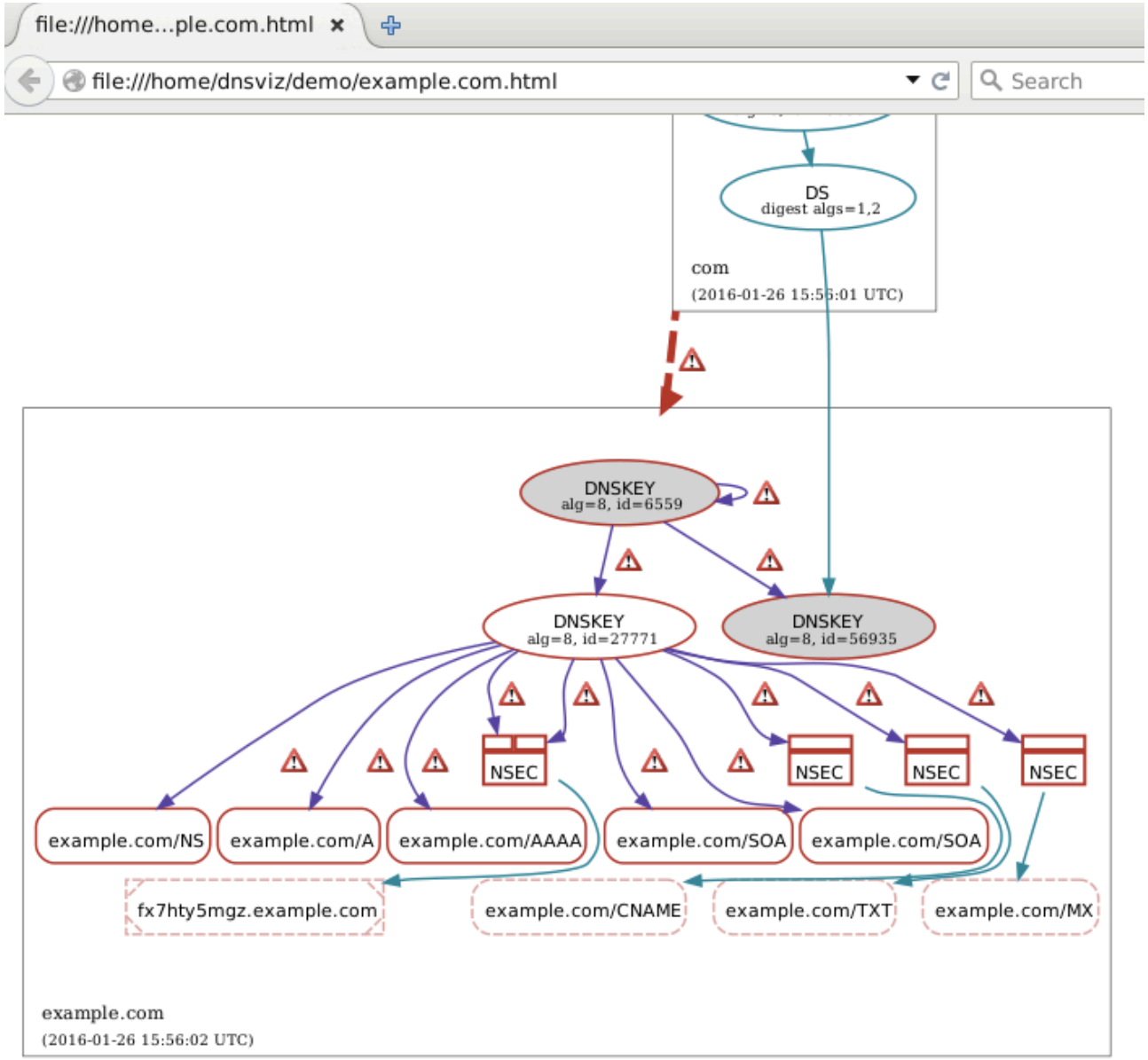
- Reload zone

```
# service bind9 reload
```

```
$ ./dnsviz_analyze example.com
```

```
$ firefox example.com.html &
```

View `dnsviz graph` Output: Expired RRSIGs



Remove RRSIGs (9.22 – 9.25)

- Remove RRSIG covering AAAA record (on **sld1**)

```
# nano db.example.com.signed
```

or

```
# vi db.example.com.signed
```

- Check zone

```
# named-checkzone example.com db.example.com.signed
```

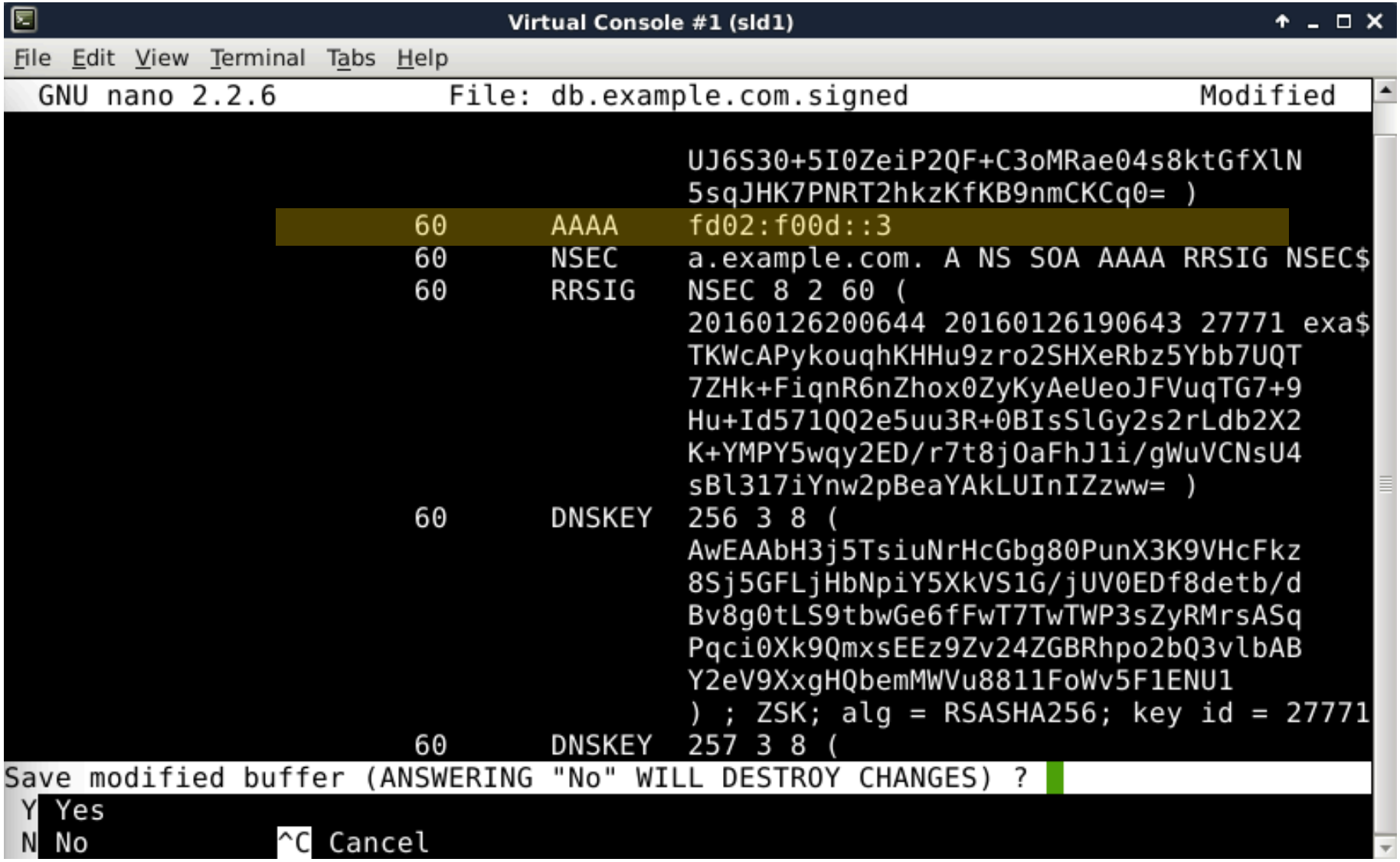
- Reload zone

```
# service bind9 reload
```

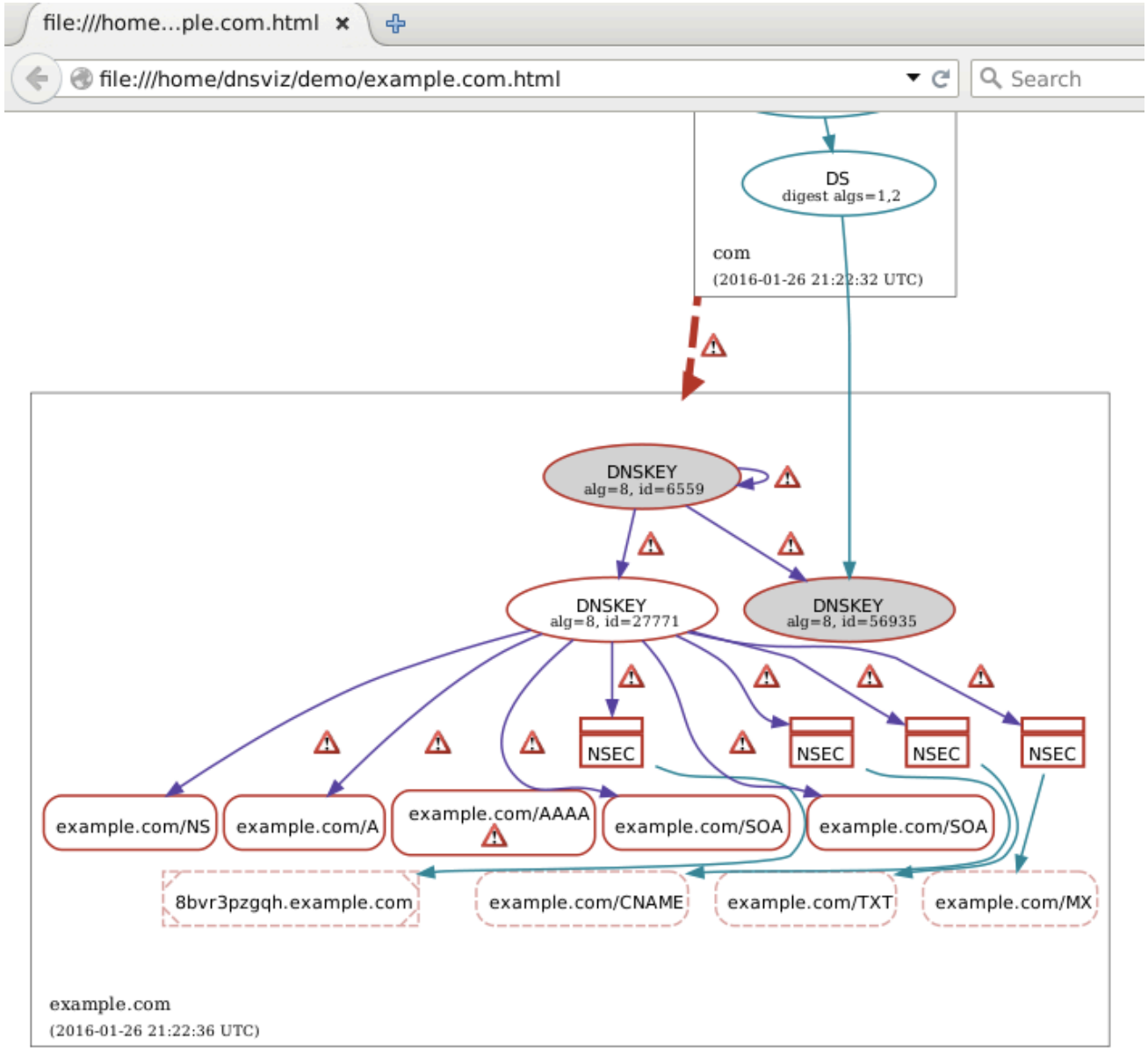
```
$ ./dnsviz_analyze example.com
```

```
$ firefox example.com.html &
```

Remove RRSIG for AAAA Record from Zone



View `dnsviz graph` Output: Missing RRSIGs



Modify TCP Connectivity (9.26 – 9.27)

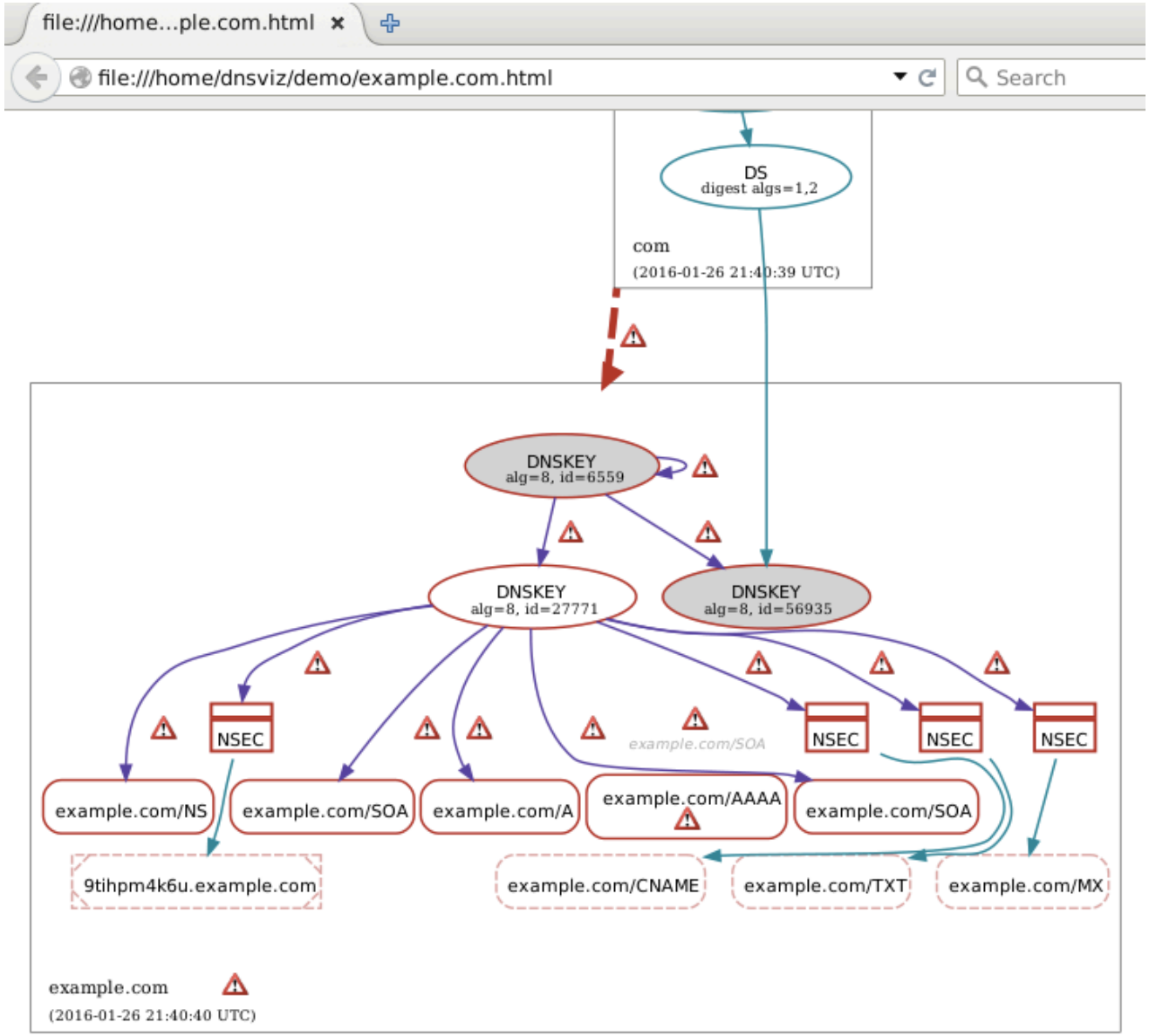
- Reject TCP connection requests

```
# ip6tables -A INPUT -m state --state NEW -p tcp \
      --dport 53 -j REJECT
```

```
$ ./dnsviz_analyze example.com
```

```
$ firefox example.com.html &
```

View `dnsviz graph` Output: No TCP



Modify Path MTU (9.28 – 9.29)

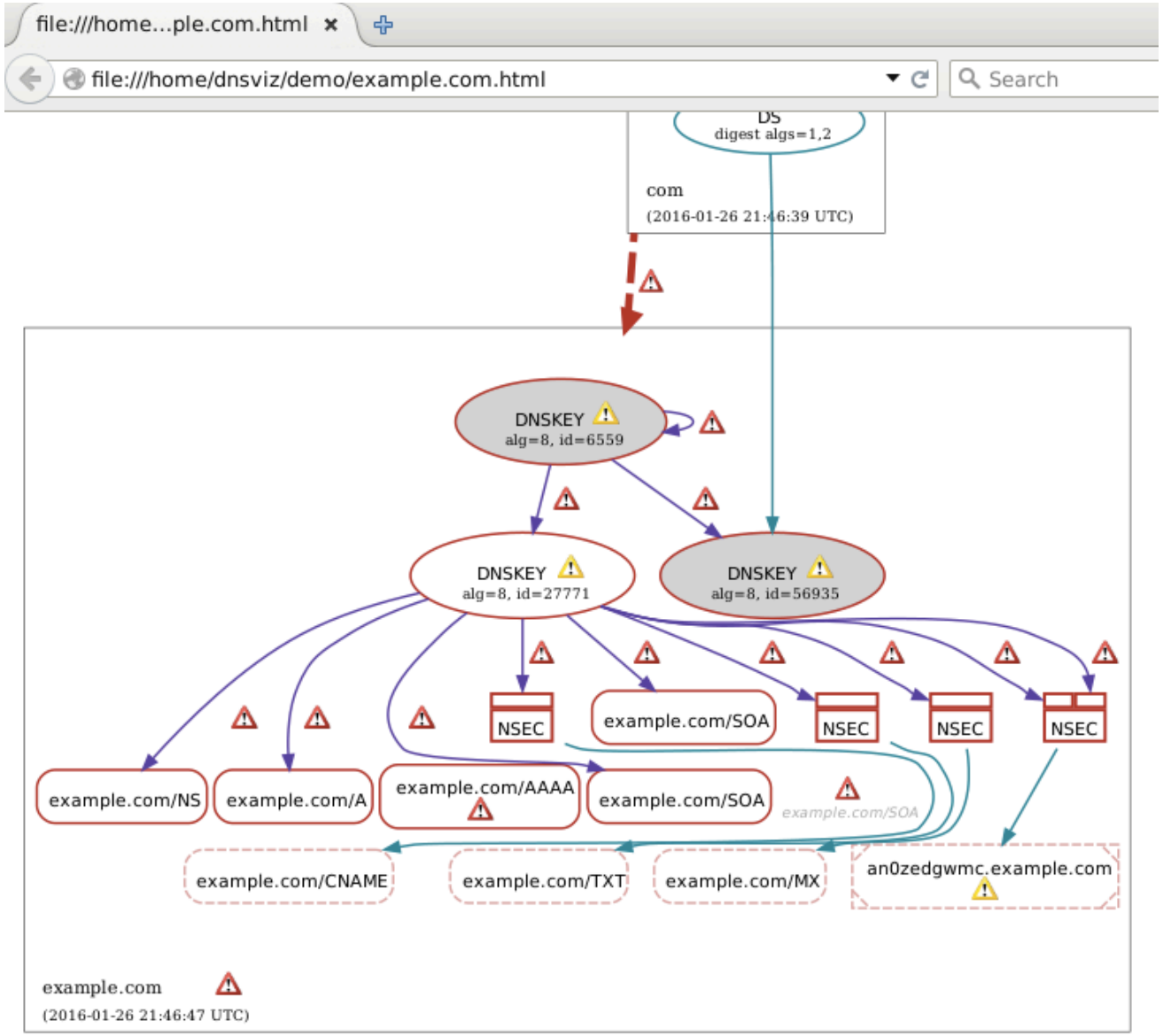
- Drop UDP responses with payloads larger than 512 bytes

```
# iptables -A OUTPUT -p udp --sport 53 \
      -m length --length 540:65535 -j DROP
```

```
$ ./dnsviz_analyze example.com
```

```
$ firefox example.com.html &
```

View `dnsviz graph` Output: Low PMTU



Add Lame Delegation (9.30 – 9.32)

- Add second delegation NS record for example.com in com zone (on **tld1**)

```
# nano db.com
```

or

```
# vi db.com
```

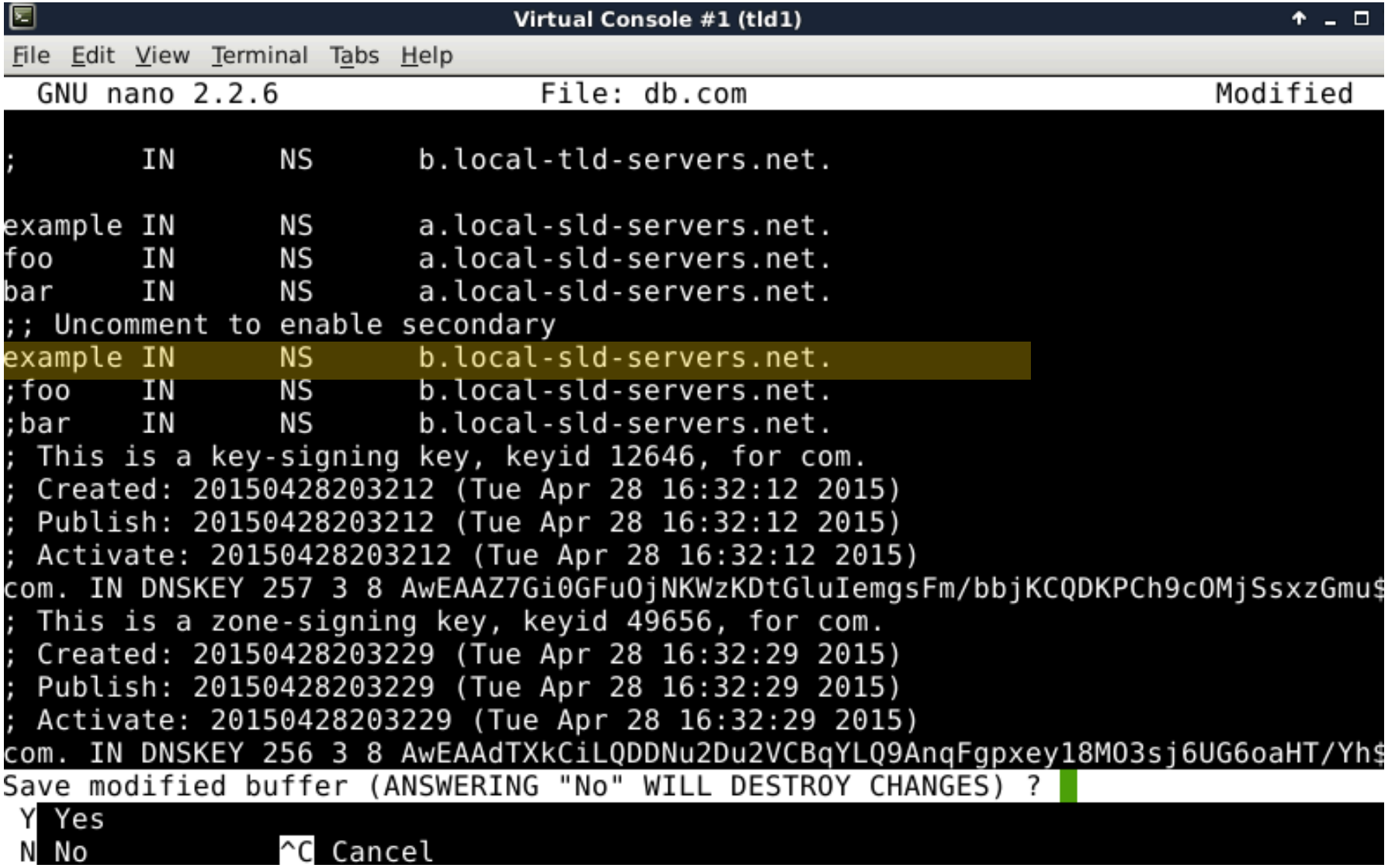
- Sign com zone (on **tld1**)

```
# ./resign_tld
```

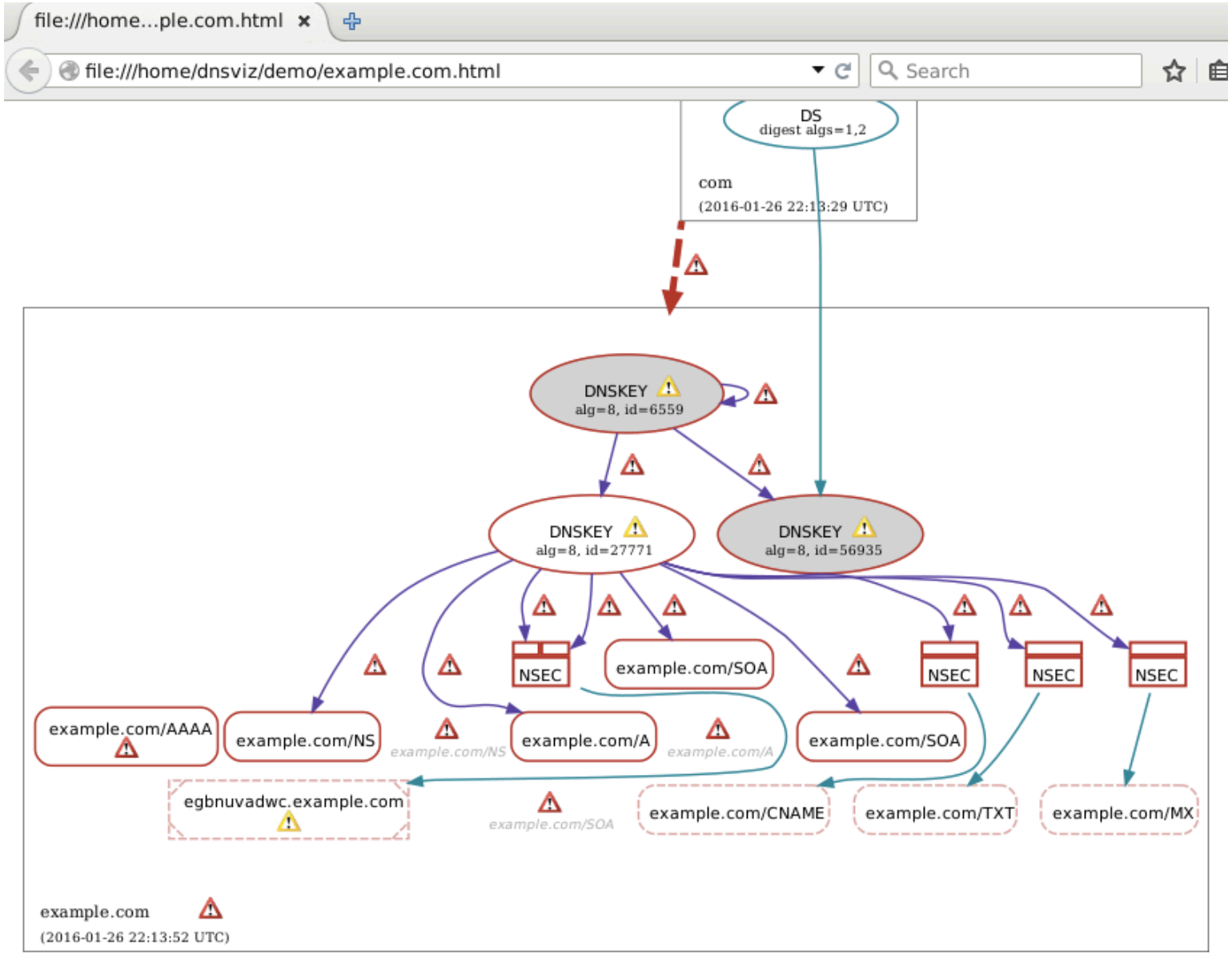
```
$ ./dnsviz_analyze example.com
```

```
$ firefox example.com.html &
```

Add Second NS Record for example.com



View `dnsviz graph` Output: Lame Delegation



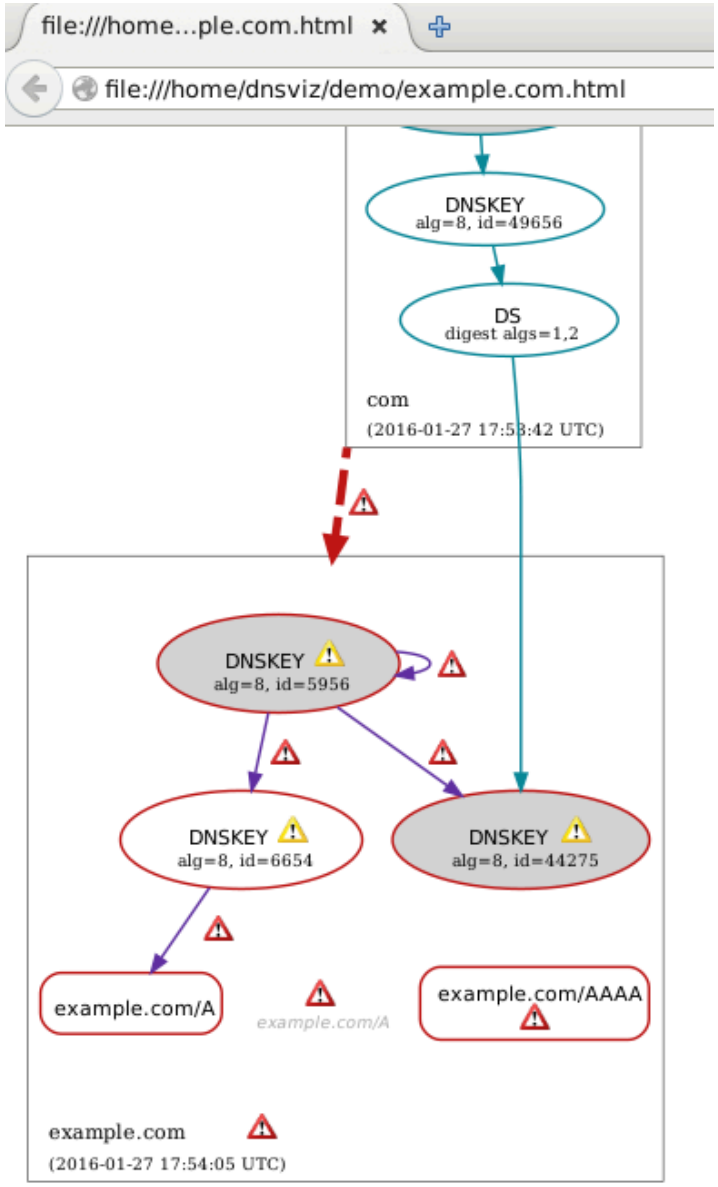
Graph Only Select RRsets (9.33)

Only graph A and AAAA RRsets

```
$ dnsviz graph -R A,AAAA -Thtml -O -t tk-local.txt < \
        example.com-working.json
```

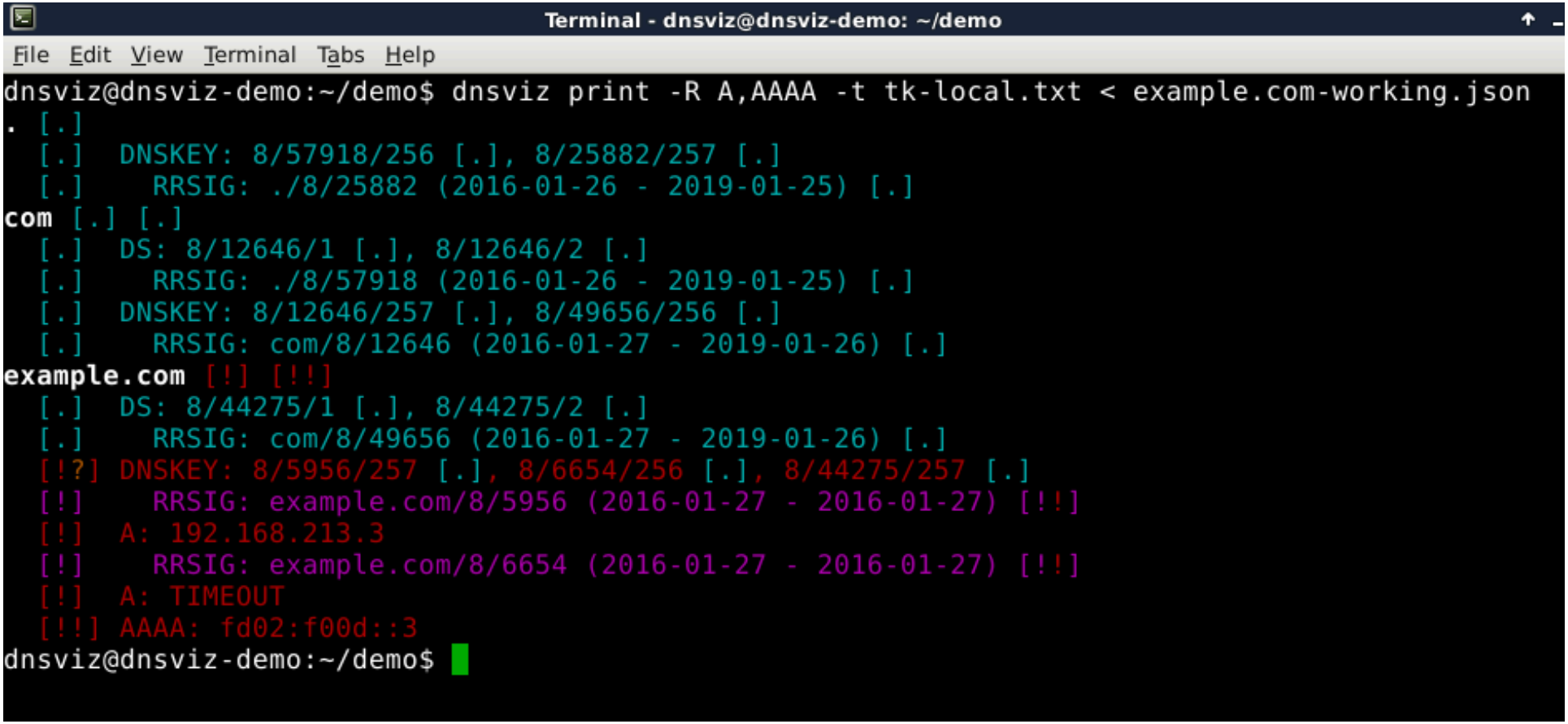
```
$ firefox example.com.html &
```

View `dnsviz graph` Output: Select RRsets

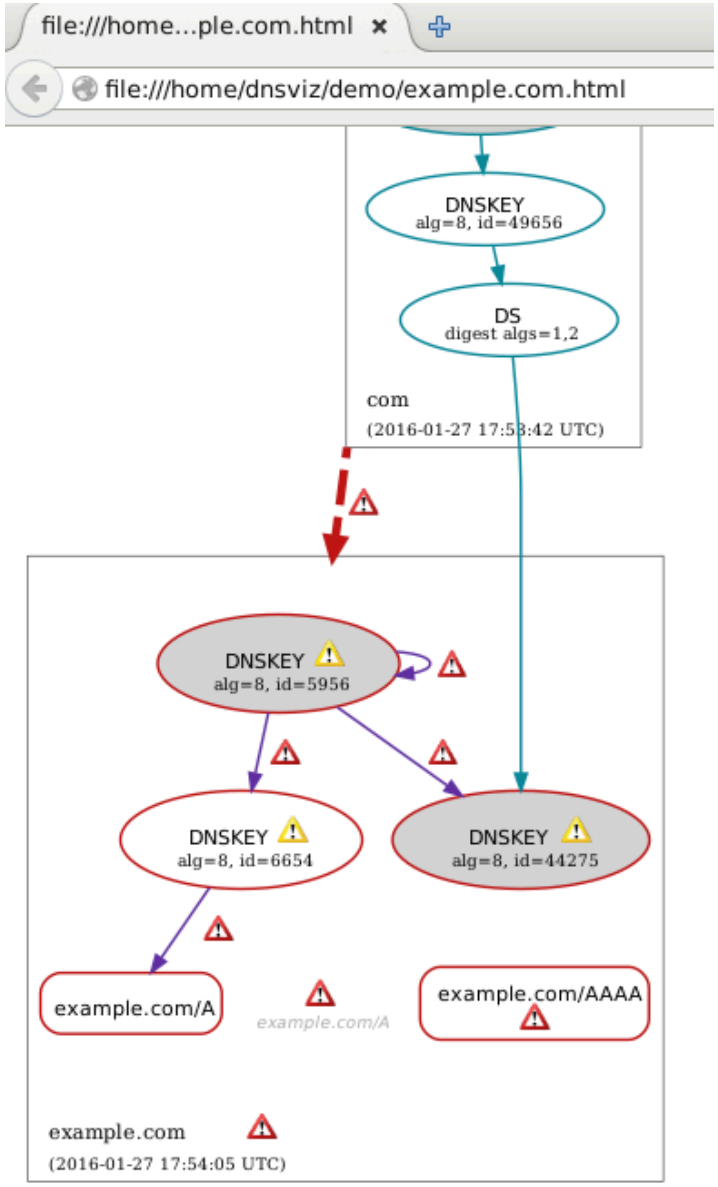


Analyze with `dnsviz print` (9.34)

```
$ dnsviz print -R A,AAAA -t tk-local.txt < \
        example.com-working.json
```



View `dnsviz graph` Output: Select RRsets



DNSViz Recursive Server Analysis

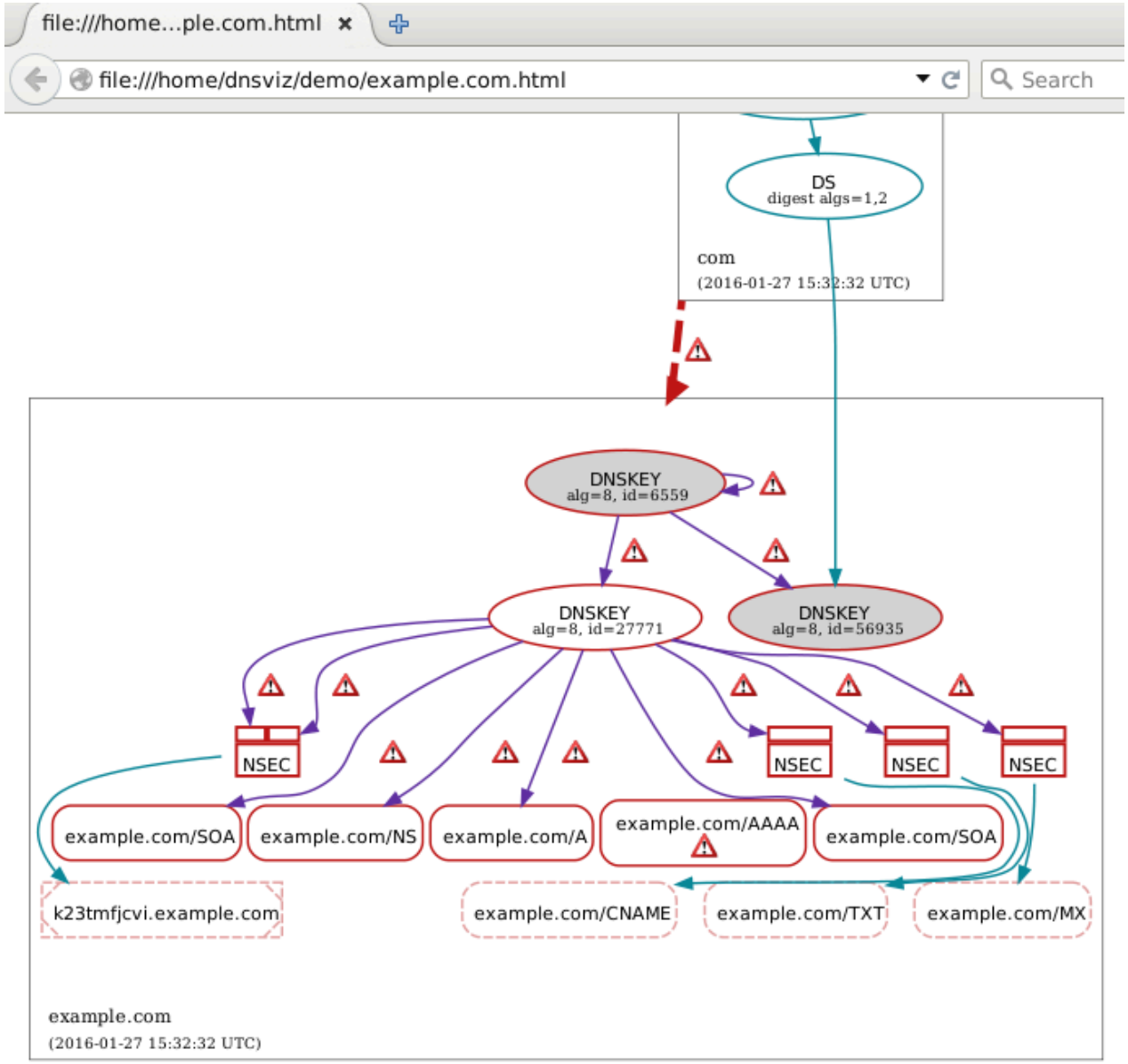
Analyze example.com on Recursive Server (10.1)

No “-A” option
means query
recursive servers

```
$ dnsviz probe example.com | dnsviz graph -Thtml -O -t tk-local.txt
```

```
$ firefox example.com.html &
```

View `dnsviz graph` Output: Recursive



DNSViz Programmatic Analysis

`dnsviz probe` Revisited (11.1)

```
$ medit example.com-working.json &
```

or

```
$ vi example.com-working.json
```

View `dnsviz probe` Output: Diagnostic Query History

example.com-working.json


```
        },
        {
            "qname": "example.com.",
            "qclass": "IN",
            "qtype": "DNSKEY",
            "options": {
                "flags": 0,
                "edns_version": 0,
                "edns_max_udp_payload": 4096,
                "edns_flags": 32768,
                "edns_options": [],
                "tcp": false
            },
            "responses": {
                "192.168.213.26": {
                    "192.168.213.1": {
                        "message": "Xh6EAAABAAQAAAABB2V4YW1wbGUDY29tAAAwAAHADAAwAAEAAAA8A
                        "msg_size": 1039,
                        "time_elapsed": 1,
                        "history": [
                            {
                                "time_elapsed": 1000,
                                "cause": "TIMEOUT",
                                "action": "NO_CHANGE"
                            },
                            {
                                "time_elapsed": 1000,
                                "cause": "TIMEOUT",
                                "action": "NO_CHANGE"
                            },
                            {
                                "time_elapsed": 2000
```

View `dnsviz probe` Output: Diagnostic Query History

example.com-working.json


```
"msg_size": 1039,
"time_elapsed": 1,
"history": [
    {
        "time_elapsed": 1000,
        "cause": "TIMEOUT",
        "action": "NO_CHANGE"
    },
    {
        "time_elapsed": 1000,
        "cause": "TIMEOUT",
        "action": "NO_CHANGE"
    },
    {
        "time_elapsed": 2000,
        "cause": "TIMEOUT",
        "action": "NO_CHANGE"
    },
    {
        "time_elapsed": 4003,
        "cause": "TIMEOUT",
        "action": "CHANGE_UDP_MAX_PAYLOAD",
        "action_arg": 512
    },
    {
        "time_elapsed": 0,
        "cause": "TC",
        "cause_arg": 40,
        "action": "USE_TCP"
    },
```

`dnsviz grok` Revisited (10.3 – 10.4)

```
$ dnsviz grok -l warning -p < example.com-broken.json \
        > example.com-working-p.json
```

```
$ medit example.com-working-p.json &
```

or

```
$ vi example.com-working-p.json
```

View `dnsviz grok` Output: Errors, Warnings, Statuses

example.com-working-p.json


```
            "example.com./IN/A": {
                "answer": [
                    {
                        "id": "example.com./IN/A",
                        "rrsig": [
                            {
                                "id": "example.com./8/6654",
                                "status": "EXPIRED",
                                "errors": [
                                    {
                                        "description": "The Signature Expiration field of the RRSIG RR (201
                                        "code": "EXPIRATION_IN_PAST"
                                    }
                                ]
                            }
                        ]
                    }
                ],
                "error": [
                    {
                        "description": "No response was received from the server over UDP (tried 8 times)."
                        "code": "TIMEOUT",
                        "servers": [
                            "192.168.213.27",
                            "fd02:f00d:18::27"
                        ],
                        "query_options": [
                            "UDP_0_NOEDNS"
                        ]
                    }
                ]
```

View `dnsviz grok` Output: Errors, Warnings, Statuses

example.com-working-p.json


```
        "example.com./IN/SOA": {
            "answer": [
                {
                    "id": "example.com./IN/SOA",
                    "rrsig": [
                        {
                            "id": "example.com./8/6654",
                            "status": "EXPIRED",
                            "errors": [
                                {
                                    "description": "The Signature Expiration field of the RRSIG RR (2016-
                                    "code": "EXPIRATION_IN_PAST"
                                },
                                {
                                    "description": "The cryptographic signature of the RRSIG RR does not
                                    "code": "SIGNATURE_INVALID"
                                }
                            ]
                        }
                    ]
                }
            ],
            "error": [
                {
                    "description": "The TCP connection was refused (ECONNREFUSED).",
                    "code": "NETWORK_ERROR",
                    "servers": [
                        "fd02:f00d:18::26"
                    ],
                    "query_options": [
                        "TCP 0 EDNS0 32768 4096"
```

View `dnsviz grok` Output: Errors, Warnings, Statuses

example.com-working-p.json


```
        },
        "delegation": {
            "status": "BOGUS",
            "errors": [
                {
                    "description": "No valid RRSIGs made by a key corresponding to a DS RR were found coveri
                    "code": "NO_SEP",
                    "servers": [
                        "192.168.213.26",
                        "fd02:f00d:18::26"
                    ],
                    "query_options": [
                        "TCP_0_EDNS0_32768_1038",
                        "UDP_0_EDNS0_32768_4096"
                    ]
                },
                {
                    "description": "The DS RRset for the zone included algorithm 8 (RSASHA256), but no DS RR
                    "code": "MISSING_SEP_FOR_ALG",
                    "servers": [
                        "192.168.213.26",
                        "fd02:f00d:18::26"
                    ],
                    "query_options": [
                        "TCP_0_EDNS0_32768_1038",
                        "UDP_0_EDNS0_32768_4096"
                    ]
                }
            ]
        }
    }
```

Monitoring with DNSViz

- Sample script uses combination of dnsviz get and dnsviz graph, e.g., for use with cron

```
#!/bin/sh
name=$1
date=`date +%Y%m%d%H%M%S`
probe_out=/tmp/$name-probe-$date.json
grok_out=/tmp/$name-grok-$date.json
graph_out=/tmp/$name-graph-$date.png

dnsviz probe -A -d 0 -p $name > $probe_out
dnsviz grok -l warning -p $name < $probe_out > $grok_out
if (( $( stat -c %s $grok_out ) > 0 )); then
  dnsviz graph -Tpng -o $graph_out $name $name < $probe_out
  gzip $probe_out
  cat $grok_out | \
  mutt -s “Problems with $name” -a $graph_out $grok_out.gz -- \
    joe@example.com
fi

rm $probe_out* $grok_out $graph_out
```

Summary

- Understanding and analyzing DNS and DNSSEC can be complex.
- DiG, BIND, DNSViz, and other tools can aid in understanding, troubleshooting, and monitoring.
- Maintain and monitor your DNS zones!

Further Information on DNSViz

- Source: https://github.com/dnsviz/dnsviz (License: GPLv2)
- Online version: http://dnsviz.net/
- Mailing list: https://groups.google.com/d/forum/dnsviz-users

powered by

VERISIGN™