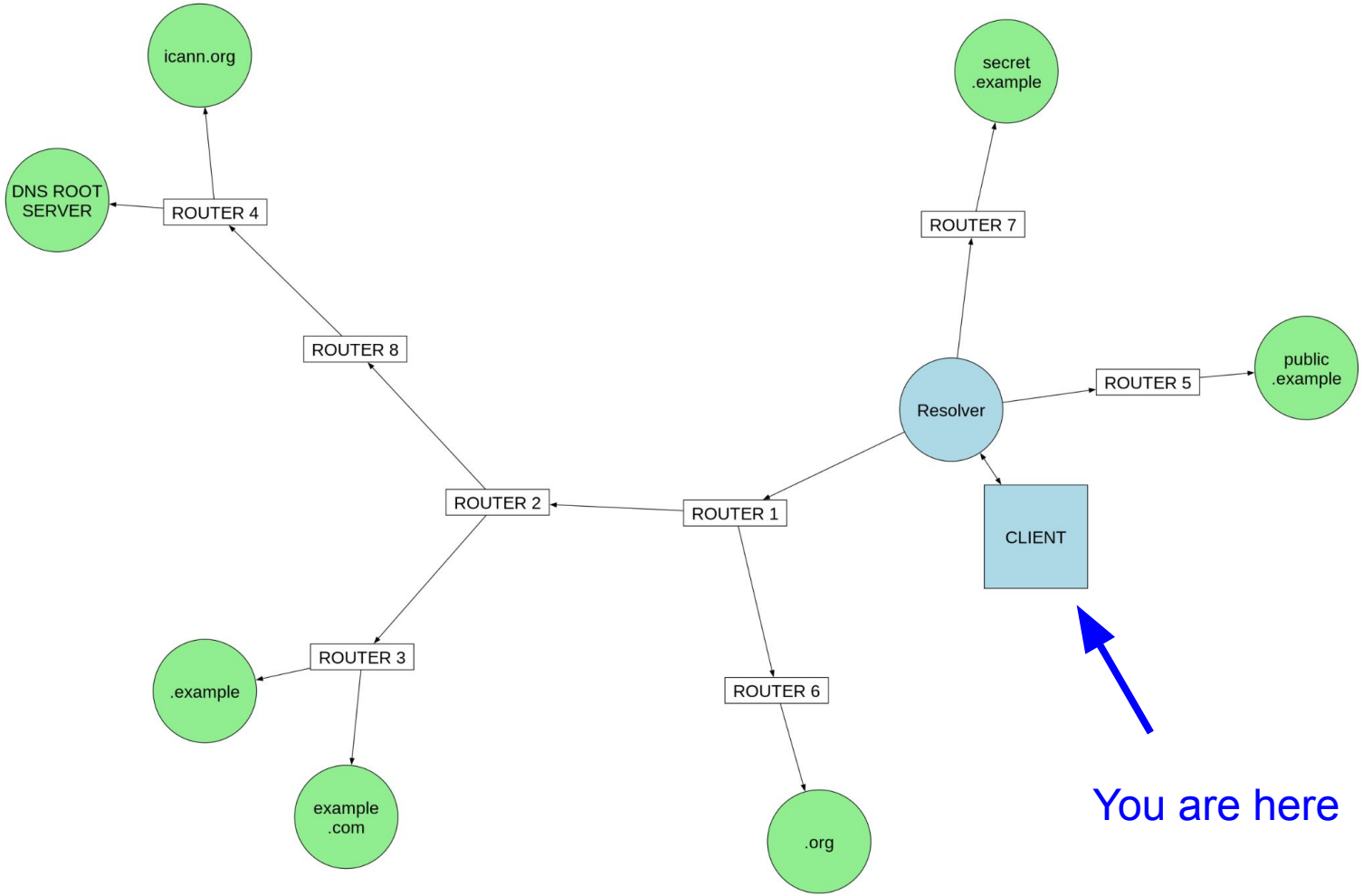
Injecting a MiM In front of a Validating Resolver

Wes Hardaker <ietf@hardakers.net>

Imagine Your Network's DNS

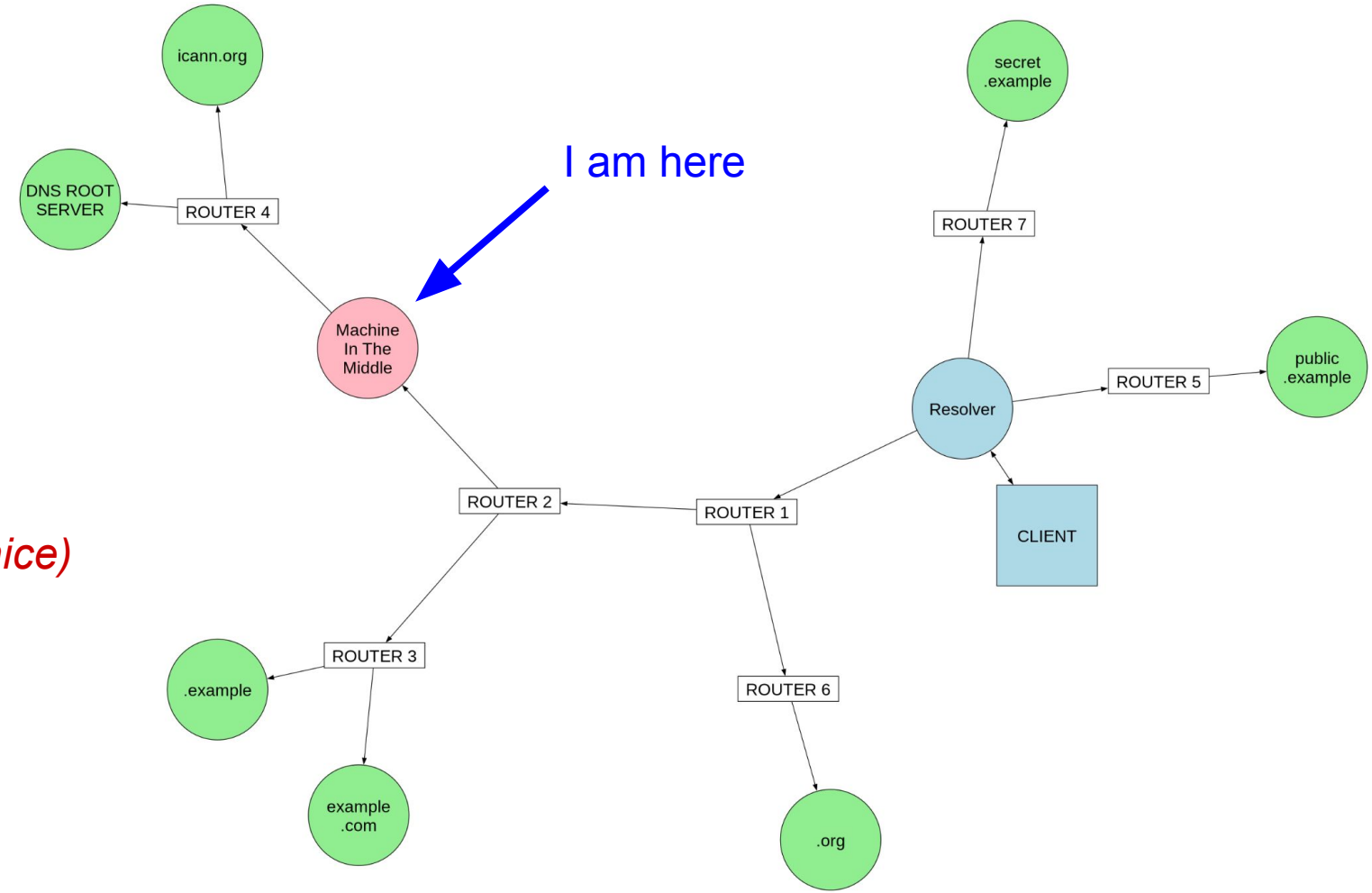
and

The Internet



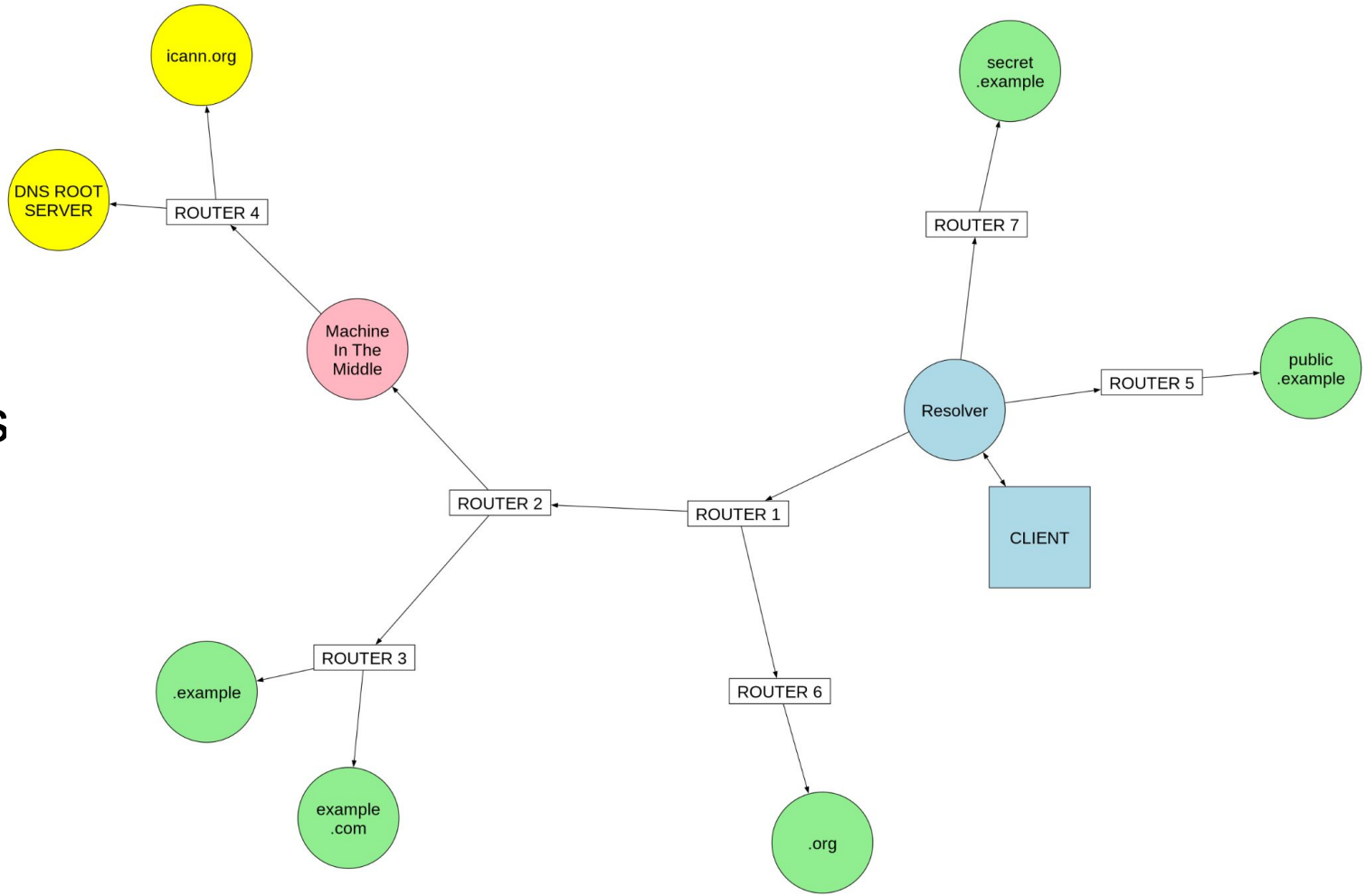
Imagine
I'm
In
Your
Network

(and I'm not nice)



I
Can
View
All Your
Resolver's
DNS
Requests

To These
Zones

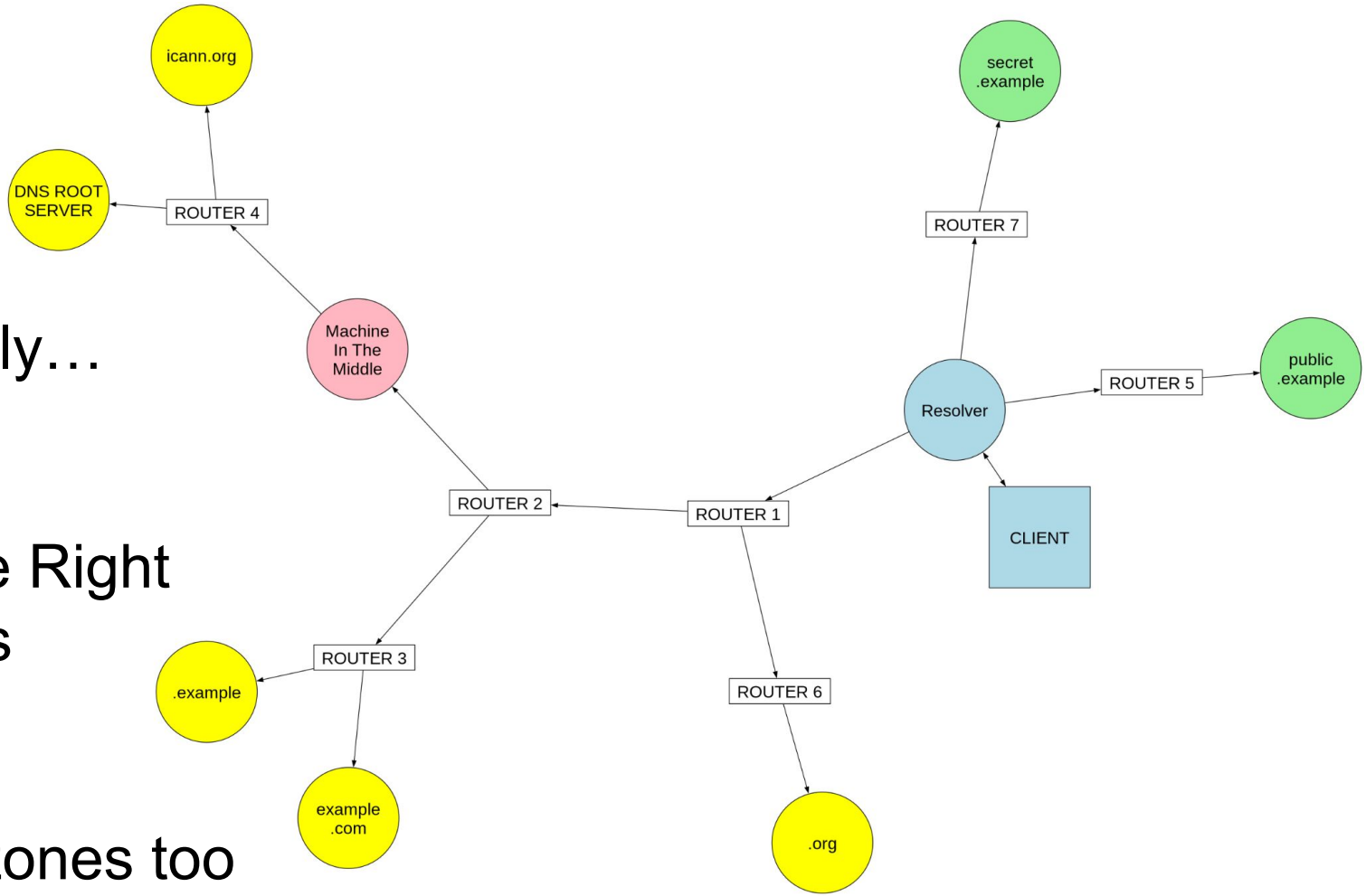


And

Surprisingly…

Under The Right Conditions

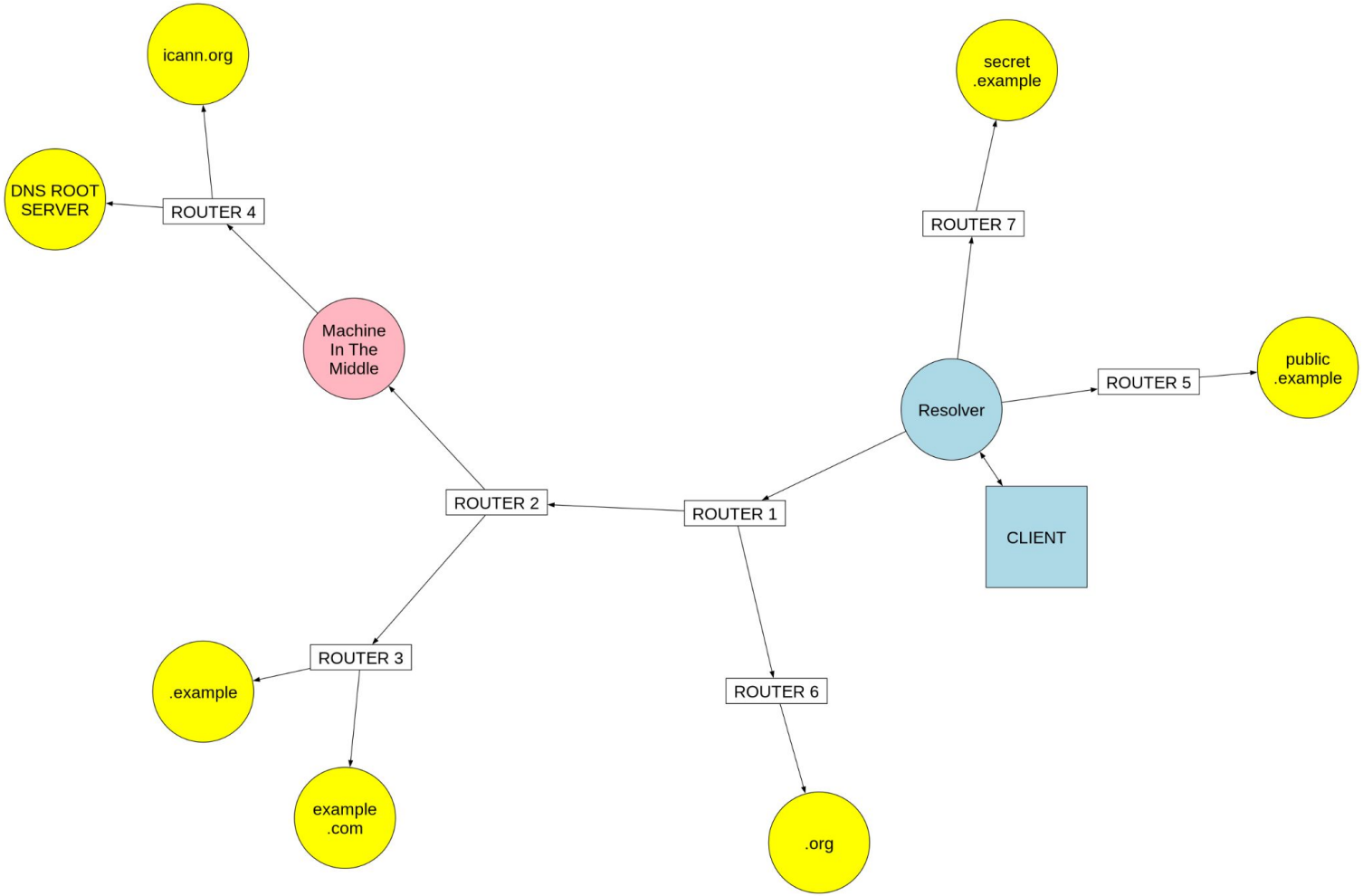
To these zones too



And

Even

These



Overview

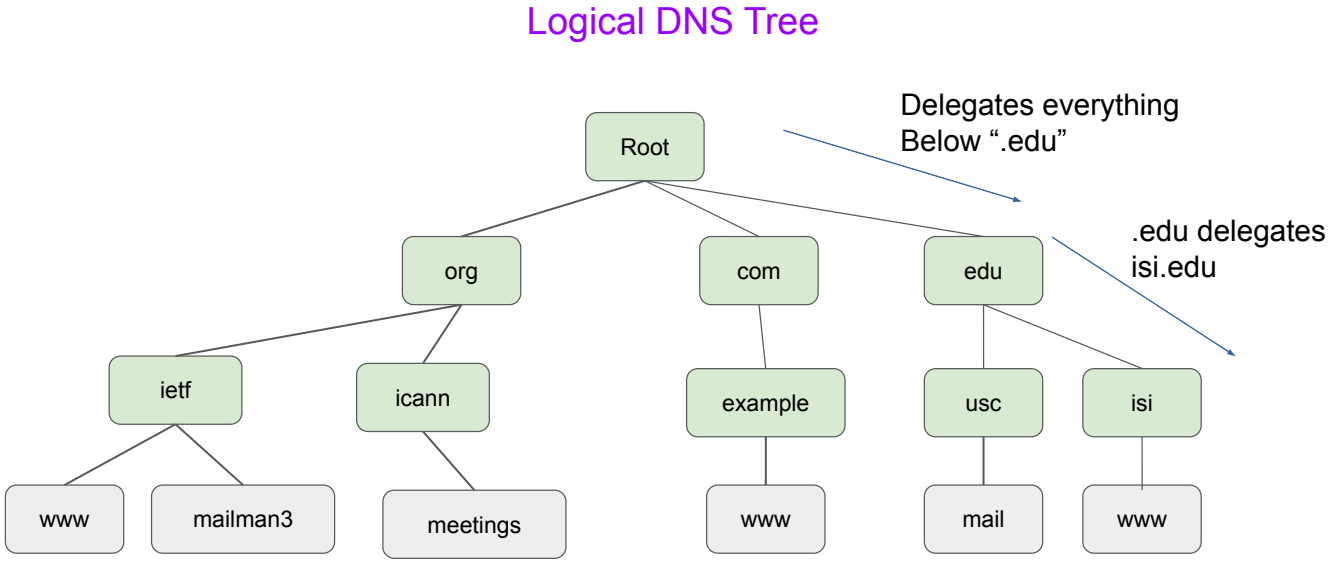
1. The Issue: Parent-centric resolvers trust unvalidated data

2. Attack demonstration

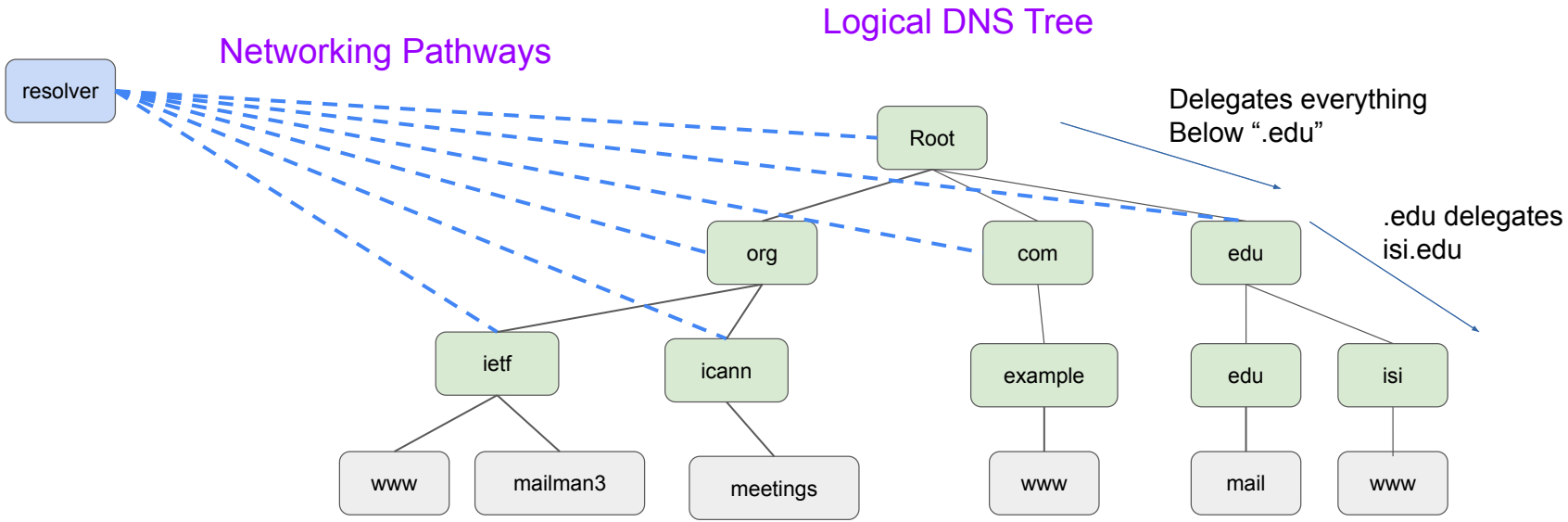
3. So what? What's the worst that could happen…

4. Mitigations

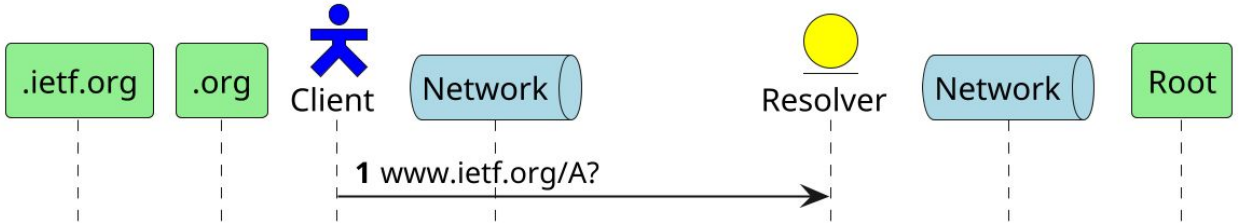
DNS Resolvers Communicate With Everything



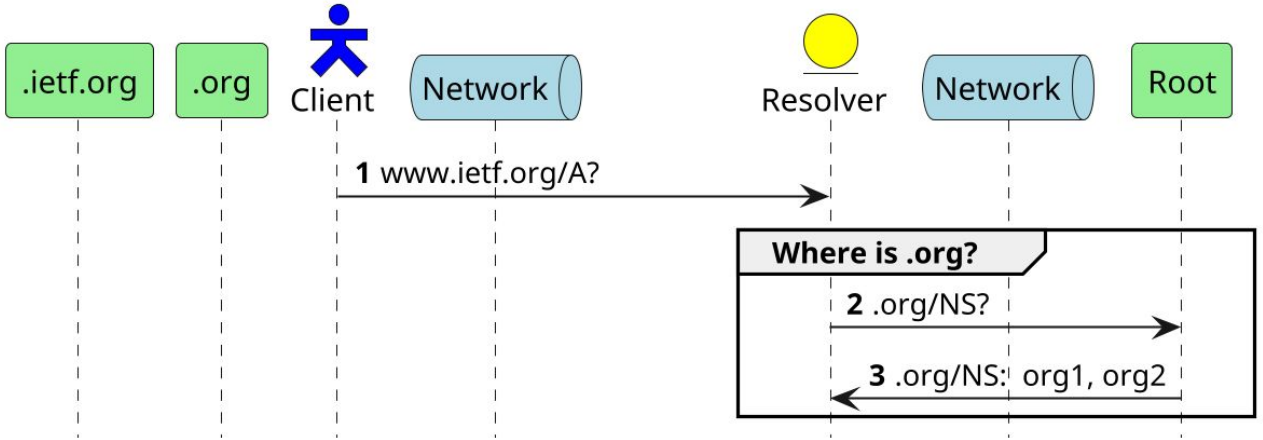
DNS Resolvers Communicate With Everything



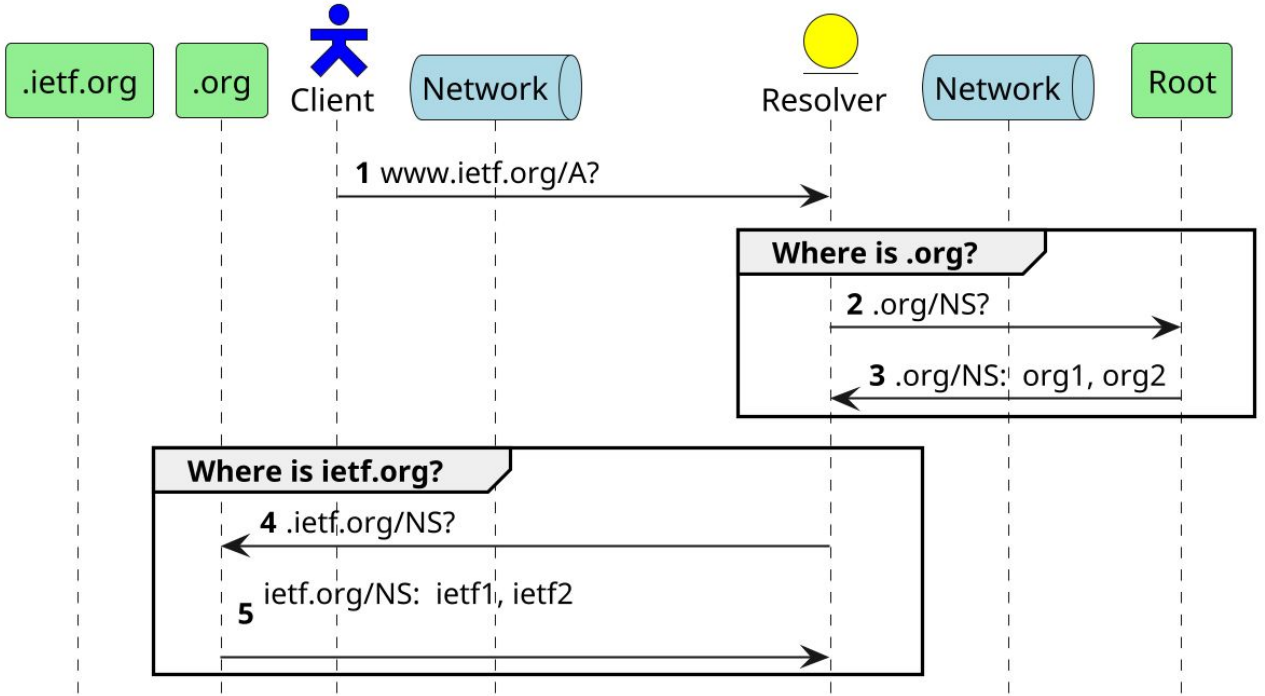
DNS Resolution Steps



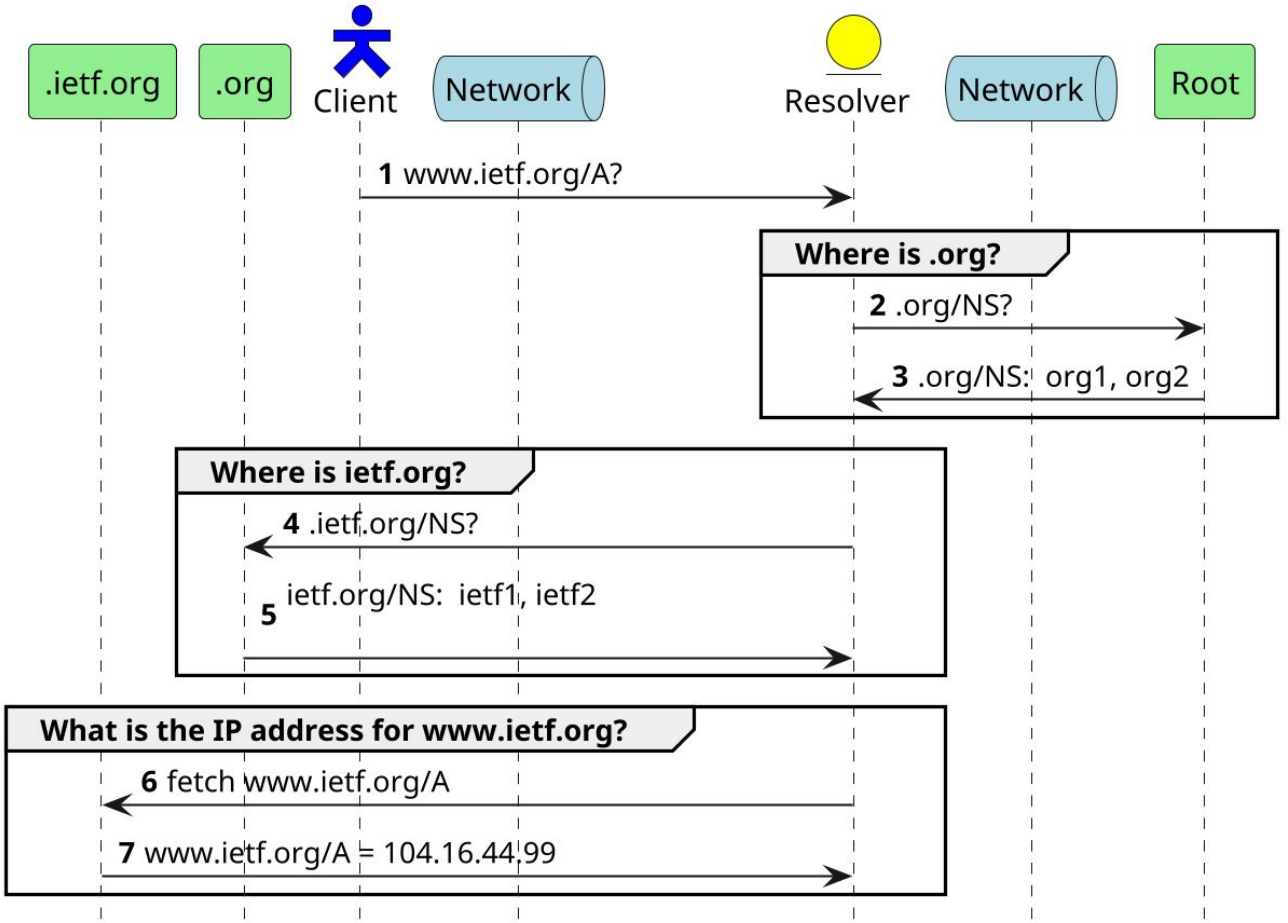
DNS Resolution Steps



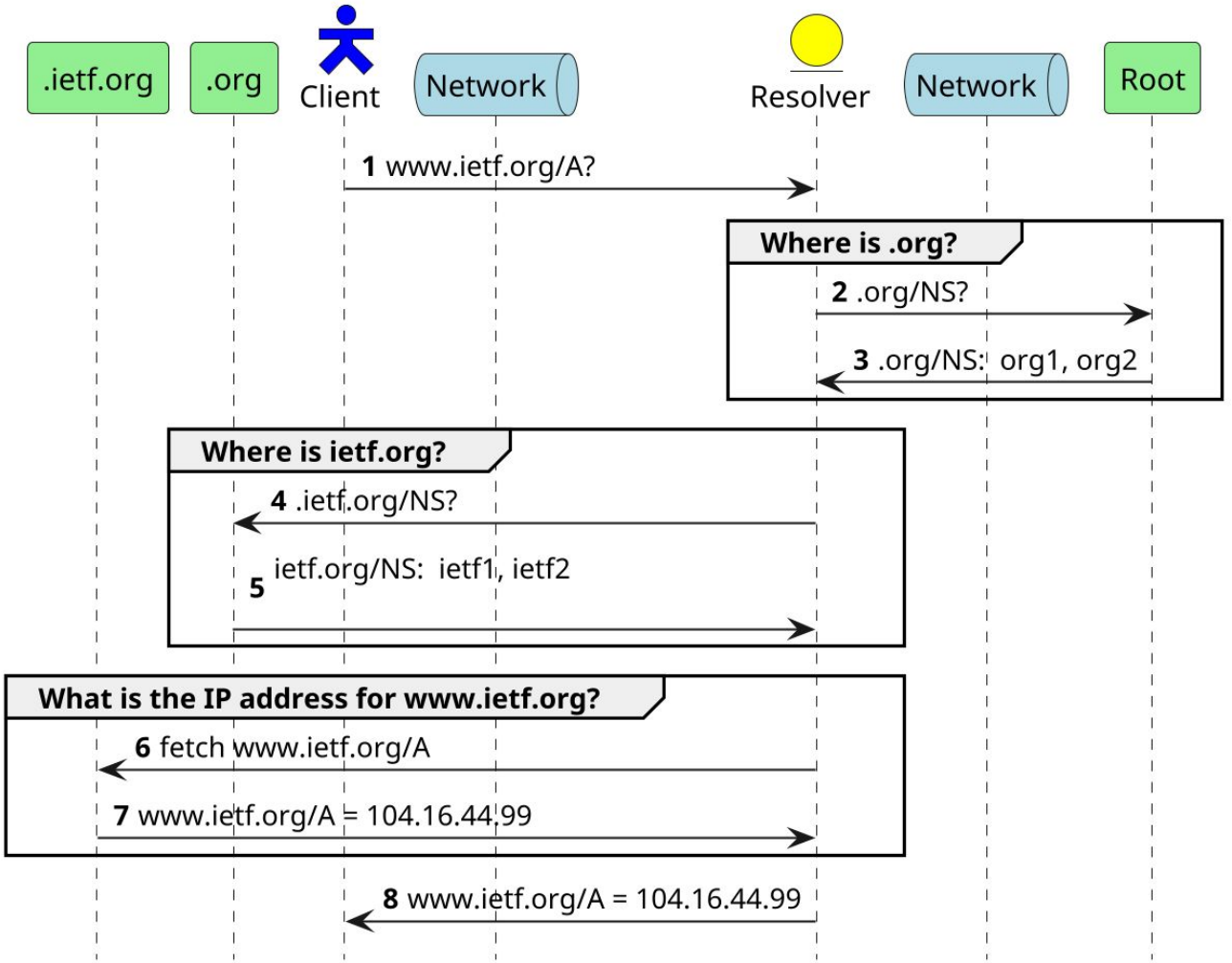
DNS Resolution Steps



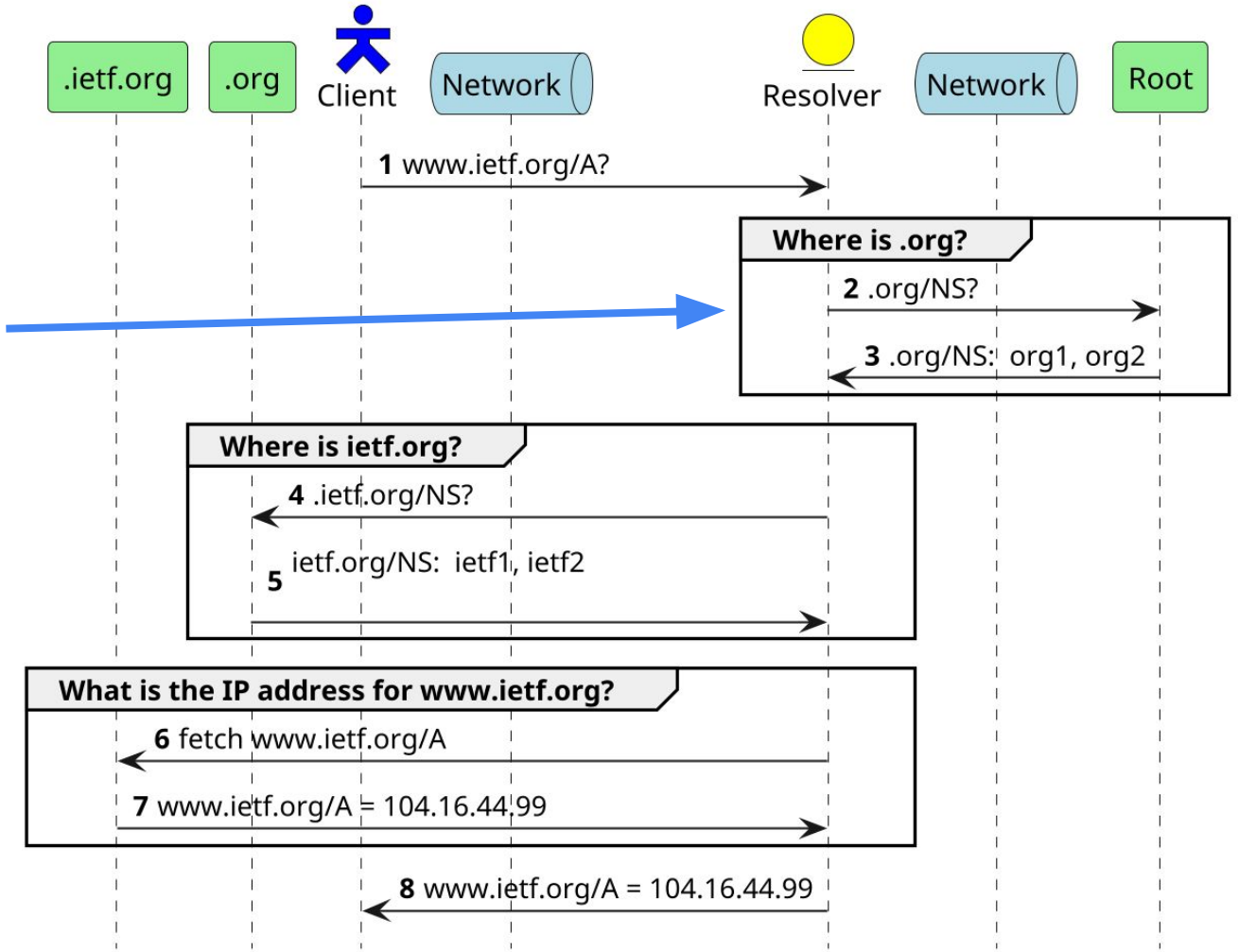
DNS Resolution Steps



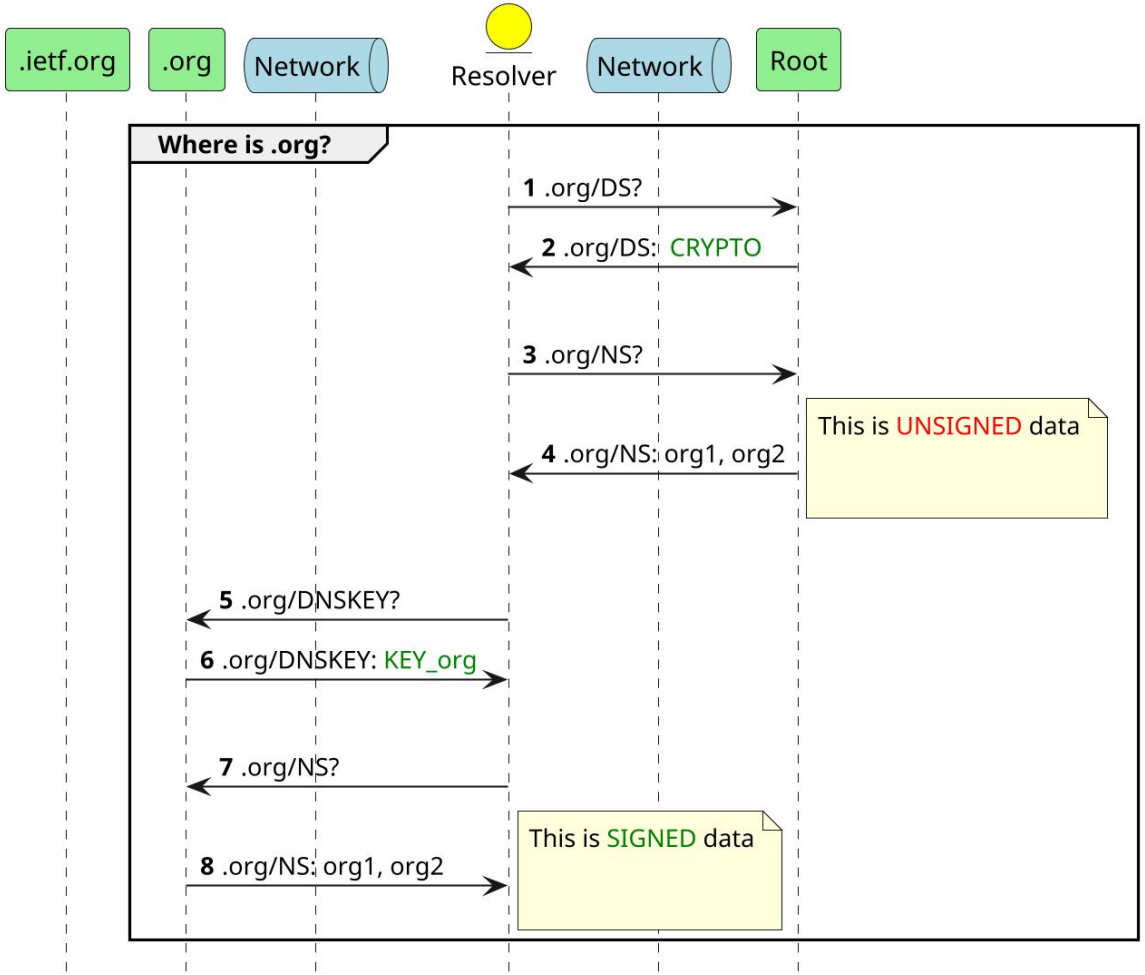
DNS Resolution Steps



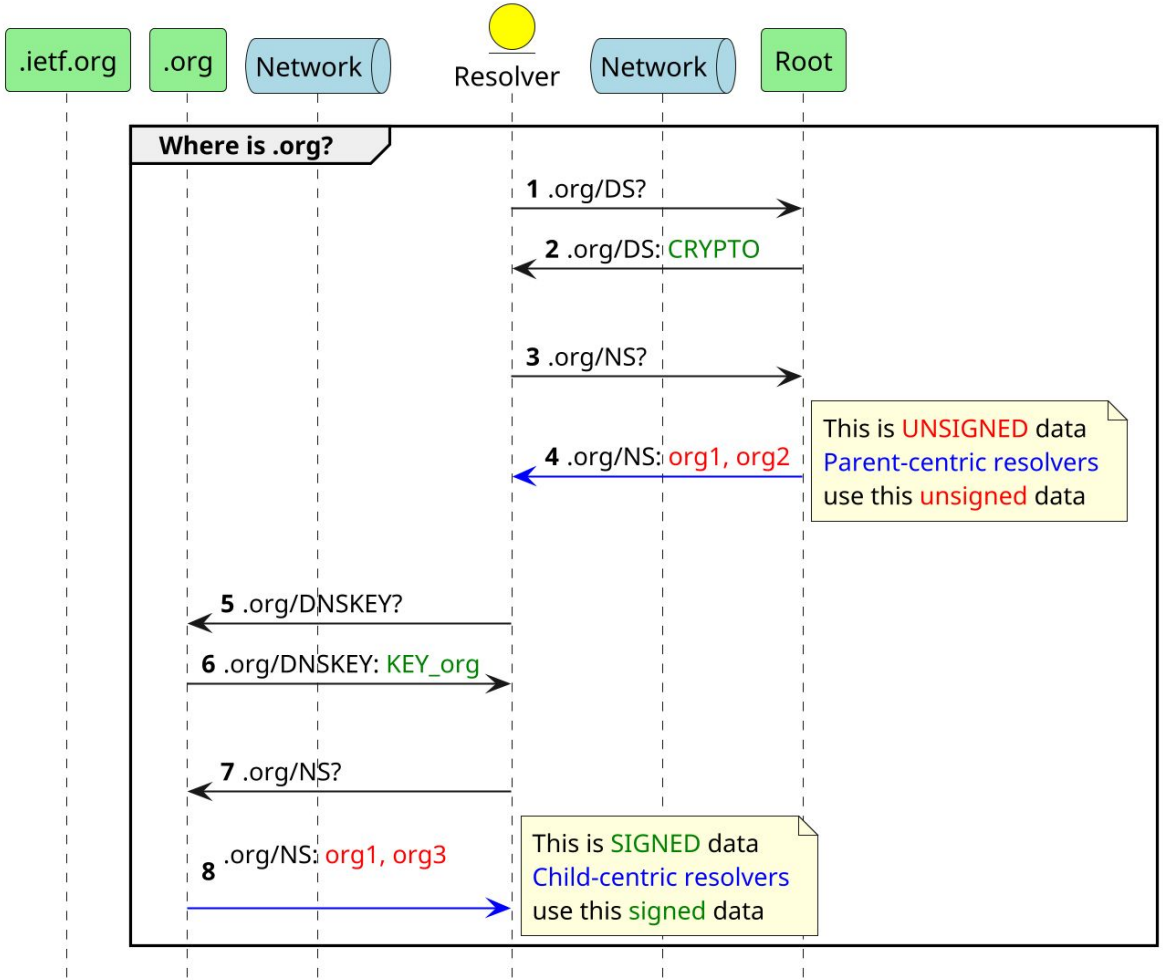
Let's dive into just this transaction with DNSSEC validation



DNSSEC Validation of .org data



What Happens When Differences Arise?



Child-centric vs Parent-centric Resolvers

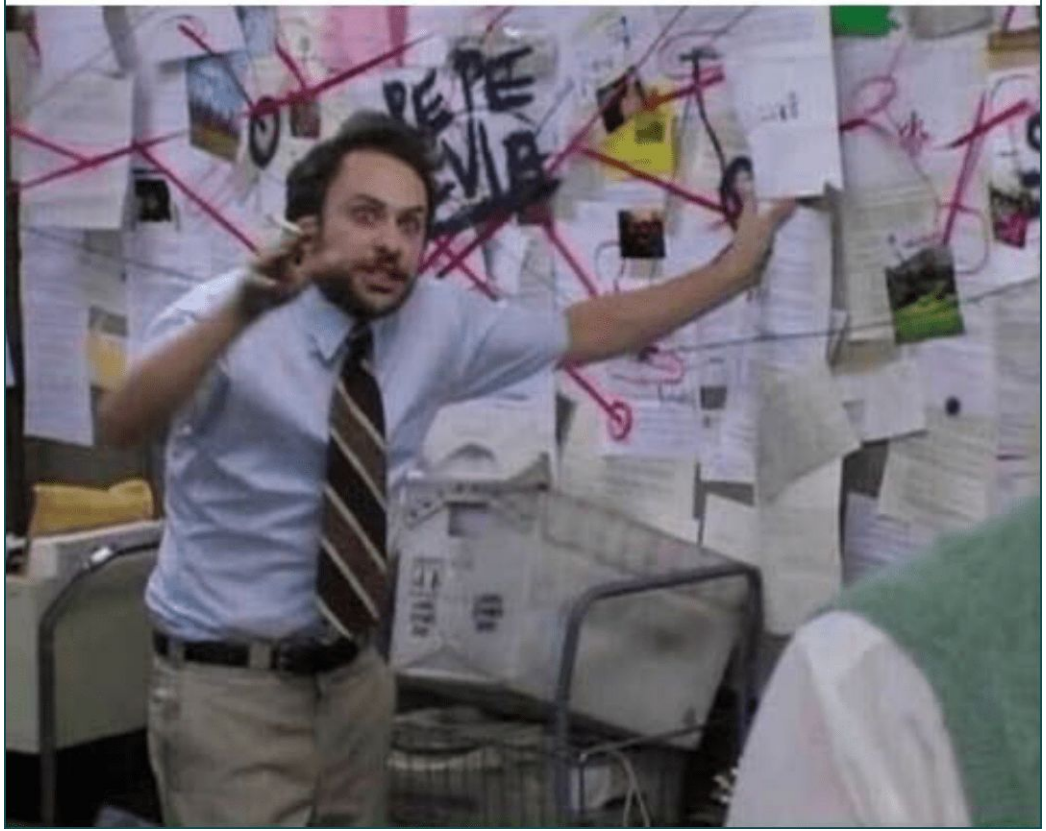
Child-centric resolvers

- Has only **one cache**: stores all data
- Double checks child zone information
- **All** data is DNSSEC validated
- Are **slower**

Parent-centric resolvers

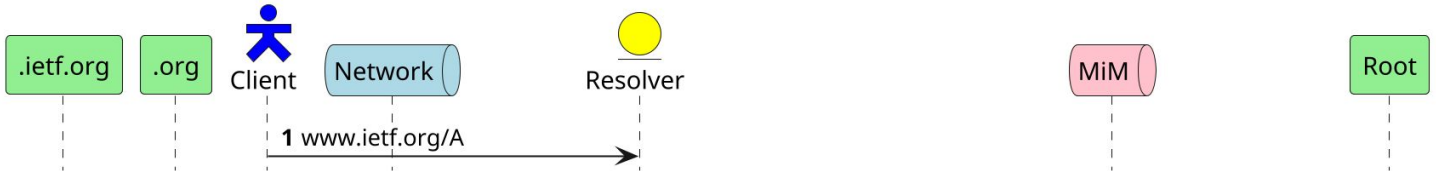
- Has **two caches**: child data and delegation
- Believes the parent's delegation information
- **Only client data** is DNSSEC validated
- Are **faster** and more resilient in some cases

"what's the worst thing that could happen?"

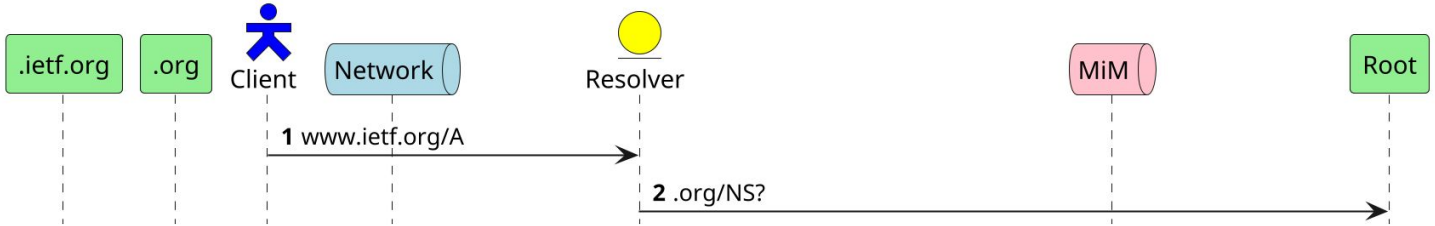



Disney
the PARENT TRAP
© 2019 Disney

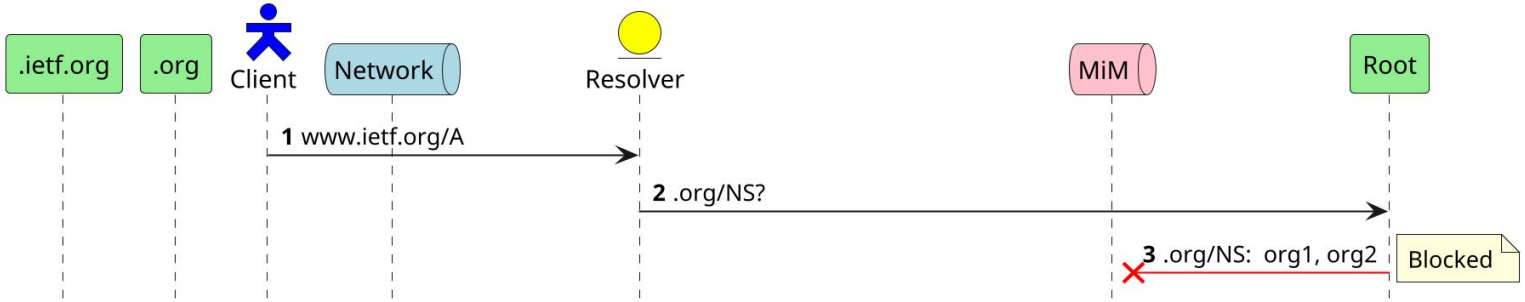
The Parent Trap



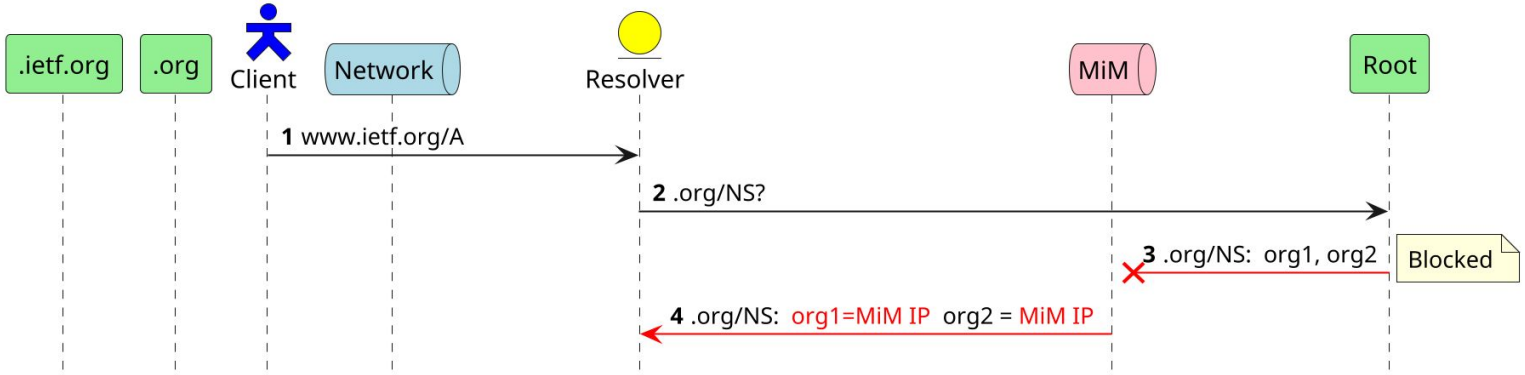
The Parent Trap



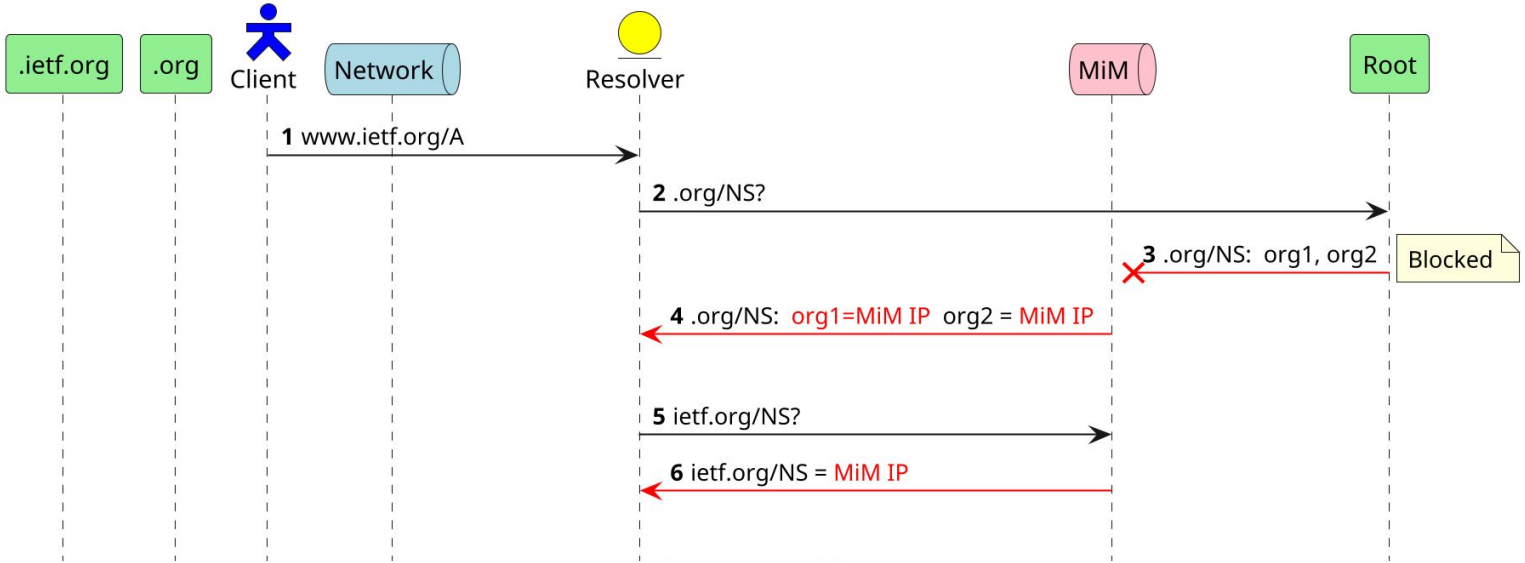
The Parent Trap



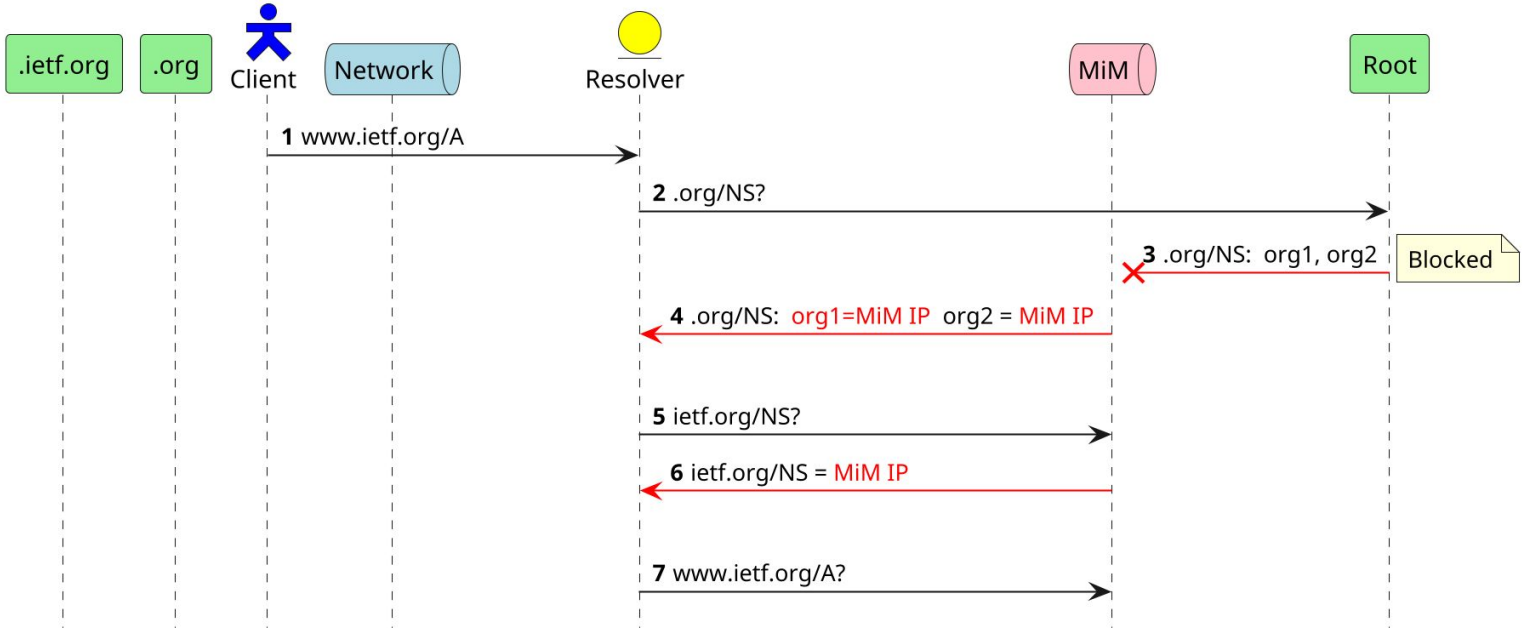
The Parent Trap



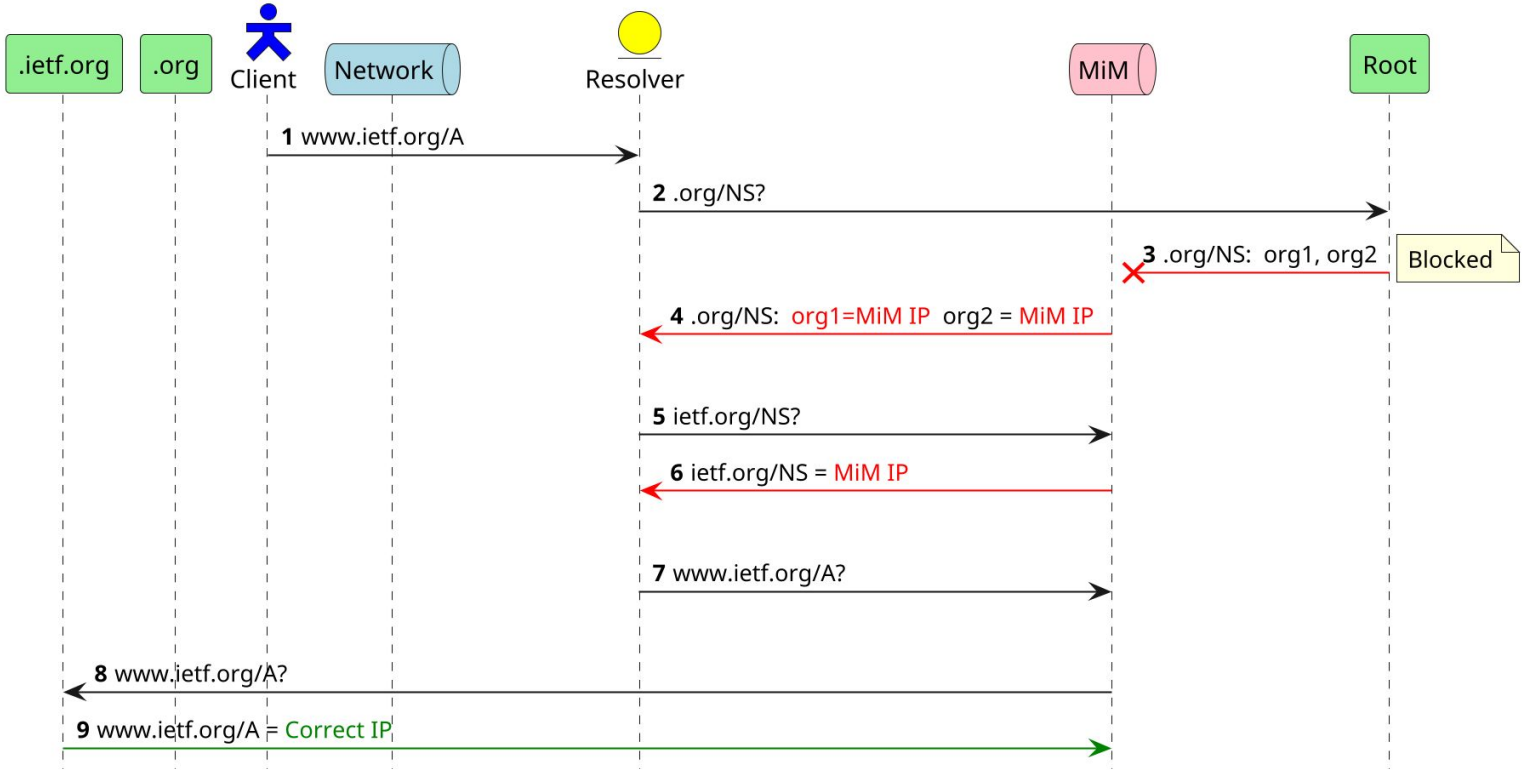
The Parent Trap



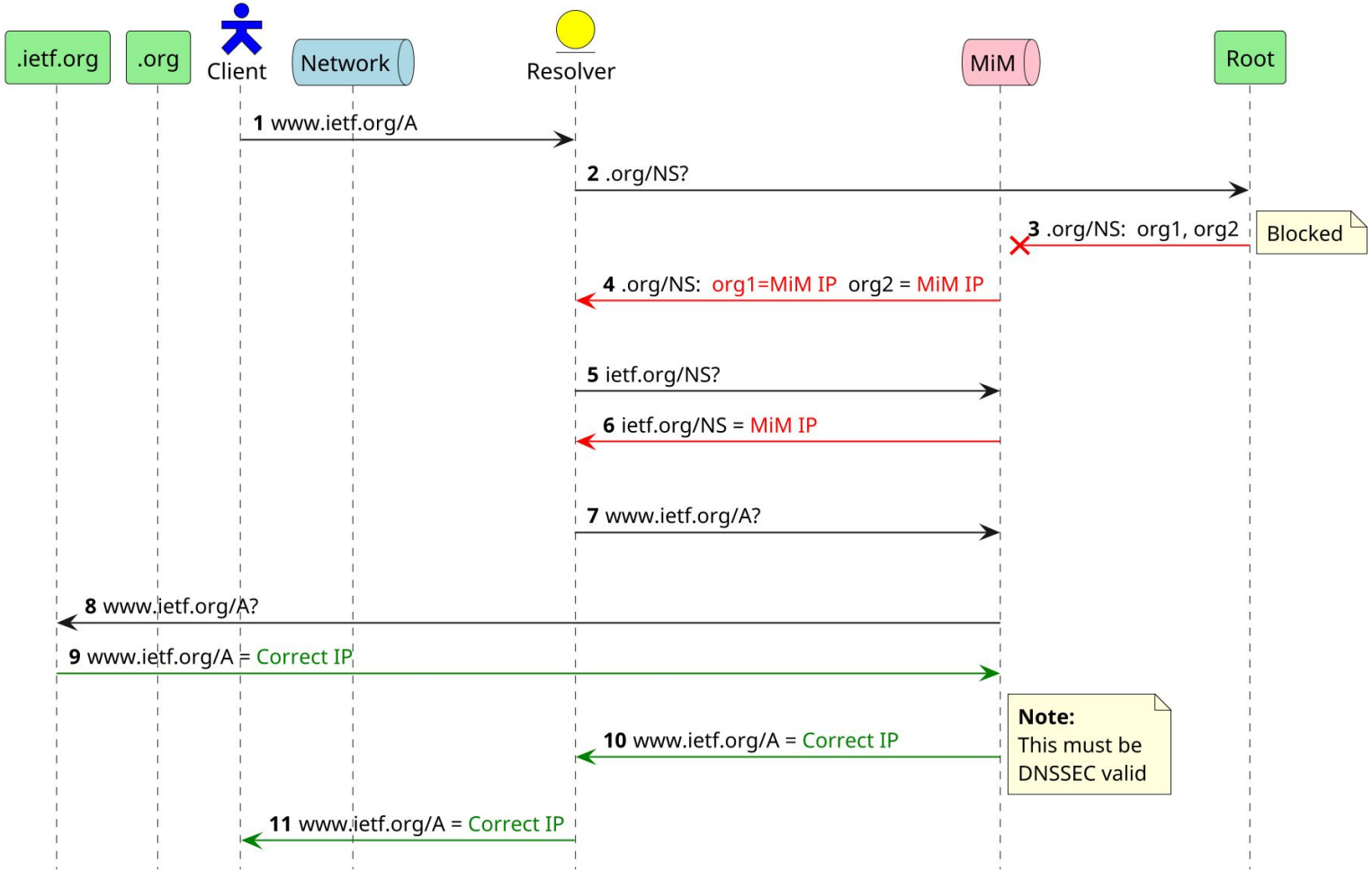
The Parent Trap

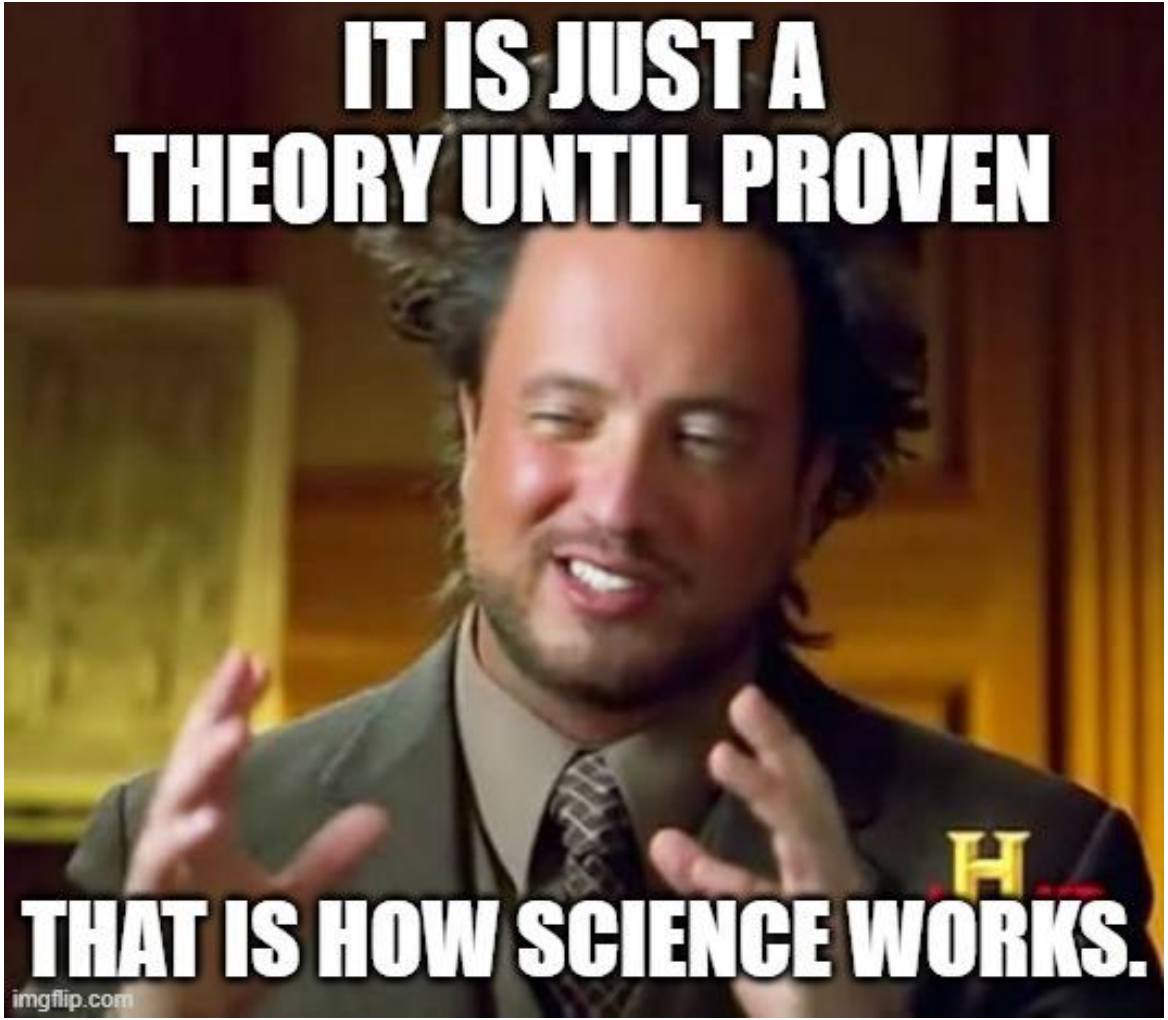


The Parent Trap



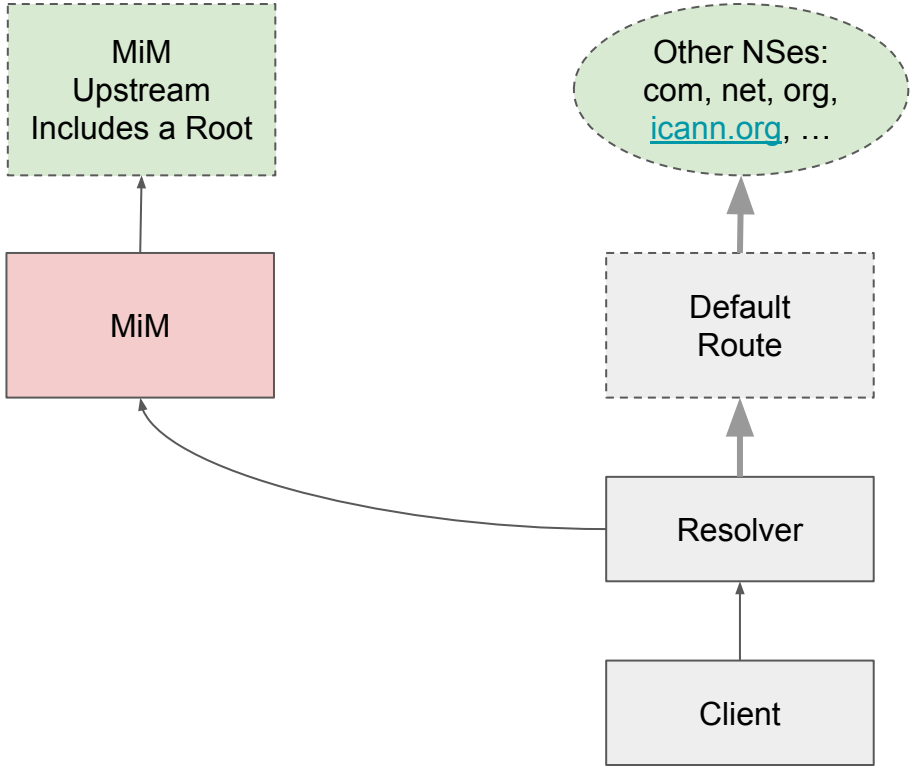
The Parent Trap



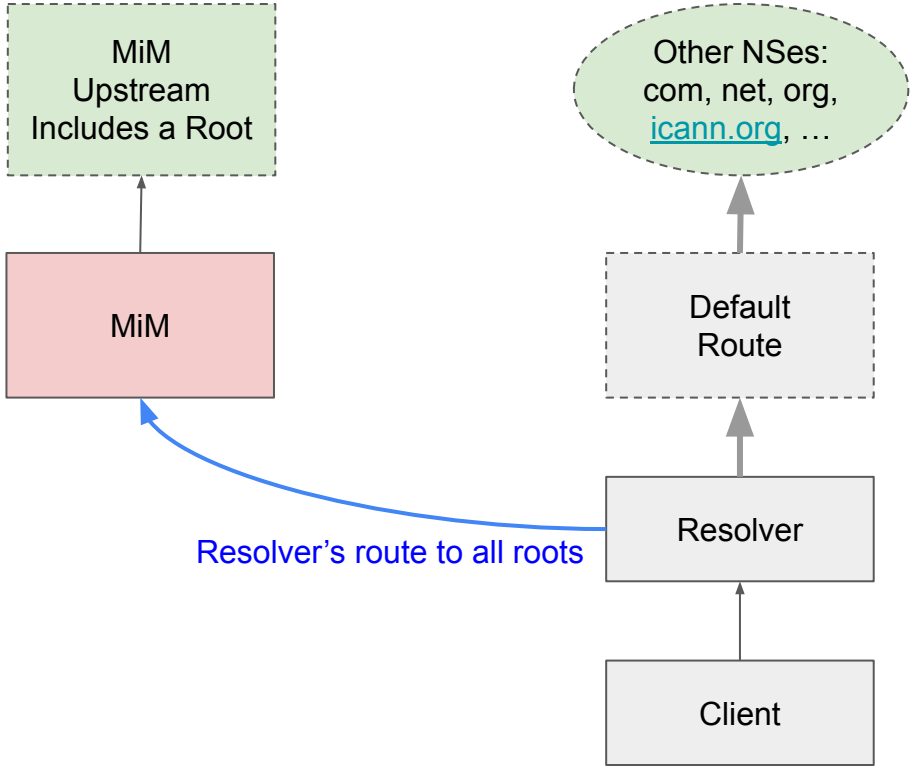

IT IS JUST A
THEORY UNTIL PROVEN
THAT IS HOW SCIENCE WORKS.
imgflip.com

Demonstration Setup

Test Network Design



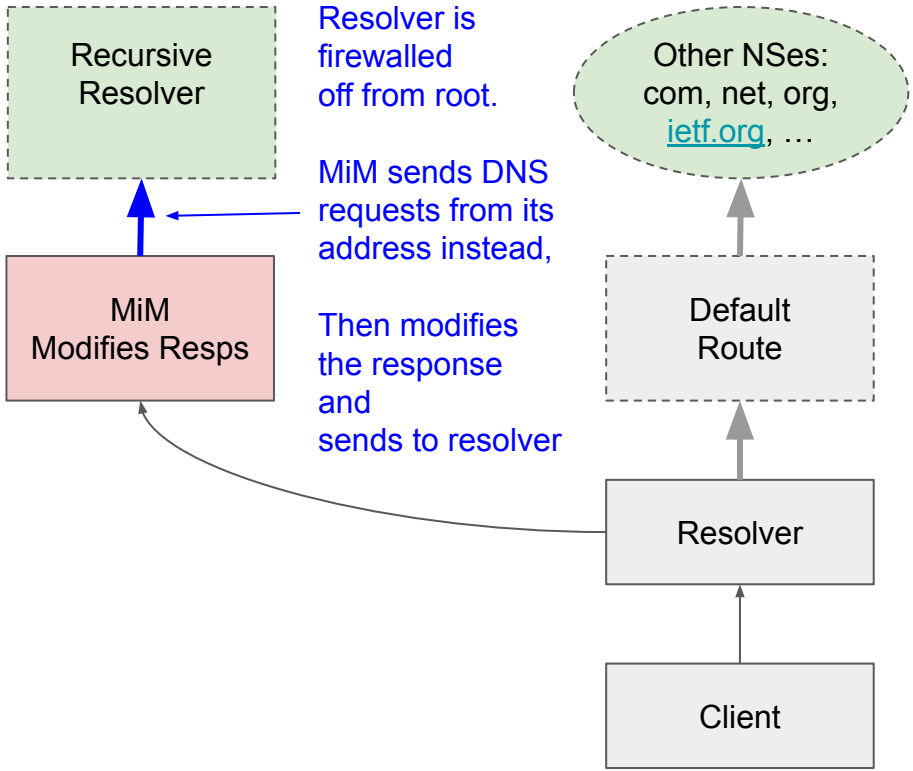
resolver to the root vs *resolver to everything else*



Parent Cache

Child Cache

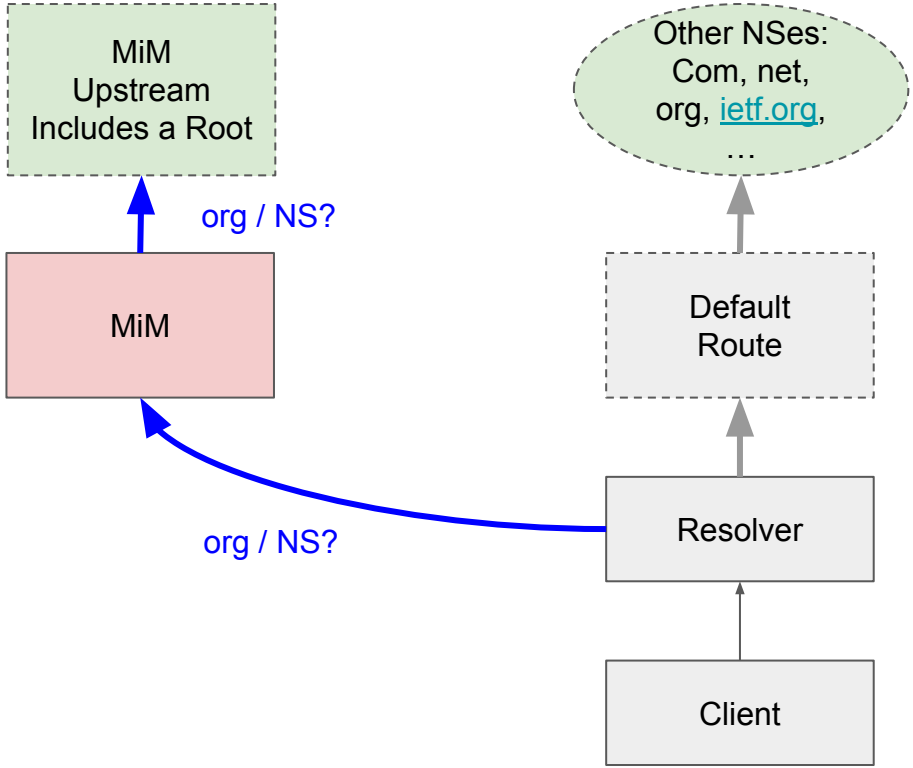
Implementation Notes:



Parent Cache

Child Cache

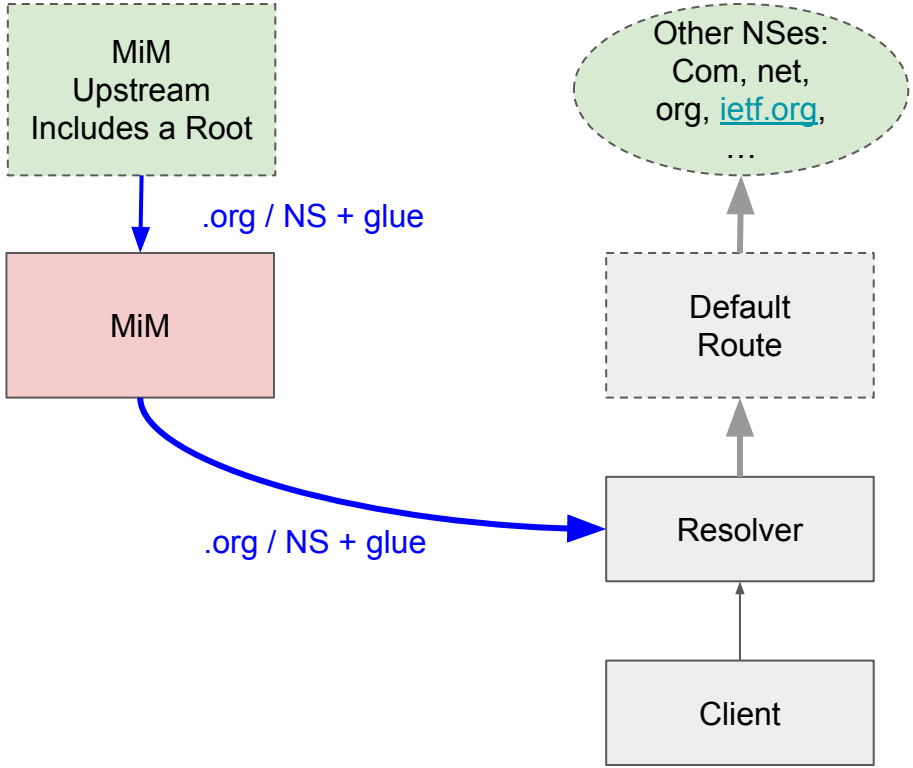
Demonstration Flow: resolver needs to know where .org is



Parent Cache

Child Cache

The DNS root will answer

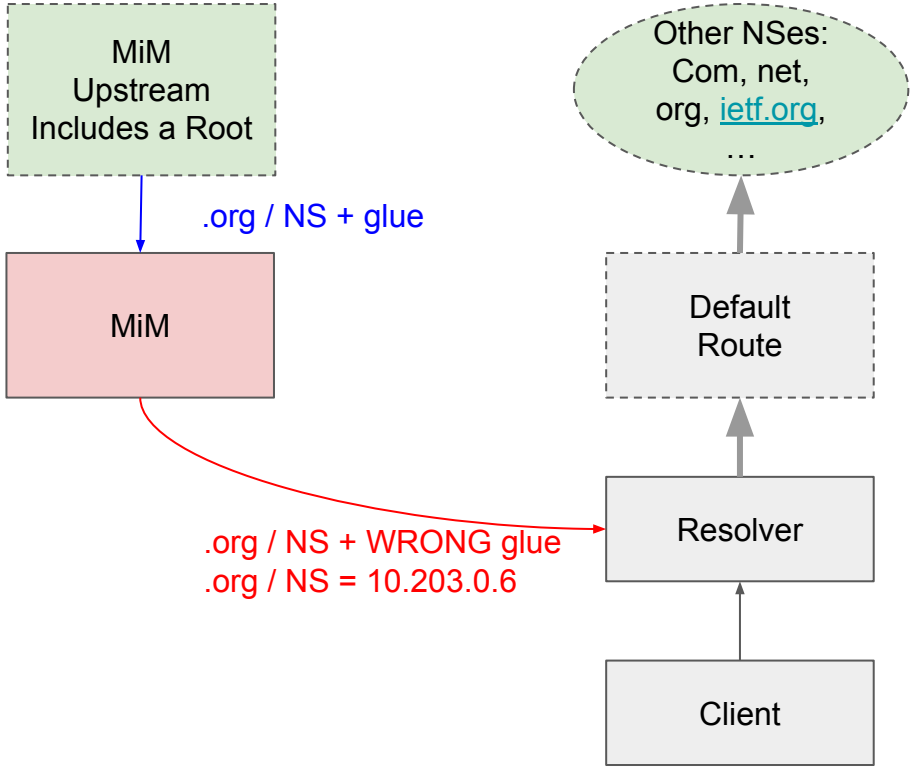


Parent Cache

.org NS Glue

Child Cache

A MiM can modify the resultts



Parent Cache

.org NS Glue

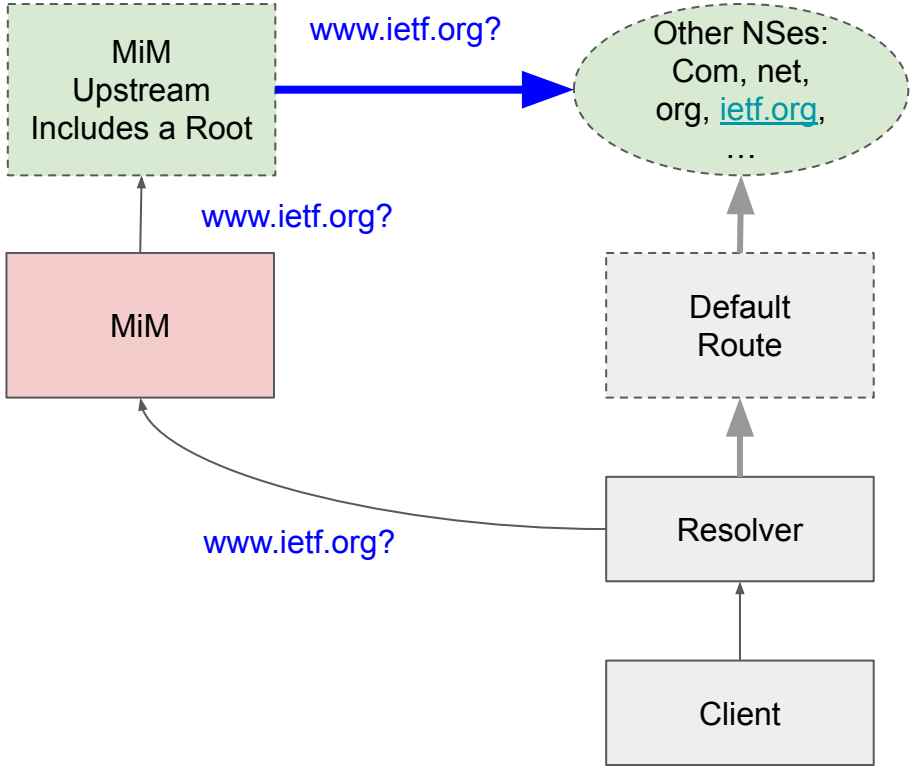
Child Cache

This continues…

- First: .org: where are you???

- Next: .ietf.org: where are you???

Demo – many steps later

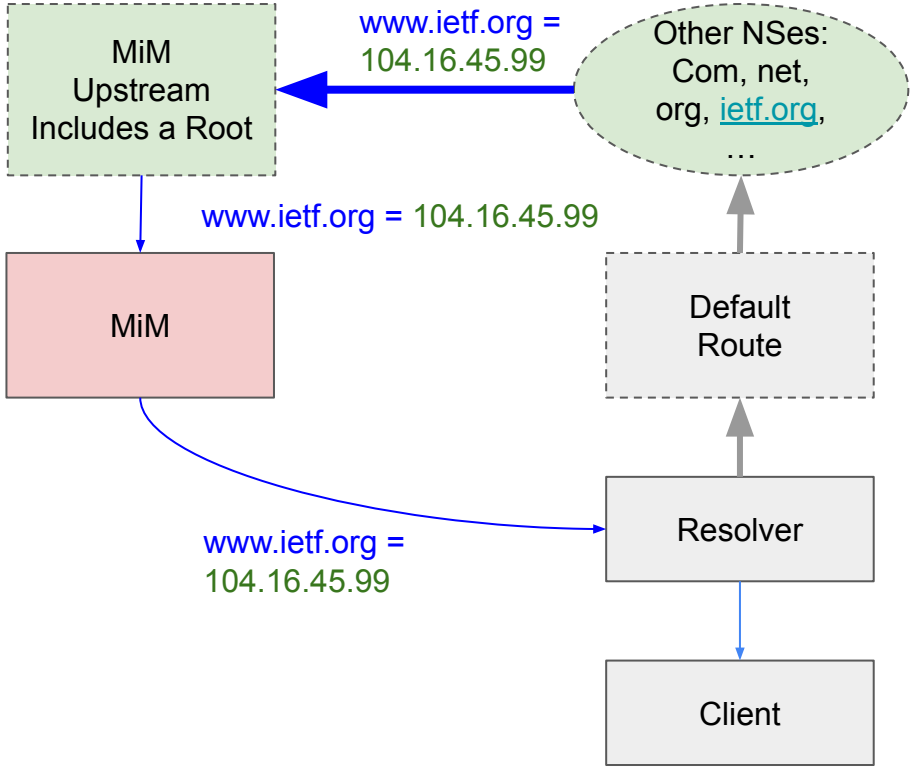


Parent Cache

.org	NS	Glue
.ietf.org	NS	Glue

Child Cache

Demo – many steps later



Parent Cache

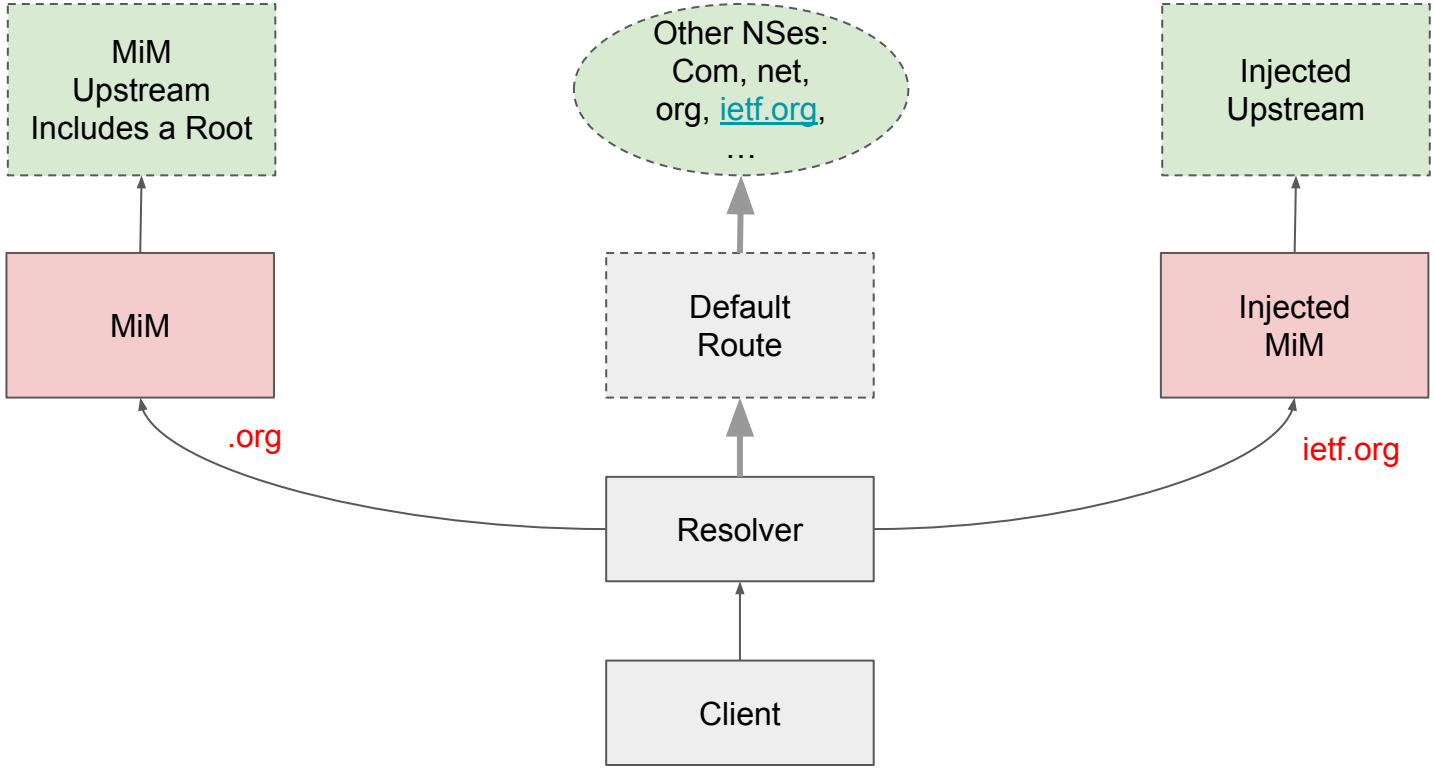
.org	NS	Glue
.ietf.org	NS	Glue

Child Cache

www.ietf.org	A

Demonstration

Note: other MiMs can be injected too



So What?

Ramifications of MiM Injections

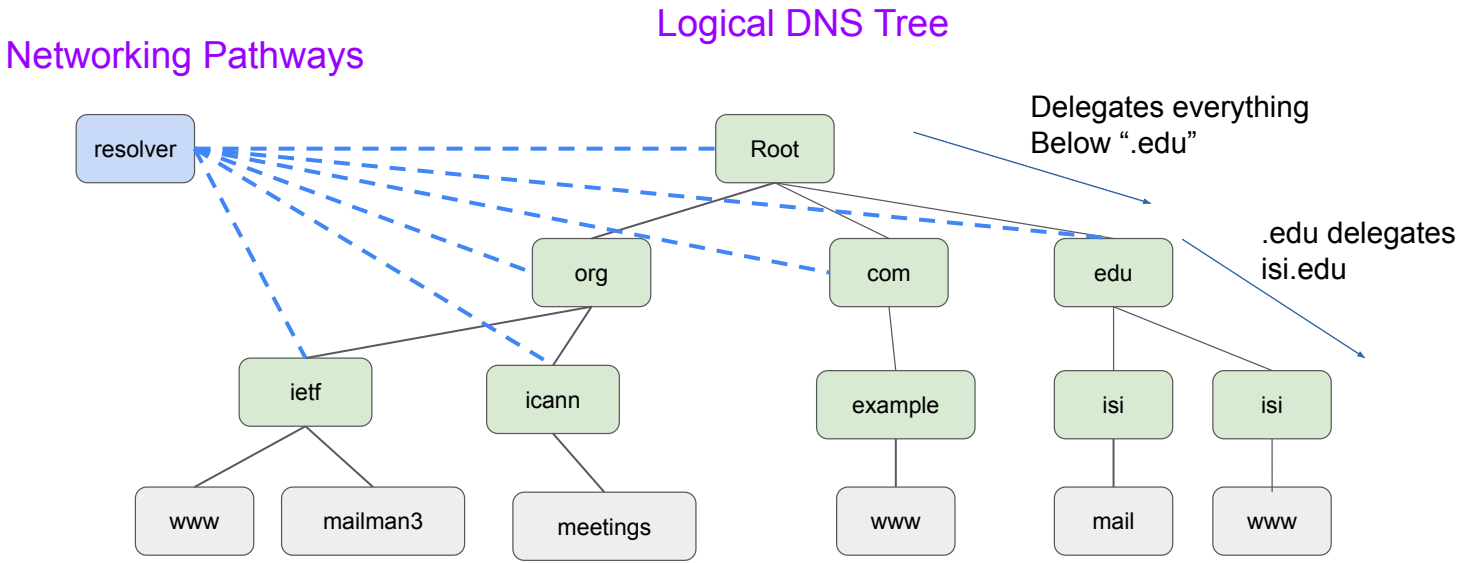
TL;DR: increased attack surface

1. Visibility of DNS requests not normally out of view

2. Modification of DNS responses normally out of view

Let’s look at someone else’s DNS requests

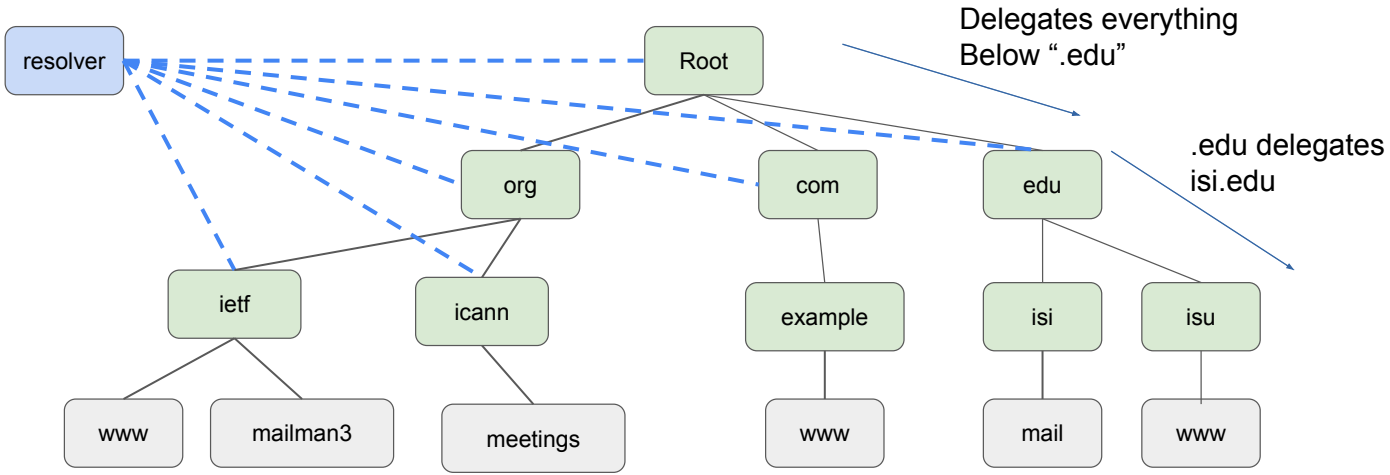
DNS Resolvers Communicate With Everything



Greater visibility of DNS requests and responses

Reminder: this requires a machine-in-the-middle already!

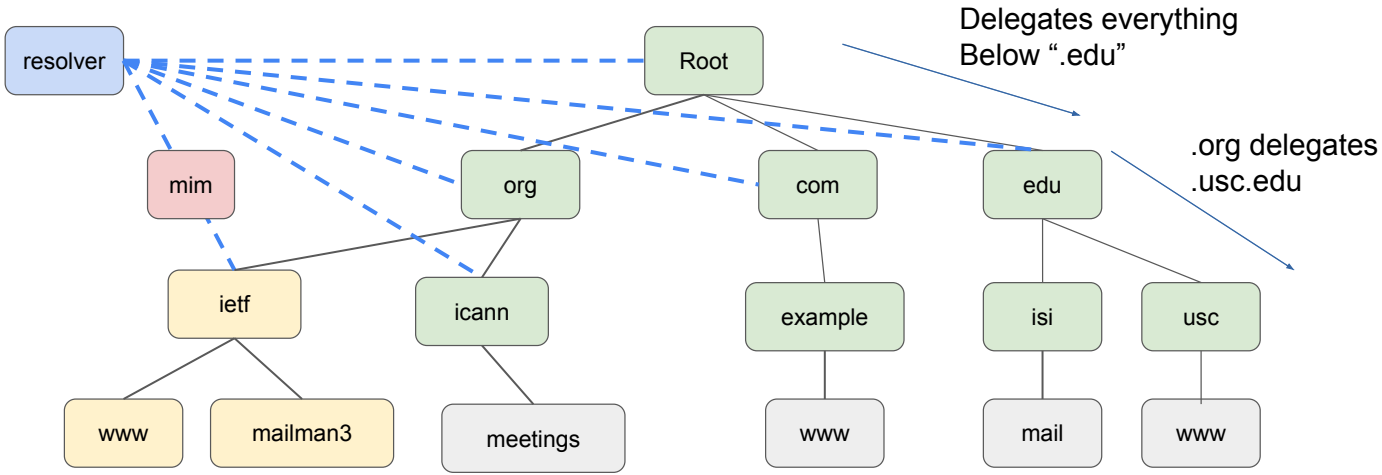
But: I can now force my vantage point in front of others



Greater visibility of DNS requests and responses

Reminder: this requires a machine-in-the-middle already!

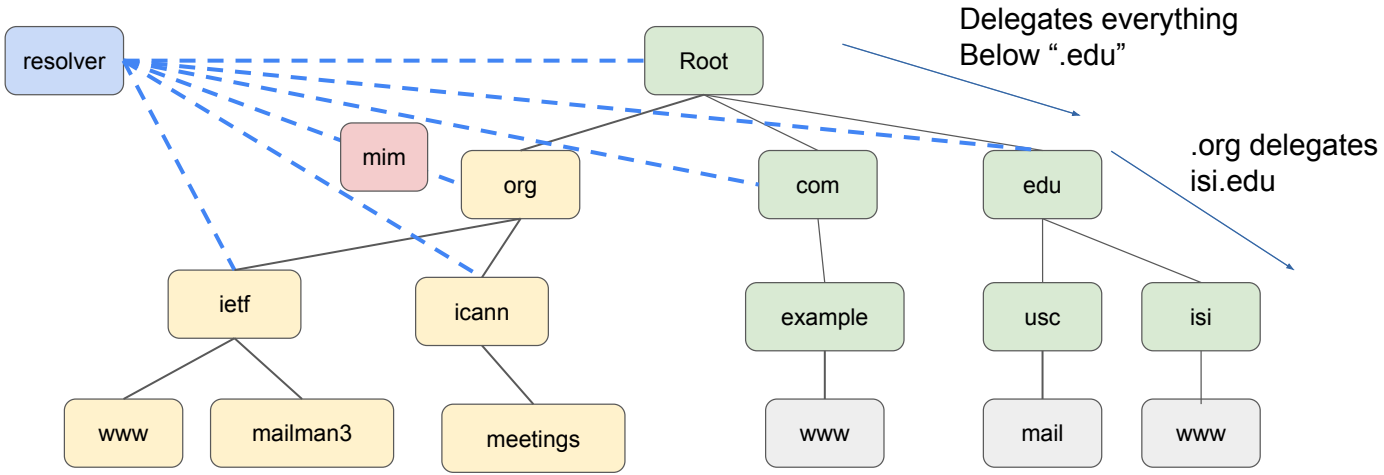
If the MiM can only see one domain's traffic…. not a big deal.



Greater visibility of DNS requests and responses

Reminder: this requires a machine-in-the-middle already!

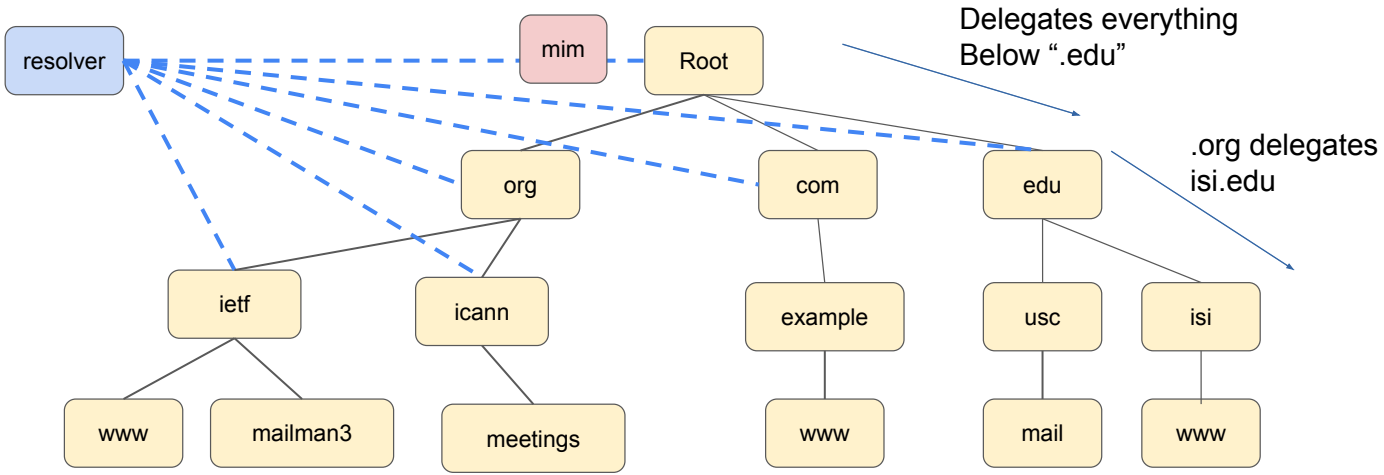
If the MiM can see higher points in the tree, they can see everything below



Greater visibility of DNS requests and responses

Reminder: this requires a machine-in-the-middle already!

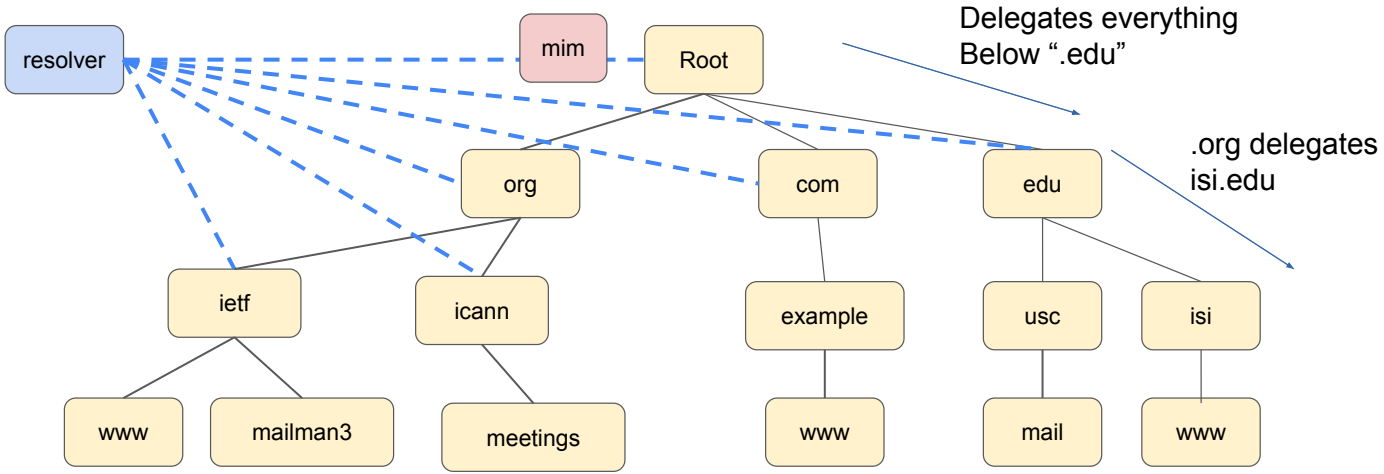
And a MiM before the root gives a vantage point of the entire Internet's DNS



Greater visibility of DNS requests and responses

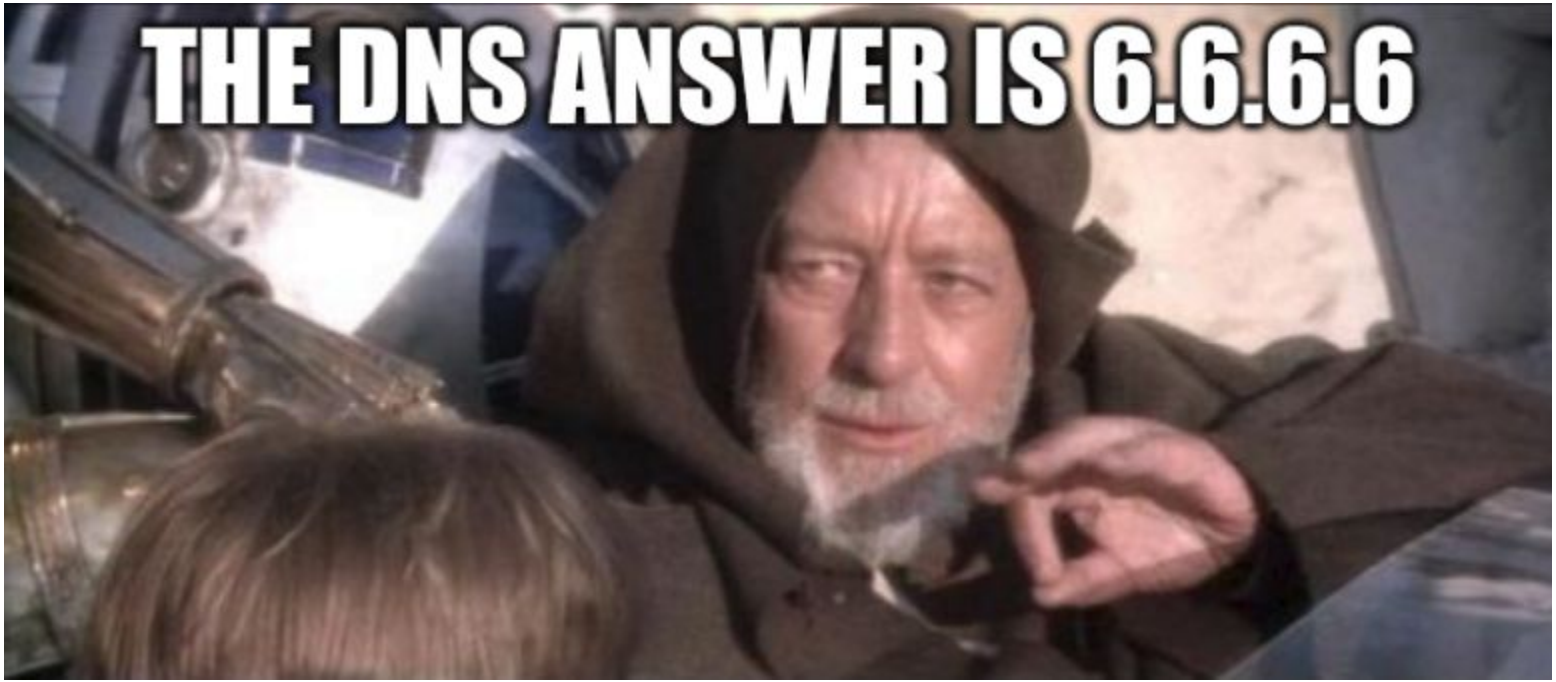
And a MiM before the root gives a vantage point of the entire Internet's DNS

Even though **ALL** these zones are **DNSSEC signed**



Parent-centric ramifications of not checking DNSSEC

- MiM vantage points can be inserted into the DNS tree
  - Starting with any vantage point higher in the tree
- Difficult to detect in the client
- A MiM can:
  - Modify delegation information
    - A / AAAA glue *(this is hard to detect)*
    - NS names *(this is easier to detect)*
  - View all DNS transactions beyond their normal vantage point
  - Modify unsigned contents lower in the tree they normally can’t see


THE DNS ANSWER IS 6.6.6.6

Attacking non-DNSSEC protected zones

Expanded vantage points of non-DNSSEC protected zones can:

- Altering MX records
  - Re-routing mail through a attacker controlled relay
- Altering DKIM/DMARC
  - Spoofing mail to the attack target from other DNS services
- Altering DNS HTTPS records
  - downgrading http version
  - ECH manipulation
  - Adding or changing ipv4/v6 glue

FAQ

Does client-centric resolution solve this?

Yes, because the child's notion of the NS and address records are signed and used instead

Can the end-user / client detect this?

Not easily

Can DNSSEC catch this?

No, because parent-side NS/glue are never checked

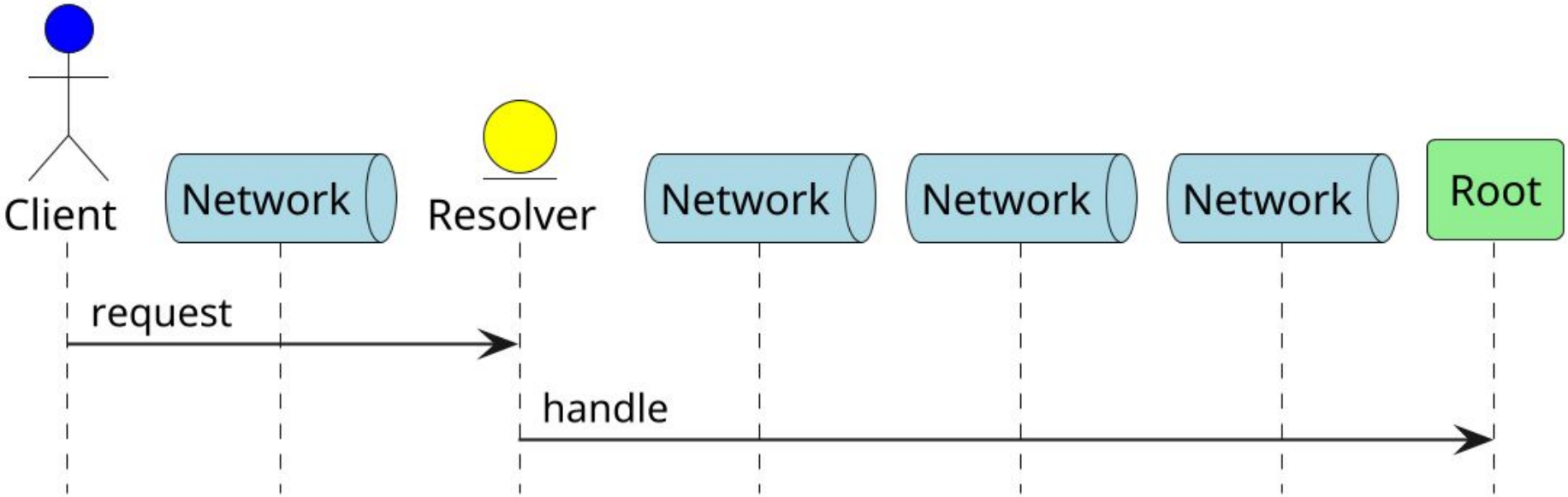
But: Data going to clients is validated through DS -> DNSKEY -> … -> DNSKEY -> RR

What about DELEG?

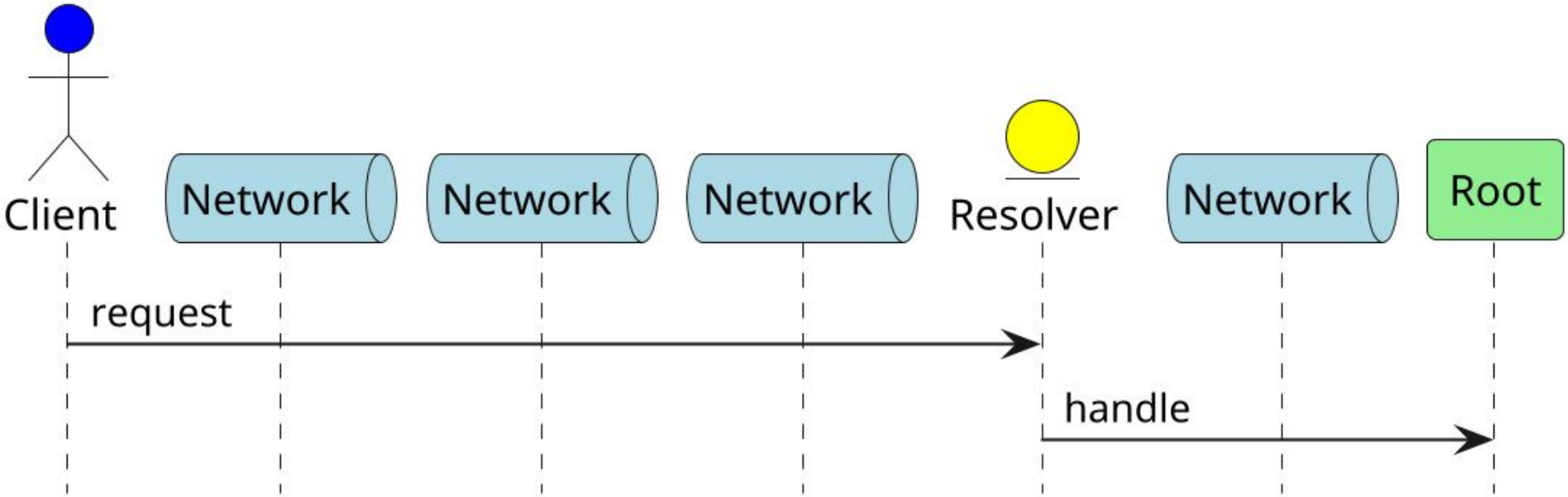
Mitigations

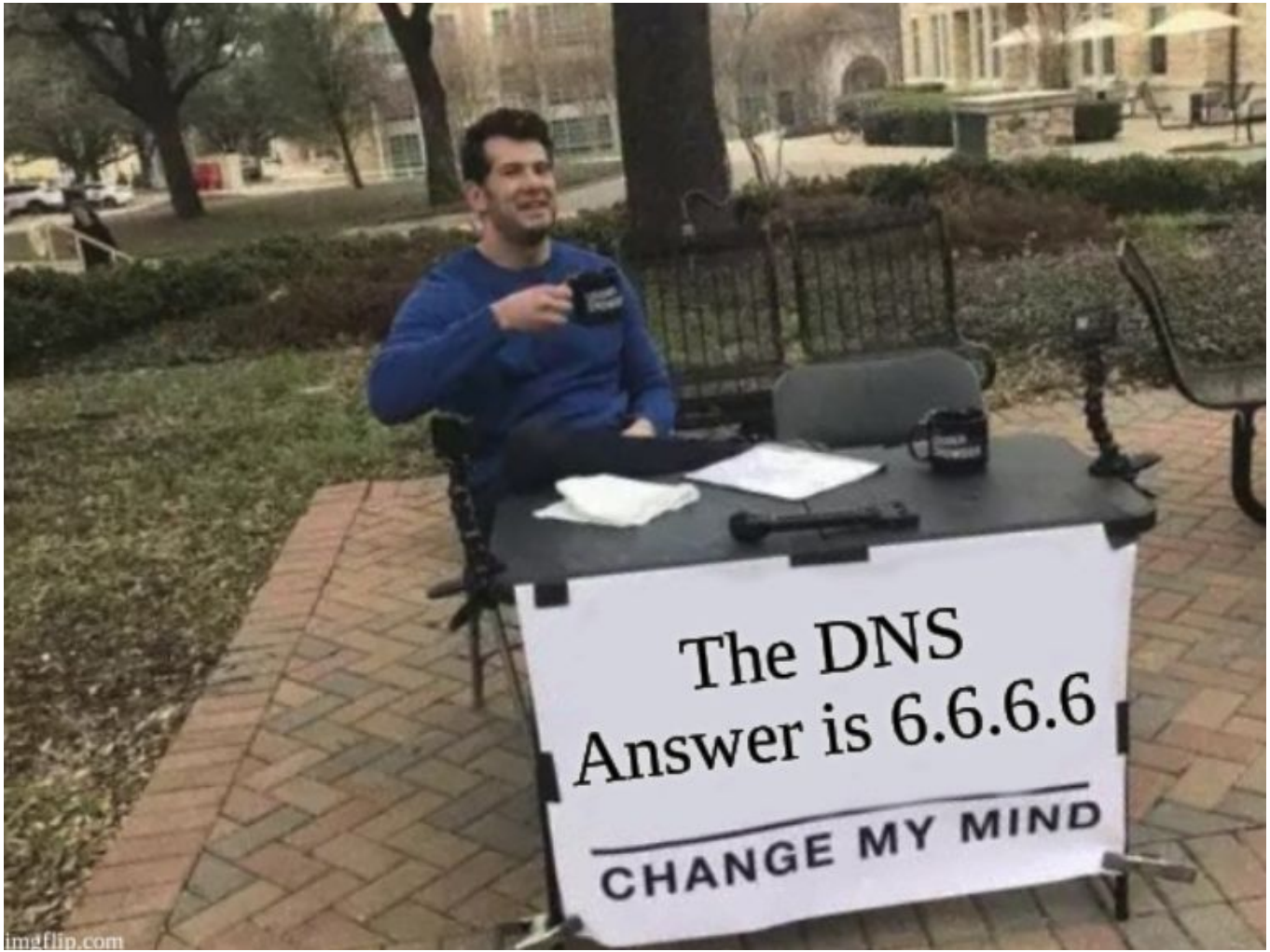
- Use DNSSEC validating, child-centric resolvers
  - See draft-ietf-dnsop-ns-revalidation
- Deploy parent-centric resolvers close to authoritative server infrastructure
  - (aka peer widely)
- Deploy TLS between your clients and your resolver
- Look forward to DELEG records, which are parent-signed

Resolvers are typically placed near clients



One option: placing resolvers close to authoritative servers

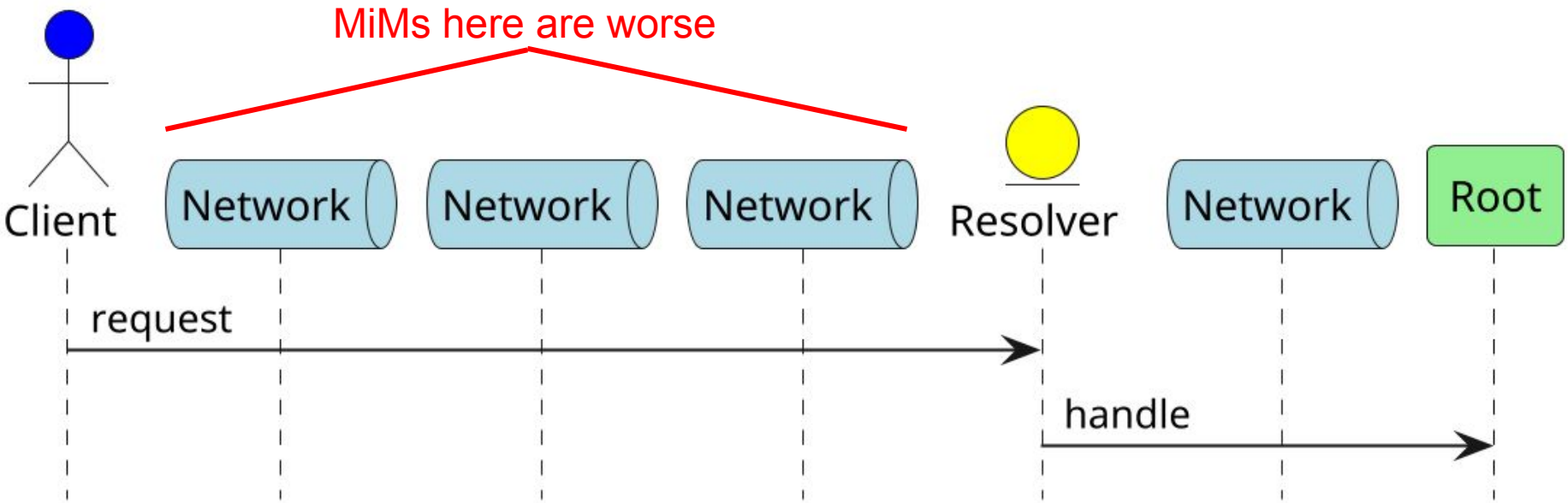




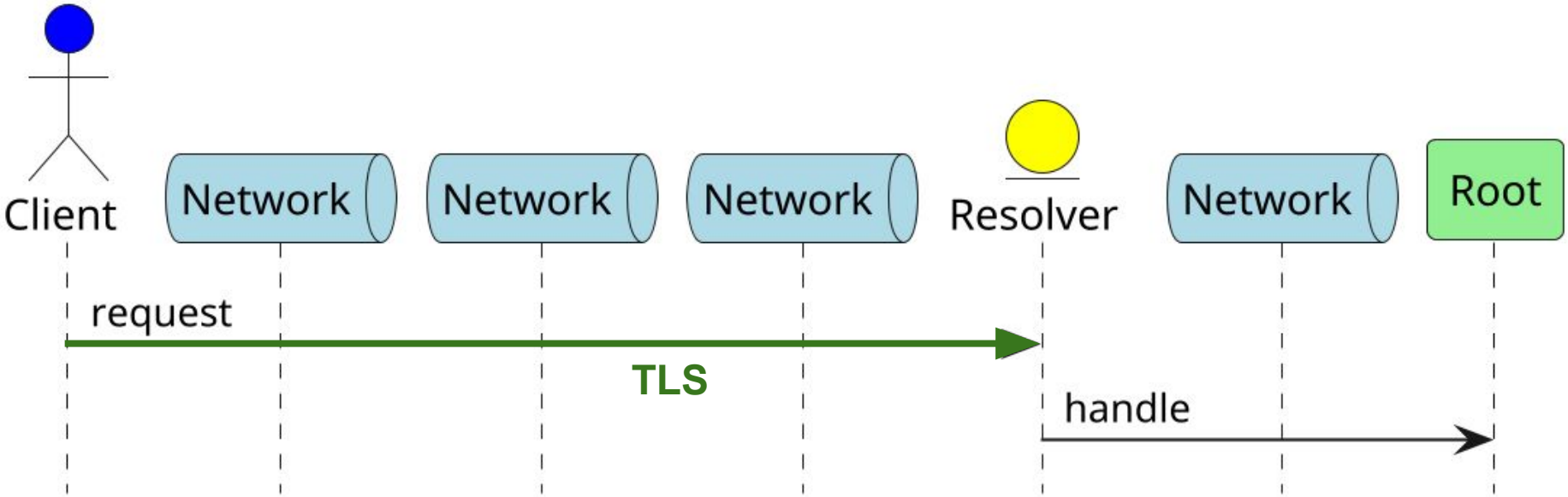
Thanks for the new MiM spot

You've only moved the problem

You've just moved the problem



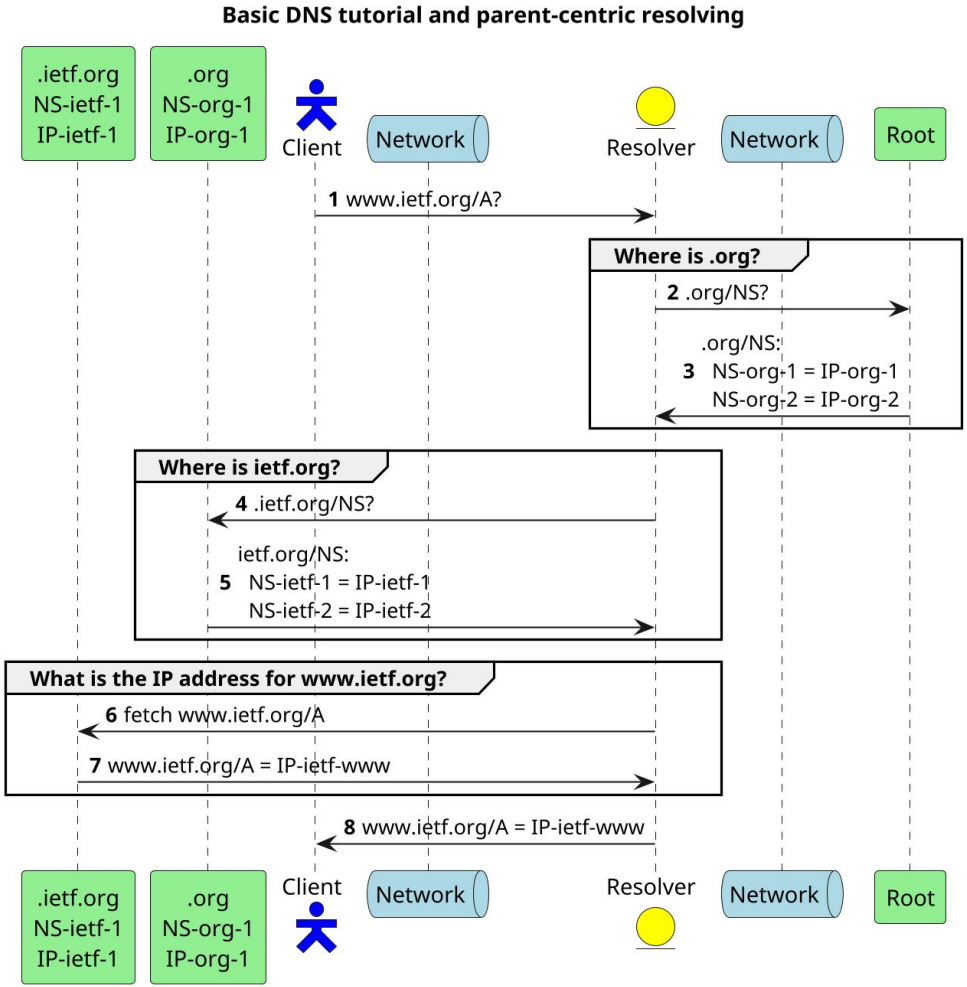
Solution: deploy TLS here

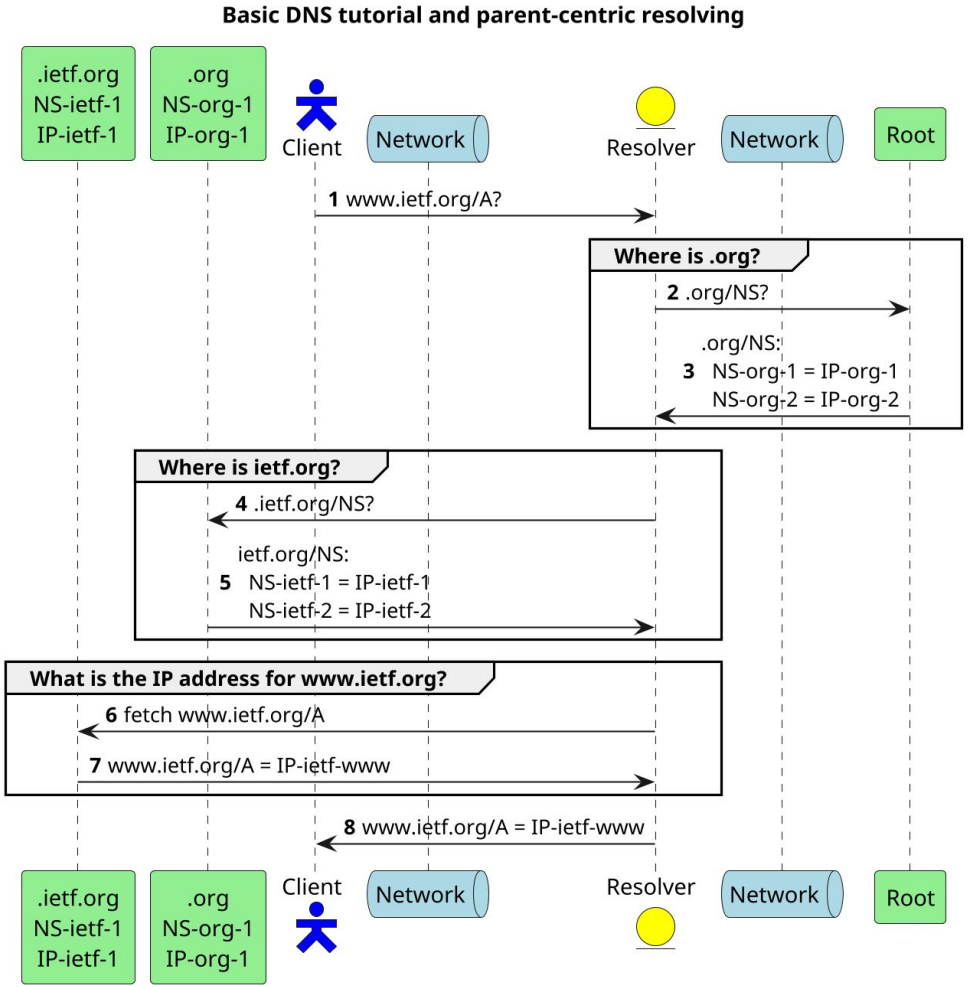


Takeaways

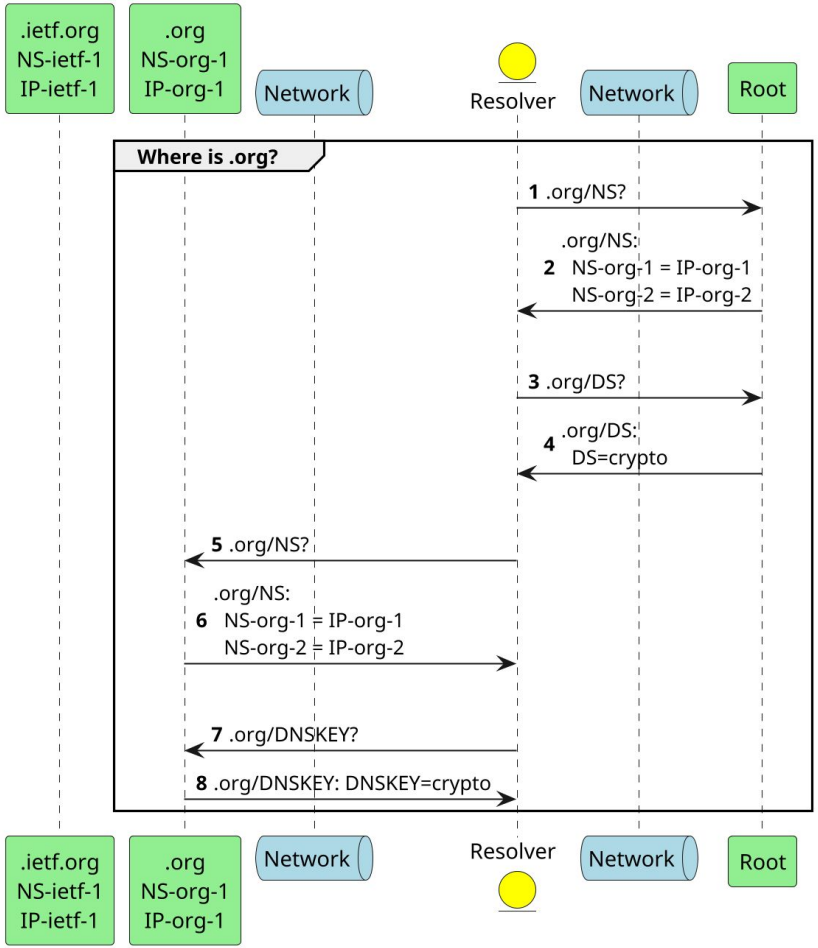
- Parent-centric, DNSSEC validating resolvers:
  - Can be subject to expanded view injection attacks for all zones
  - Can be subject to expanded view modification attacks for unsigned zones
  - Other work has proven them to be faster and more robust, however
- Mitigations
  - Choose resolver software carefully
  - Deploy parent-centric resolvers with wide-peering
- DELEG can’t come fast enough

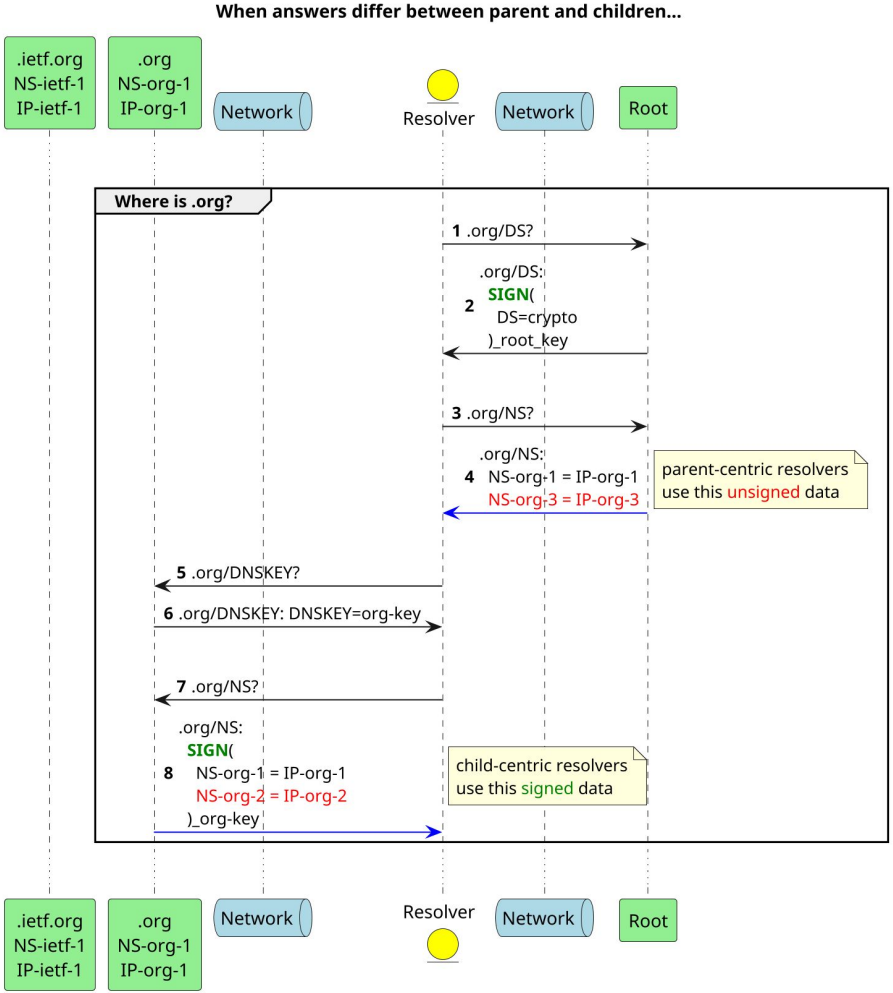
Backup Extra Complex Slides

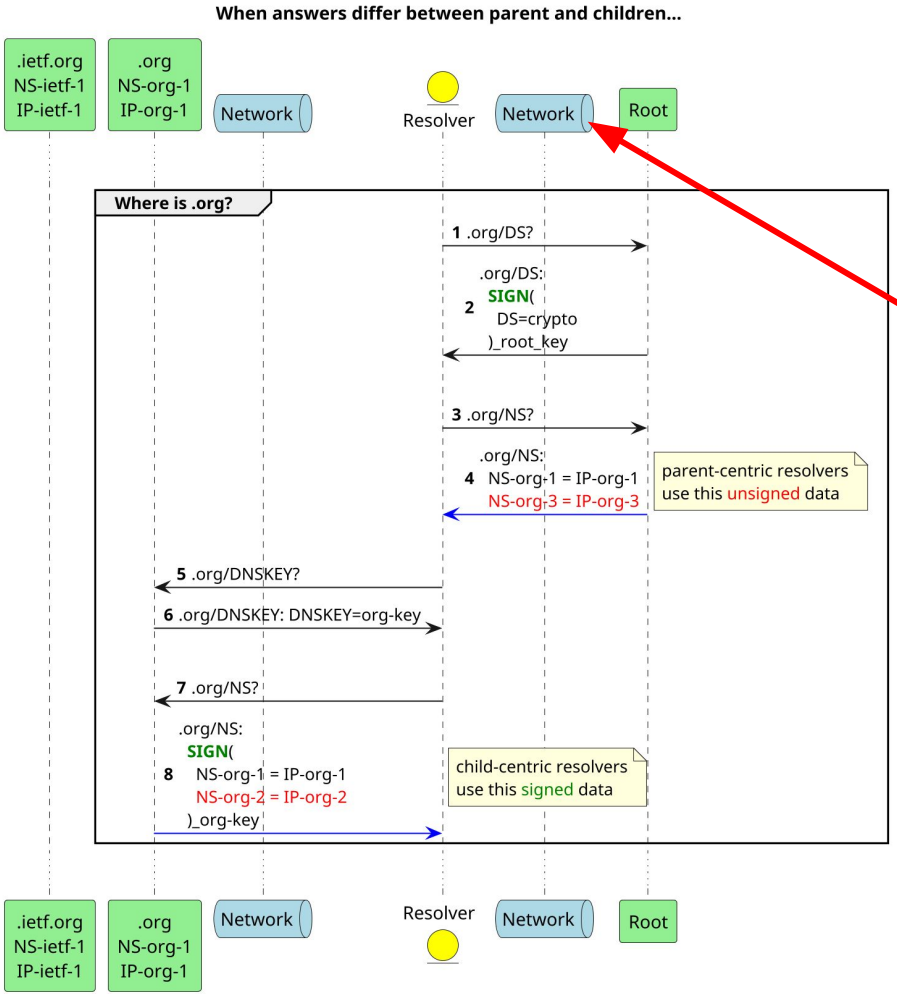

Basic DNS tutorial and parent-centric resolving
.ietf.org
NS-ietf-1
IP-ietf-1
.org
NS-org-1
IP-org-1
Client
Network
Resolver
Network
Root
1 www.ietf.org/A?
Where is .org?
2 .org/NS?
.org/NS:
3 NS-org-1 = IP-org-1
NS-org-2 = IP-org-2
Where is ietf.org?
4 .ietf.org/NS?
ietf.org/NS:
5 NS-ietf-1 = IP-ietf-1
NS-ietf-2 = IP-ietf-2
What is the IP address for www.ietf.org?
6 fetch www.ietf.org/A
7 www.ietf.org/A = IP-ietf-www
8 www.ietf.org/A = IP-ietf-www
.ietf.org
NS-ietf-1
IP-ietf-1
.org
NS-org-1
IP-org-1
Client
Network
Resolver
Network
Root



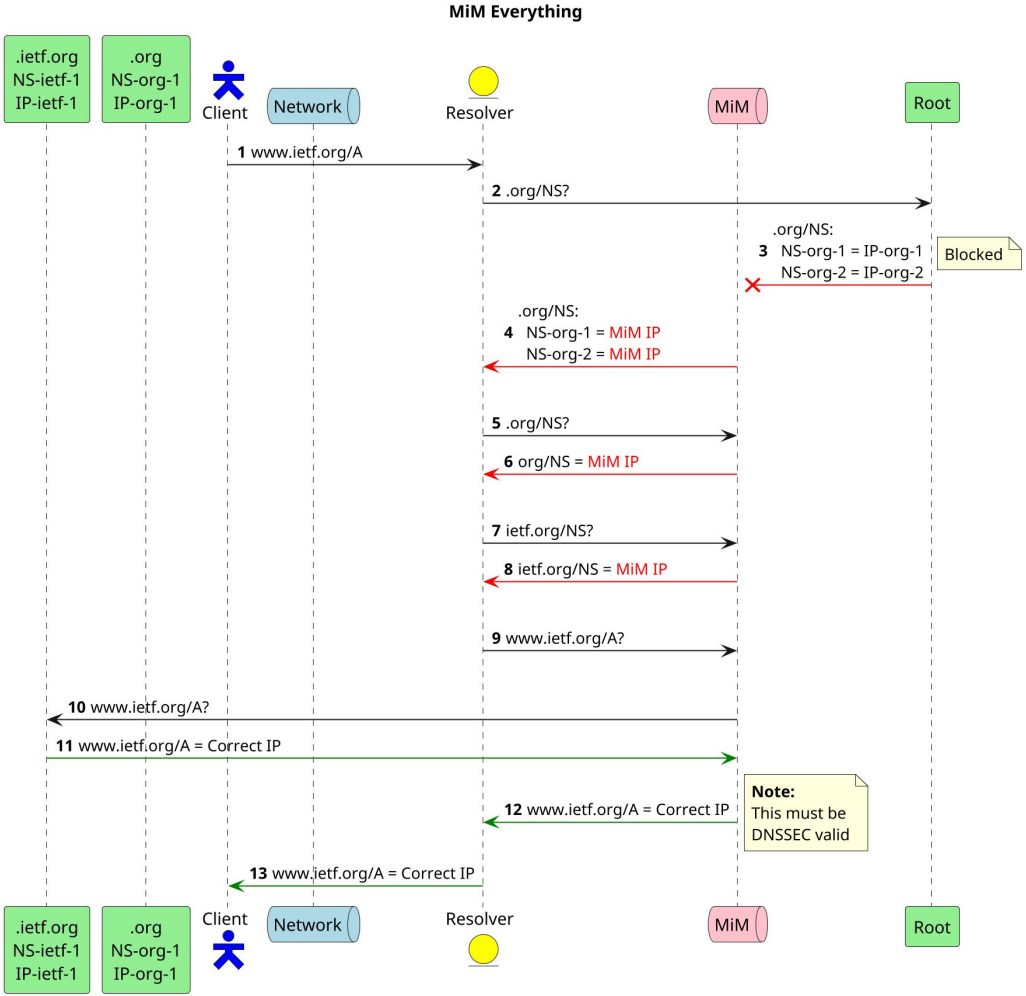
Let's dive into just this transaction with DNSSEC validation


Validated DNS referrals and child-centric resolving
.ietf.org
NS-ietf-1
IP-ietf-1
.org
NS-org-1
IP-org-1
Network
Resolver
Network
Root
Where is .org?
1 .org/NS?
.org/NS:
2 NS-org-1 = IP-org-1
NS-org-2 = IP-org-2
3 .org/DS?
.org/DS:
4 DS=crypto
5 .org/NS?
.org/NS:
6 NS-org-1 = IP-org-1
NS-org-2 = IP-org-2
7 .org/DNSKEY?
8 .org/DNSKEY: DNSKEY=crypto
.ietf.org
NS-ietf-1
IP-ietf-1
.org
NS-org-1
IP-org-1
Network
Resolver
Network
Root

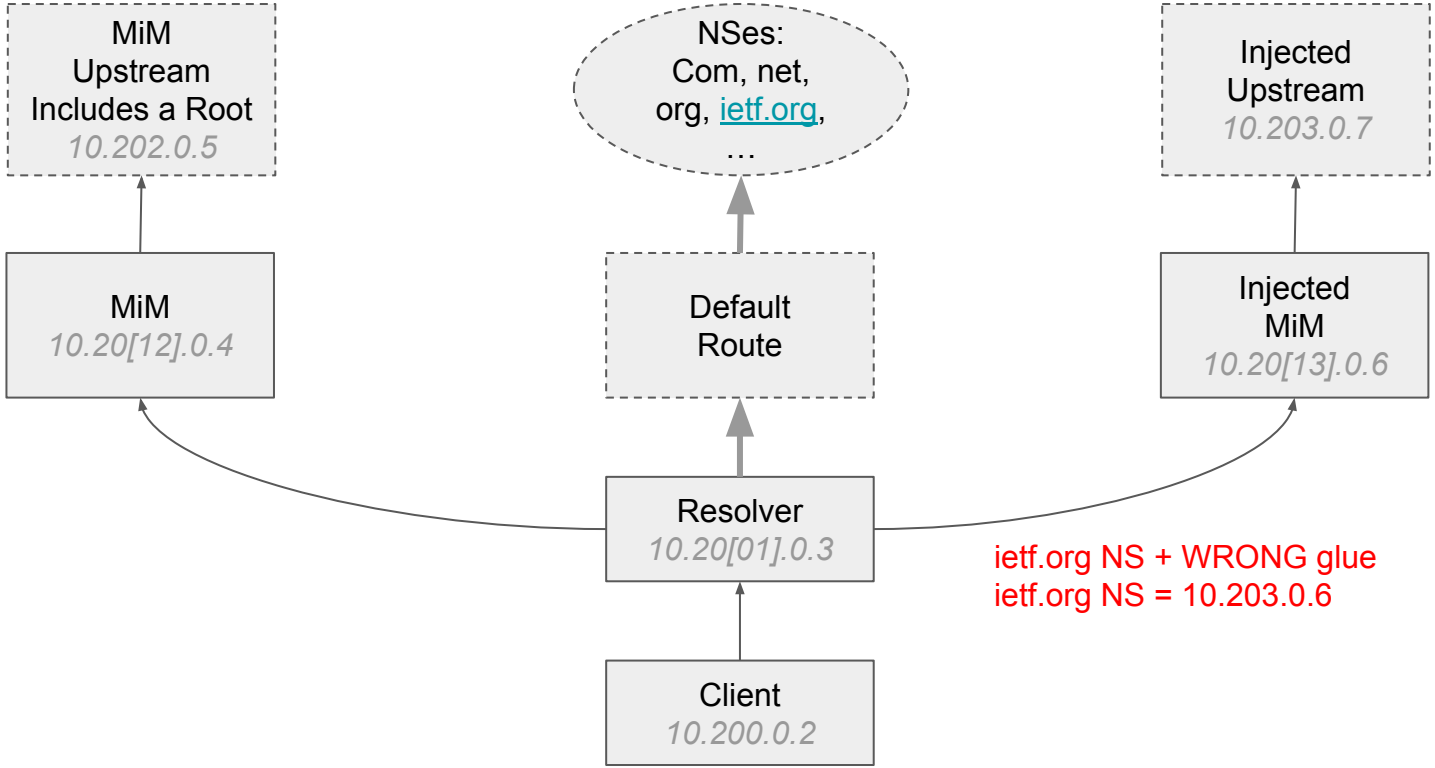

When answers differ between parent and children...
.ietf.org
NS-ietf-1
IP-ietf-1
.org
NS-org-1
IP-org-1
Network
Resolver
Network
Root
Where is .org?
1 .org/DS?
.org/DS:
2 SIGN(
DS=crypto
)_root_key
3 .org/NS?
.org/NS:
4 NS-org-1 = IP-org-1
NS-org-3 = IP-org-3
parent-centric resolvers
use this unsigned data
5 .org/DNSKEY?
6 .org/DNSKEY: DNSKEY=org-key
7 .org/NS?
.org/NS:
SIGN(
8 NS-org-1 = IP-org-1
NS-org-2 = IP-org-2
)_org-key
child-centric resolvers
use this signed data
.ietf.org
NS-ietf-1
IP-ietf-1
.org
NS-org-1
IP-org-1
Network
Resolver
Network
Root



What if there was a MiM here?


MiM Everything
.ietf.org
NS-ietf-1
IP-ietf-1
.org
NS-org-1
IP-org-1
Client
Network
Resolver
MiM
Root
1 www.ietf.org/A
2 .org/NS?
.org/NS:
3 NS-org-1 = IP-org-1
NS-org-2 = IP-org-2
Blocked
.org/NS:
4 NS-org-1 = MiM IP
NS-org-2 = MiM IP
5 .org/NS?
6 org/NS = MiM IP
7 ietf.org/NS?
8 ietf.org/NS = MiM IP
9 www.ietf.org/A?
10 www.ietf.org/A?
11 www.ietf.org/A = Correct IP
Note:
This must be
DNSSEC valid
12 www.ietf.org/A = Correct IP
13 www.ietf.org/A = Correct IP

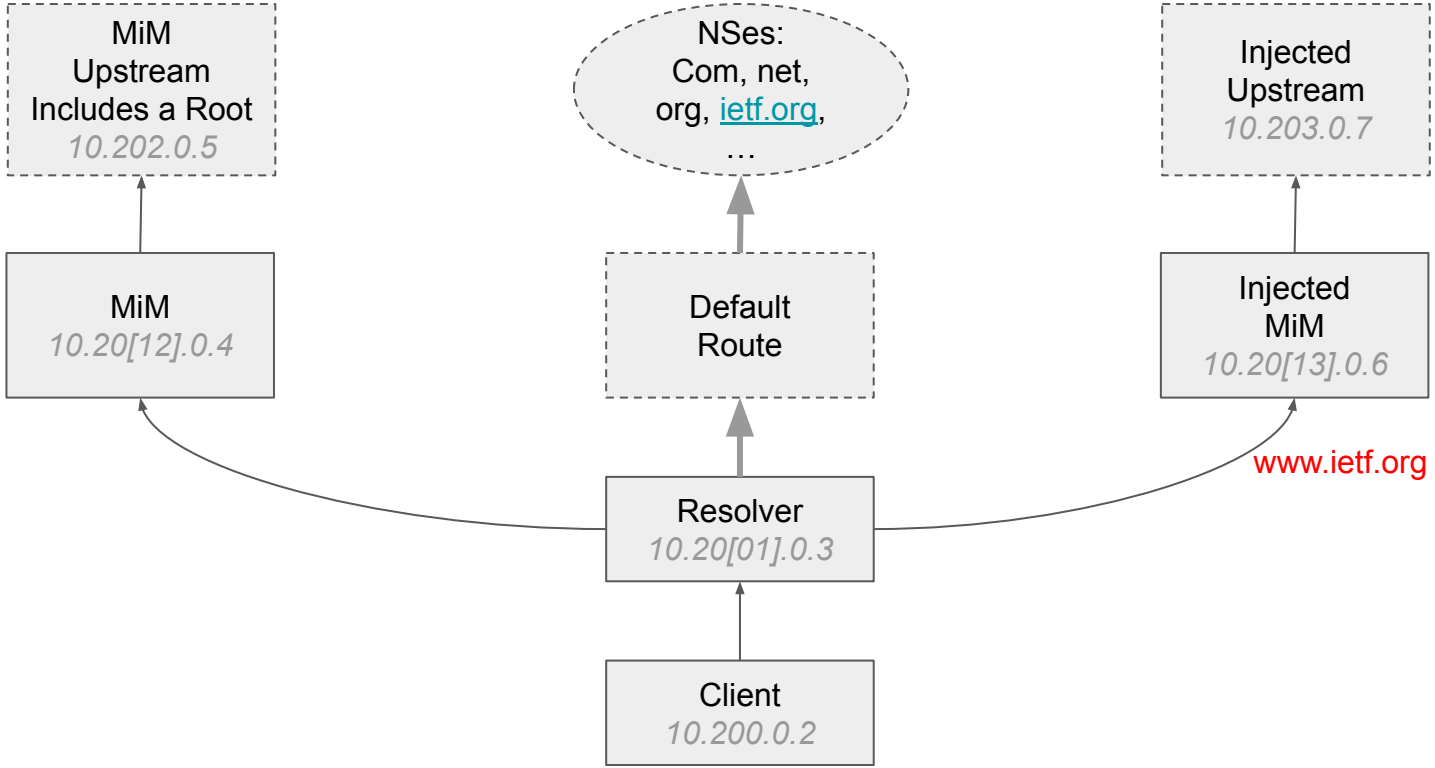
Test Network Design



Parent Cache
.org/NS = 10.203.0.6
ietf.org/NS = 10.203.0.6

Child Cache

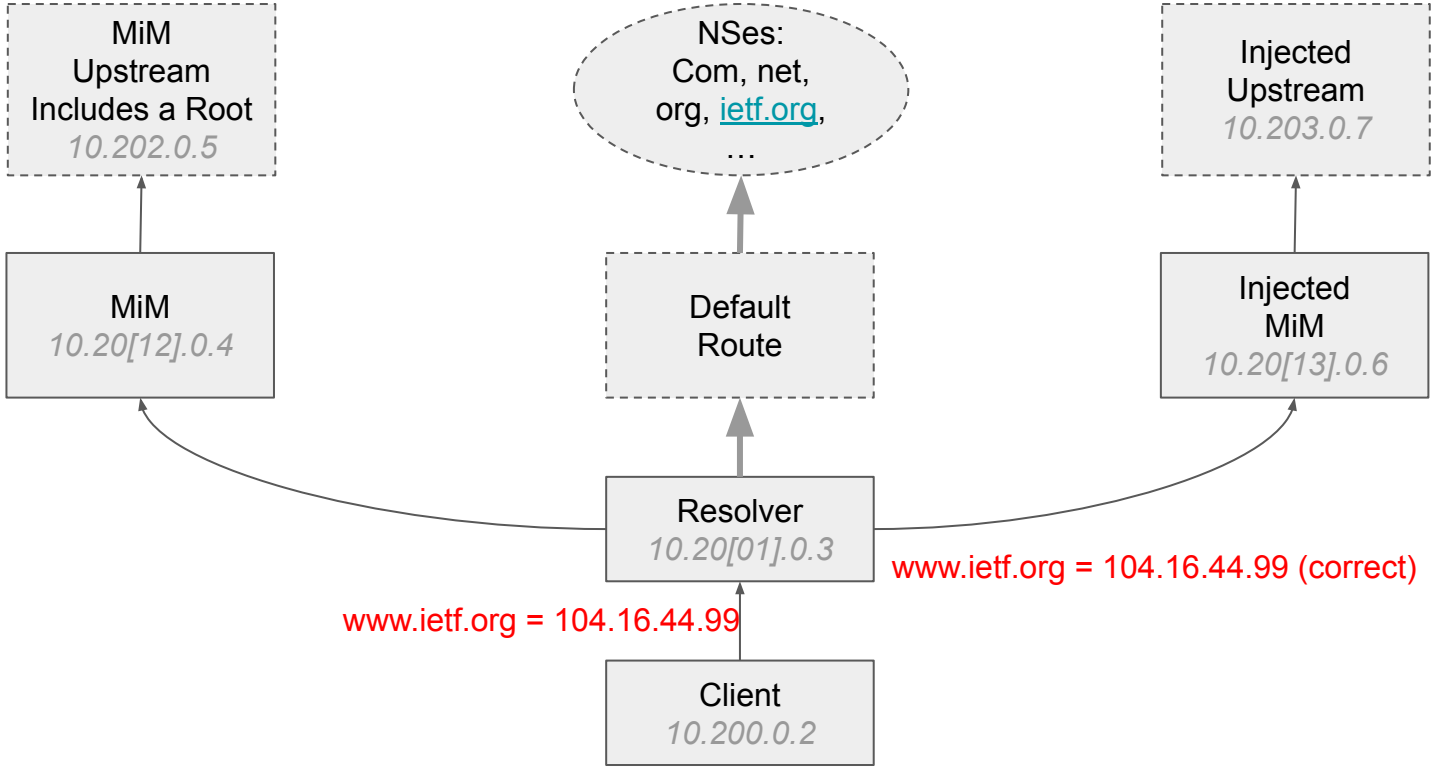
Test Network Design



Parent Cache
.org/NS = 10.203.0.6
ietf.org/NS = 10.203.0.6

Child Cache

Test Network Design



Parent Cache
.org/NS = 10.203.0.6
ietf.org/NS = 10.203.0.6

Child Cache
www.ietf.org = 104.16.44.99

*** Important demonstration components

1. Resolver routes are configured so all root traffic goes left
    a. All other Internet traffic goes out the default
2. The MiM cannot forward traffic to the root without modification
3. Only traffic to the root must come from the MiM source address
4. The MiM intercepts packets and:
    a. Modifies the source address to be its own
    b. Sends requests to an upstream resolver
    c. Gets the responses back and modifies any glue to its own IP address (or the injected MiM)
    d. If the type = NS, modifies all A responses to its own IP address
    e. Replaces the source / destination addresses to match the original request
    f. Delivers it back to the resolver under attack

Demonstration Modes

A. Normal routed traffic – only the root traffic is seen by the MiM
B. MiM injecting itself in front of all DNS requests
C. MiM injecting another MiM in front of all requests
    a. Which subsequently injects itself for all other requests