SCALE

Automatically Finding RFC Compliance Bugs in DNS Nameservers

Siva Kesava Reddy Kakarla

Ryan Beckett

Todd Millstein

George Varghese


Microsoft


University of California, Los Angeles


Intentionet

Website Domain Name → IP

DNS (Domain Name System)

Website Domain Name → IP

DNS (Domain Name System)

Many DNS Implementations

Open-Source

BIND 9

POWERDNS

NSD

CoreDNS

KNOT DNS

Closed-Source

Akamai

CLOUDFLARE

DNS

Route53

DNS Software needs to be absolutely Correct!

- Incorrect responses from DNS servers can cause service unavailability
- Attackers can exploit security vulnerabilities (code bugs) to mount DDoS attacks

DNS Software needs to be absolutely Correct!

- Incorrect responses from DNS servers can cause service unavailability
- Attackers can exploit security vulnerabilities (code bugs) to mount DDoS attacks
- DNS outages have a “large blast radius”

DNS Software needs to be absolutely Correct!

- Incorrect responses from DNS servers can cause service unavailability
- Attackers can exploit security vulnerabilities (code bugs) to mount DDoS attacks
- DNS outages have a “large blast radius”

Bind DoS Bug

ISC updates critical DoS bug in BIND DNS software

The denial-of-service flaw in BIND can be triggered by specially crafted DNS packages and is capable of knocking critical servers offline

DNS Software needs to be absolutely Correct!

- Incorrect responses from DNS servers can cause service unavailability
- Attackers can exploit security vulnerabilities (code bugs) to mount DDoS attacks
- DNS outages have a “large blast radius”

Bind DoS Bug

ISC updates critical DoS bug in BIND DNS software

The denial-of-service flaw in BIND can be triggered by specially crafted DNS packages and is capable of knocking critical servers offline

Slack Outage due to Route 53 bug

Slack is down for some people, and of course, the problem is DNS

If you've been having trouble contacting co-workers, this may be why

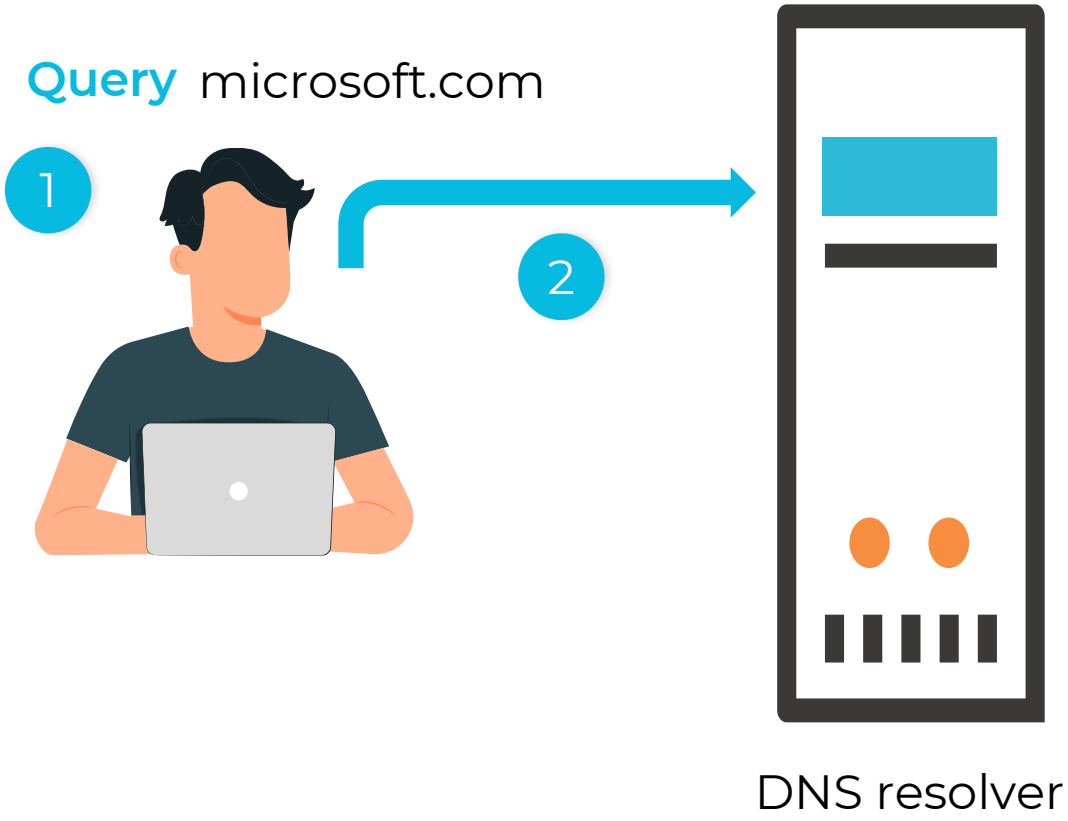
By Mitchell Clark | Updated Sep 30, 2021, 4:18pm EDT

How the Domain Name System Works

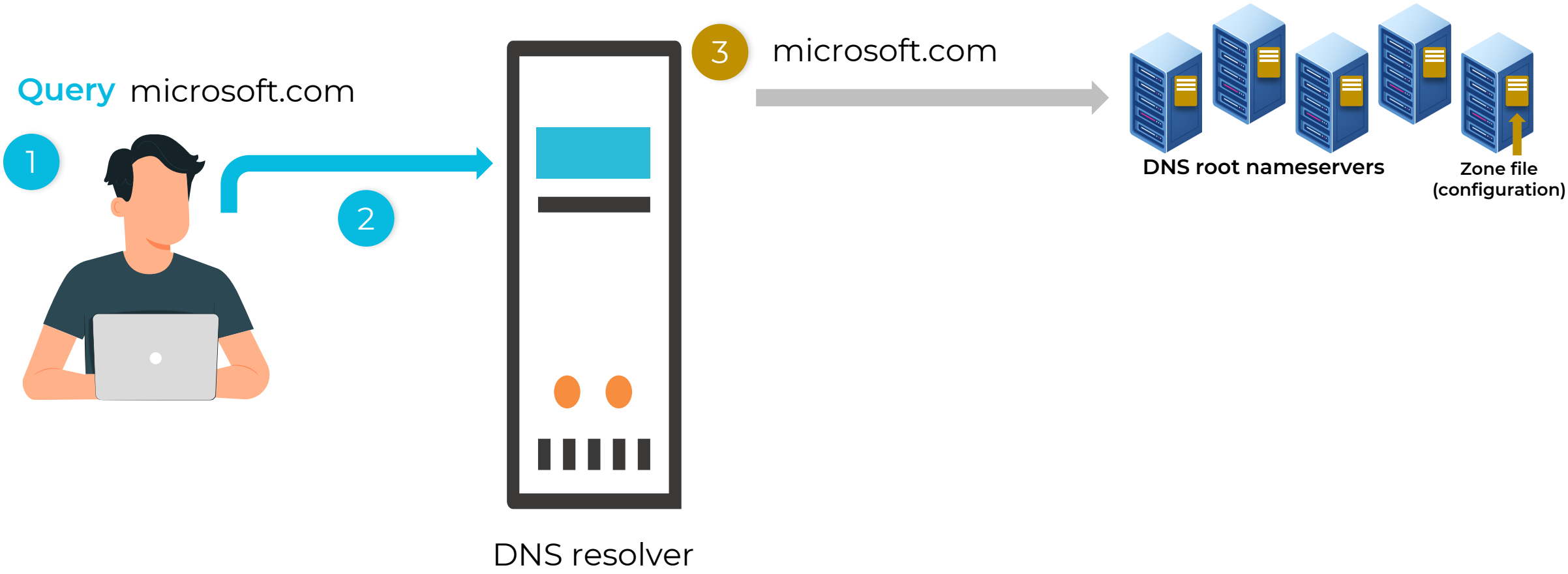
Query microsoft.com

1

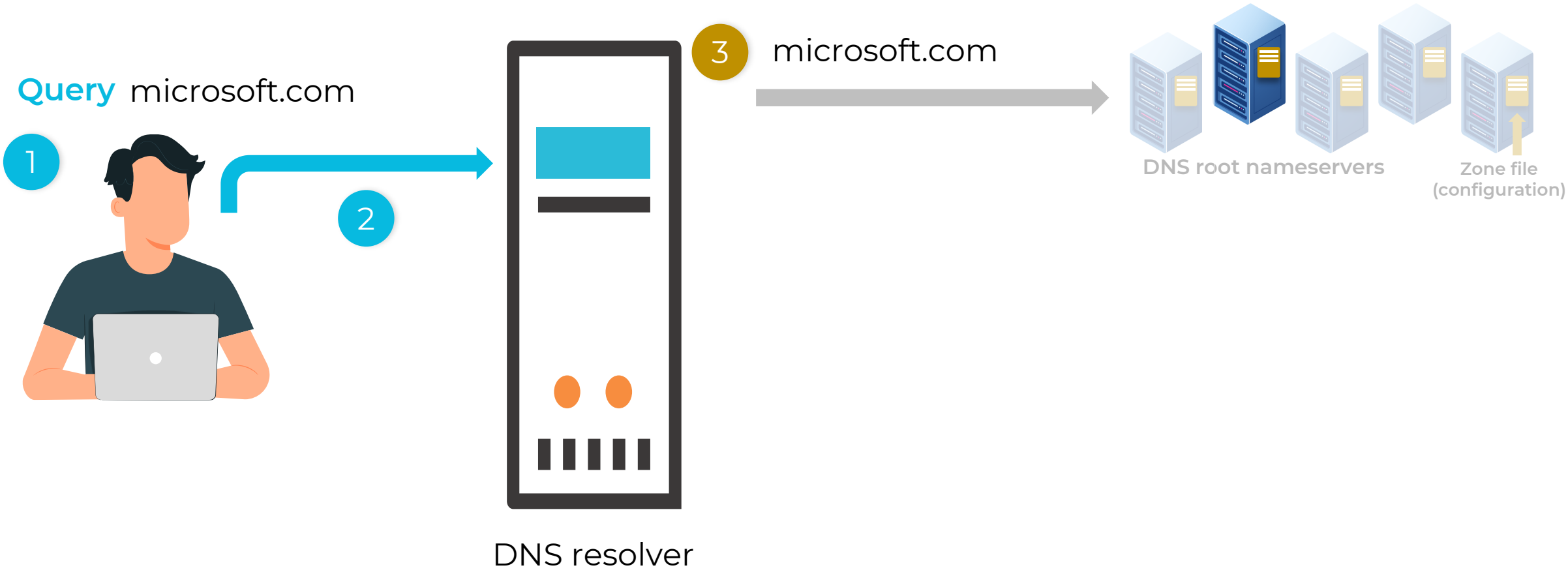
How the Domain Name System Works



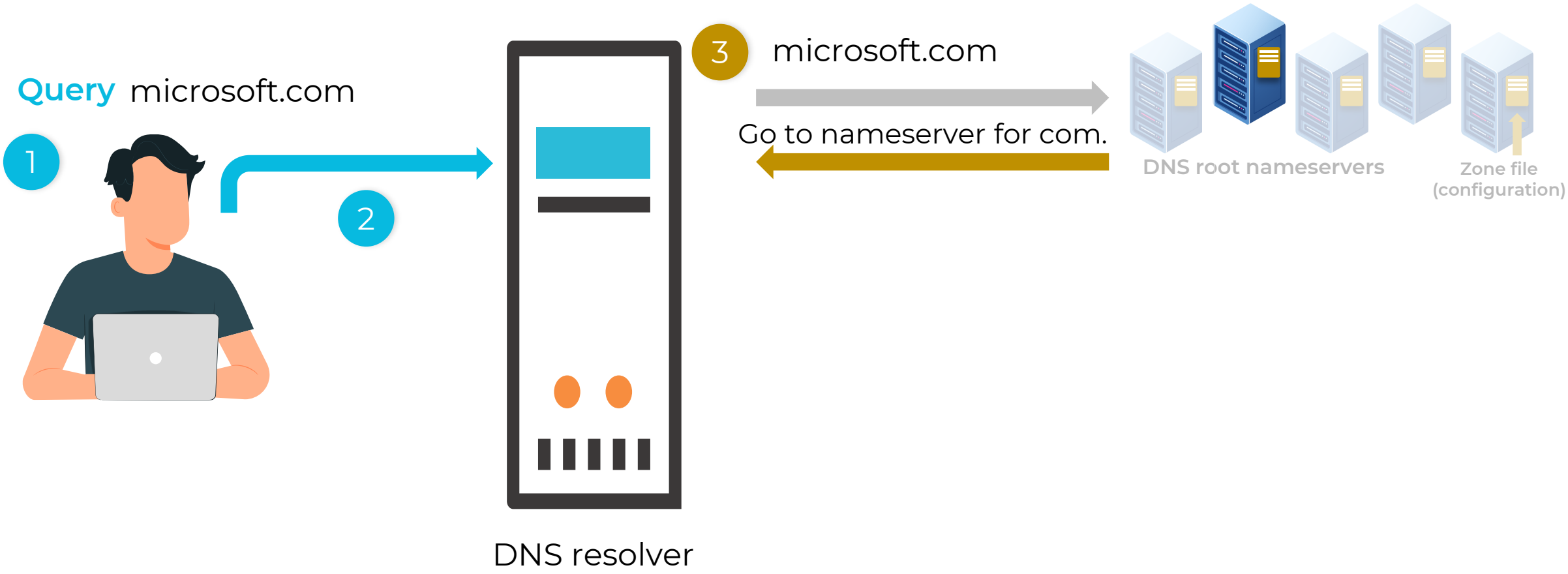
How the Domain Name System Works



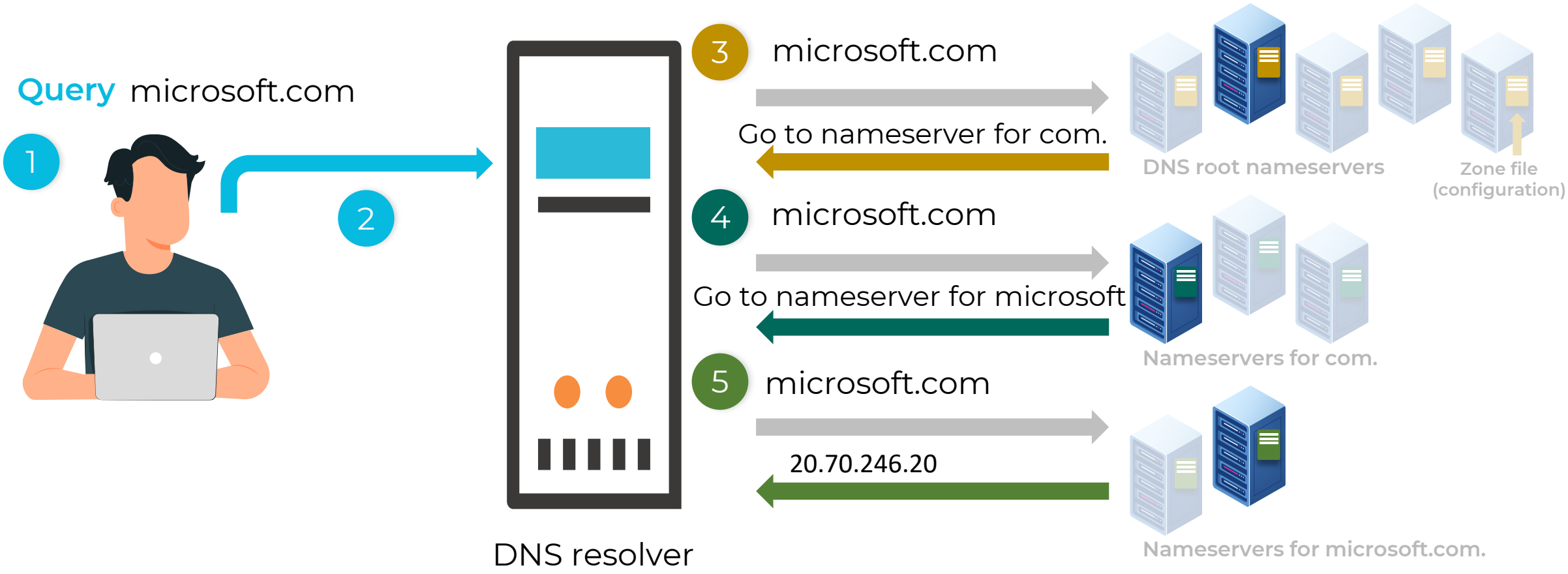
How the Domain Name System Works



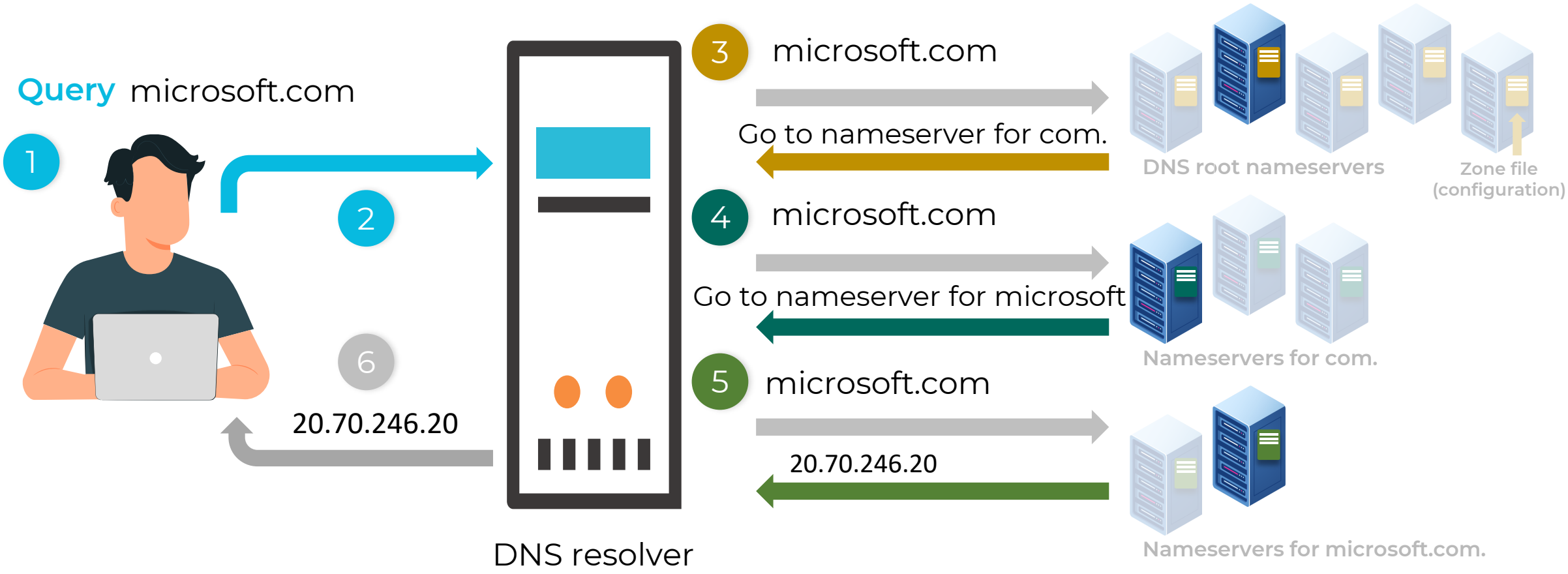
How the Domain Name System Works



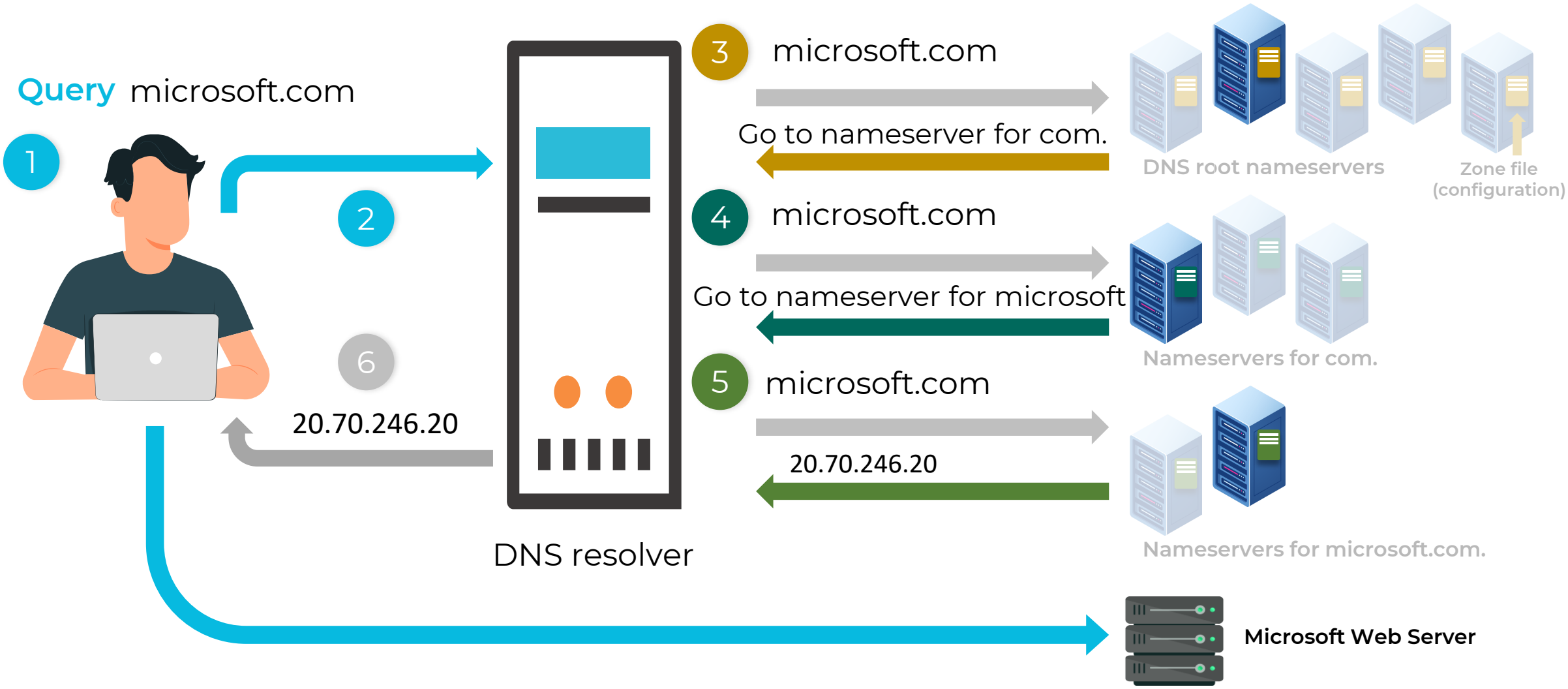
How the Domain Name System Works



How the Domain Name System Works



How the Domain Name System Works



DNS is way more complex than people think!

DNS is way more complex than people think!

Nondeterminism in which nameserver to ask next

DNS is way more complex than people think!

Nondeterminism in which nameserver to ask next

Complex record types each with unique semantics

- DNAME records: domain (partial) rewrite
- CNAME records: alias another domain name
- Wildcard records: match anything not otherwise matched
- NS records: nameserver redirection
- **56** other records types across **~30** RFCs

DNS is way more complex than people think!

Nondeterminism in which nameserver to ask next

Complex record types each with unique semantics

- DNAME records: domain (partial) rewrite
- CNAME records: alias another domain name
- Wildcard records: match anything not otherwise matched
- NS records: nameserver redirection
- **56** other records types across **~30** RFCs

> The DNS is a lot like chess; it's a simple game in terms of the rules, but phenomenally complex in the way it can be played.

Geoff Huston (APNIC)

Our Goal

Automatically generate test cases for DNS nameserver implementations covering as many RFC (specification) behaviors as possible

Our Goal

Automatically generate test cases for DNS nameserver implementations covering as many RFC (specification) behaviors as possible

Challenge – Need to generate config (zone file) and input (query) jointly

Previously Unknown BIND Crash Bug

Tool Generated Test Case

1. Zone file

Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

⟨foo.attack.foo.attack.com.,DNAME⟩ **2. Query**

Previously Unknown BIND Crash Bug

Tool Generated Test Case

1. Zone file

Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

⟨foo.attack.foo.attack.com.,DNAME⟩

2. Query

⟨foo.attack.foo.

attack.com.,DNAME⟩

BIND Server

Previously Unknown BIND Crash Bug

Tool Generated Test Case

Existence of DNAME record

1

1. Zone file

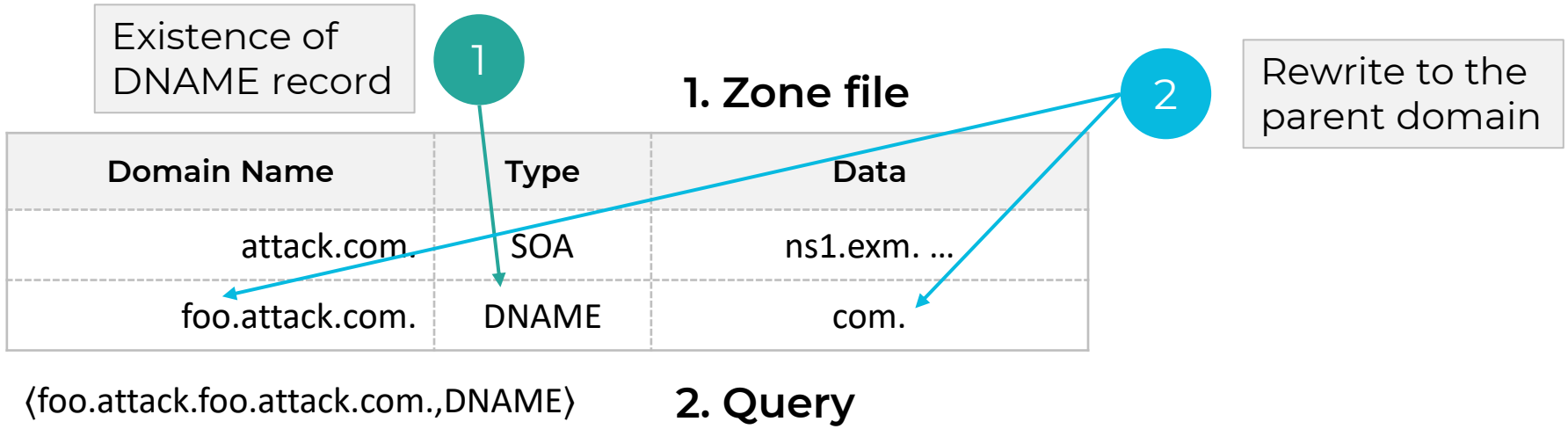
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

⟨foo.attack.foo.attack.com.,DNAME⟩

2. Query

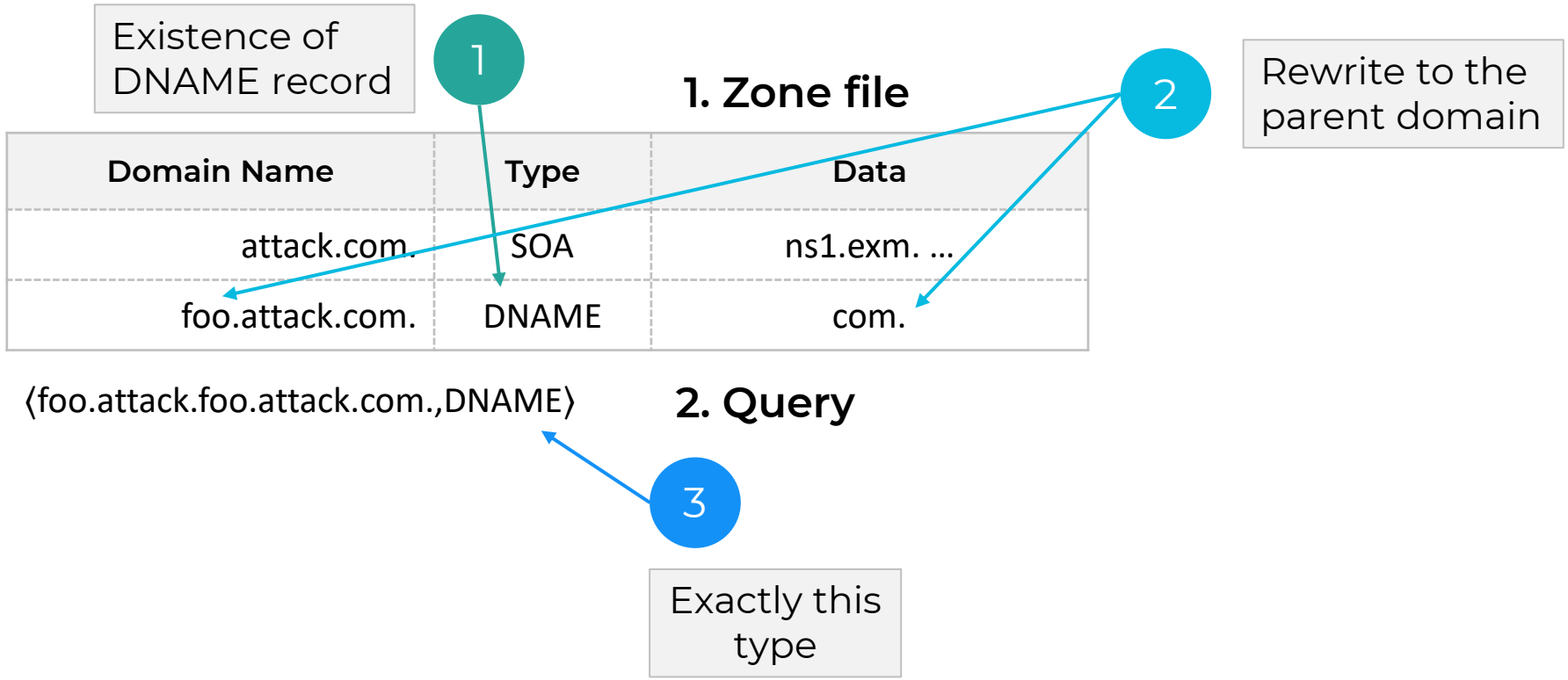
Previously Unknown BIND Crash Bug

Tool Generated Test Case



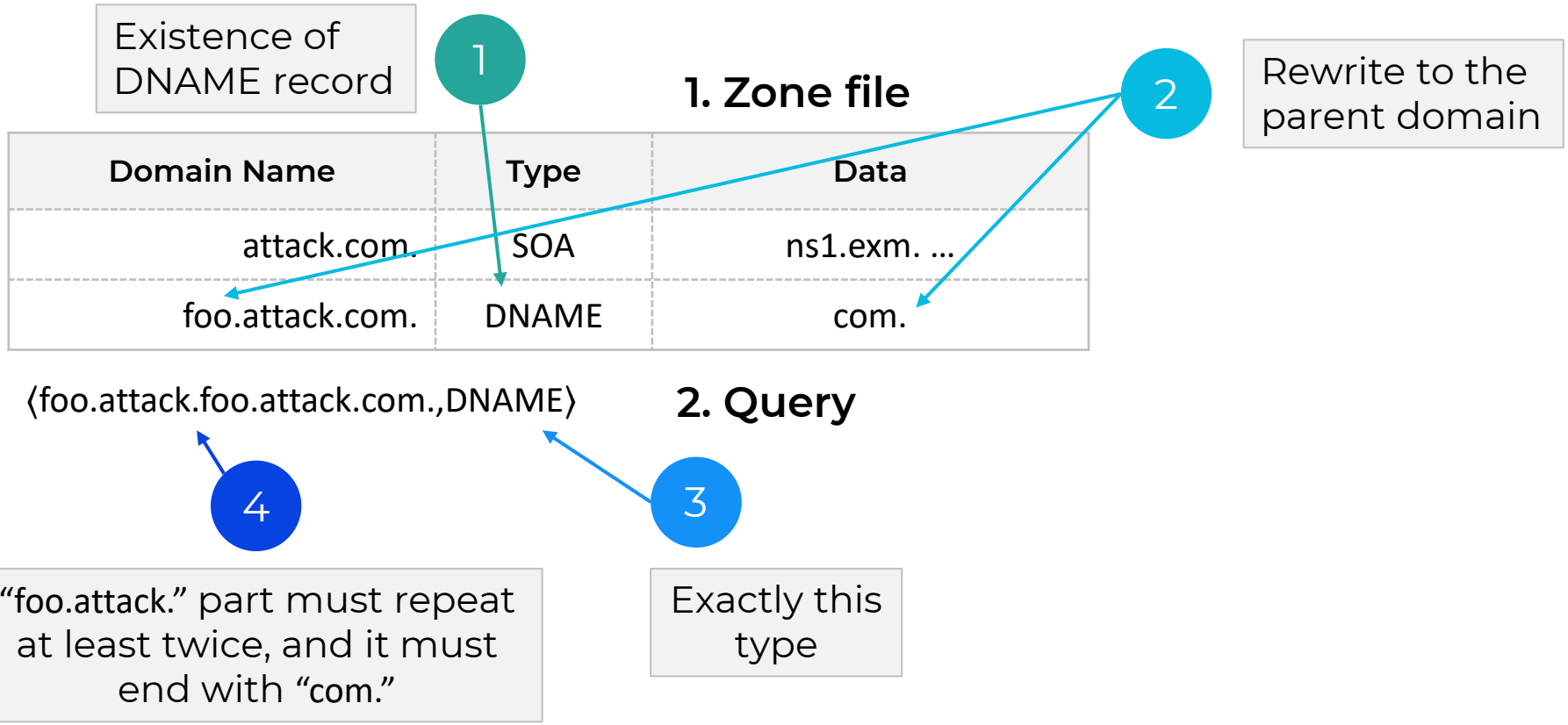
Previously Unknown BIND Crash Bug

Tool Generated Test Case



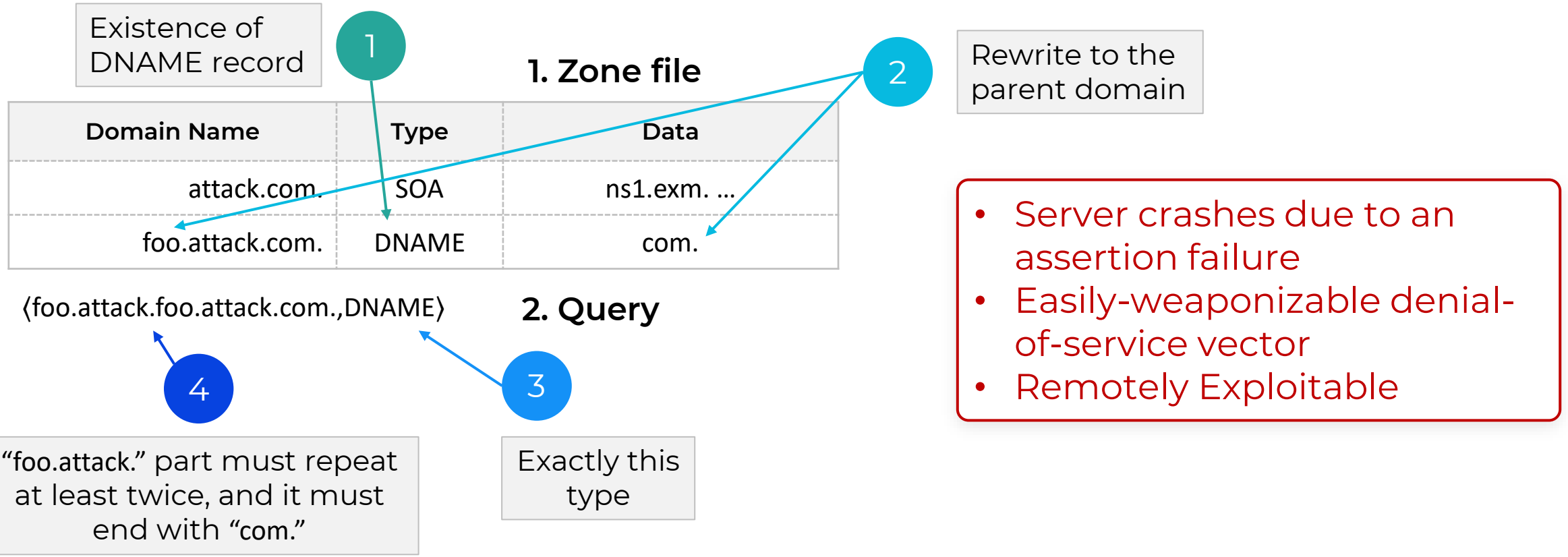
Previously Unknown BIND Crash Bug

Tool Generated Test Case



Previously Unknown BIND Crash Bug

Tool Generated Test Case



- Server crashes due to an assertion failure
- Easily-weaponizable denial-of-service vector
- Remotely Exploitable

BIND Crash Remote Exploitation

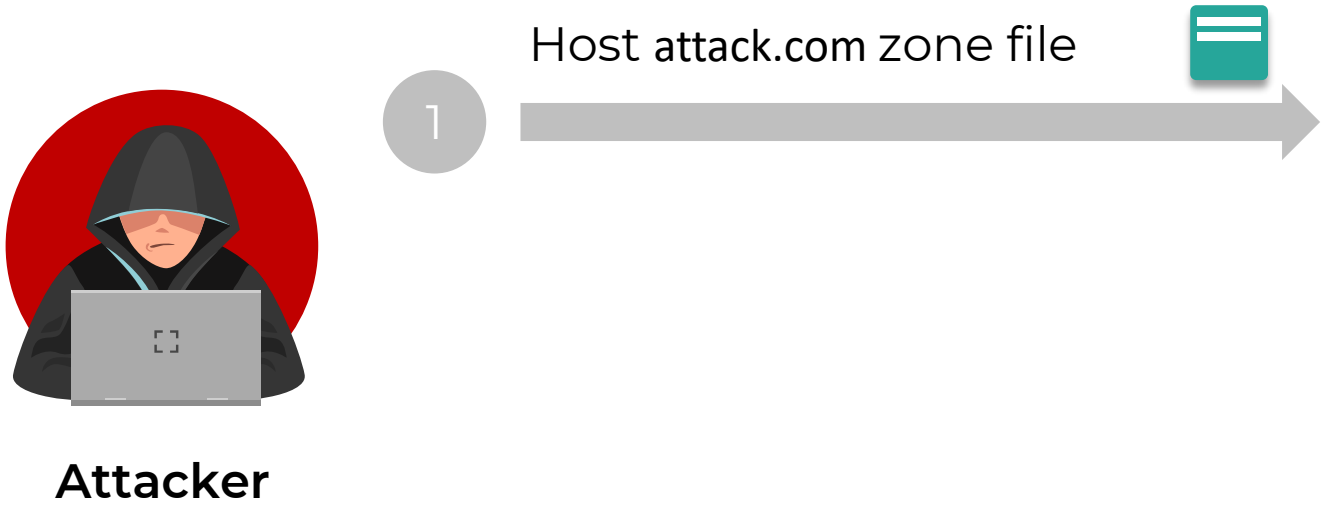
Scenario 1: Attack on a DNS hosting service that uses BIND

Attacker

BIND Crash Remote Exploitation

Scenario 1: Attack on a DNS hosting service that uses BIND

Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.



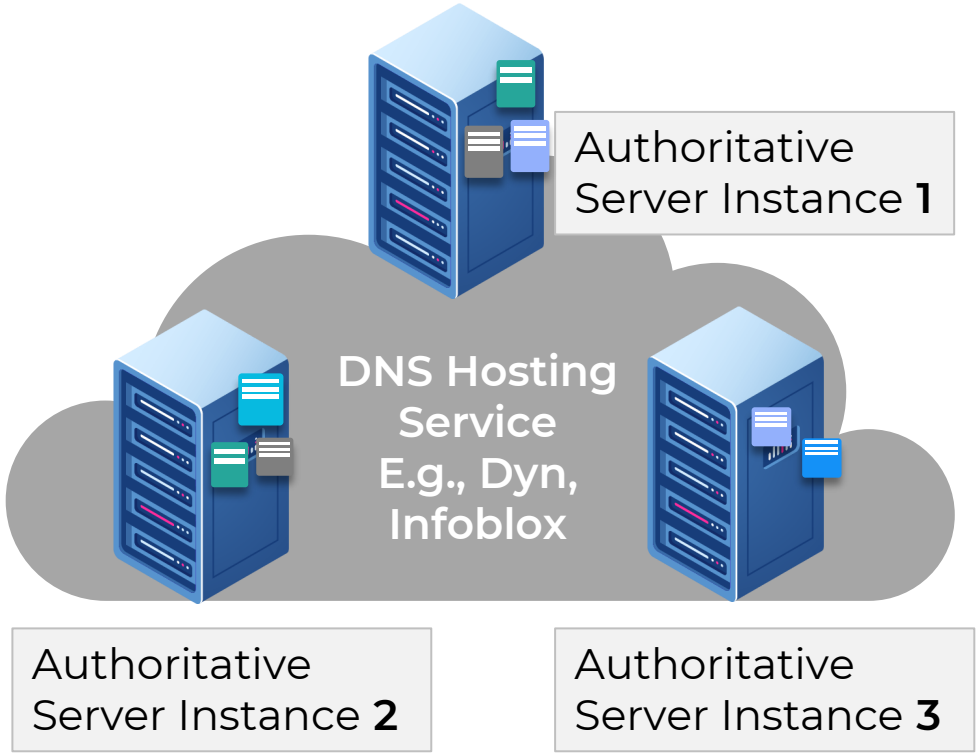
BIND Crash Remote Exploitation

Scenario 1: Attack on a DNS hosting service that uses BIND

Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

Attacker

1 Host attack.com zone file



BIND Crash Remote Exploitation

Scenario 1: Attack on a DNS hosting service that uses BIND

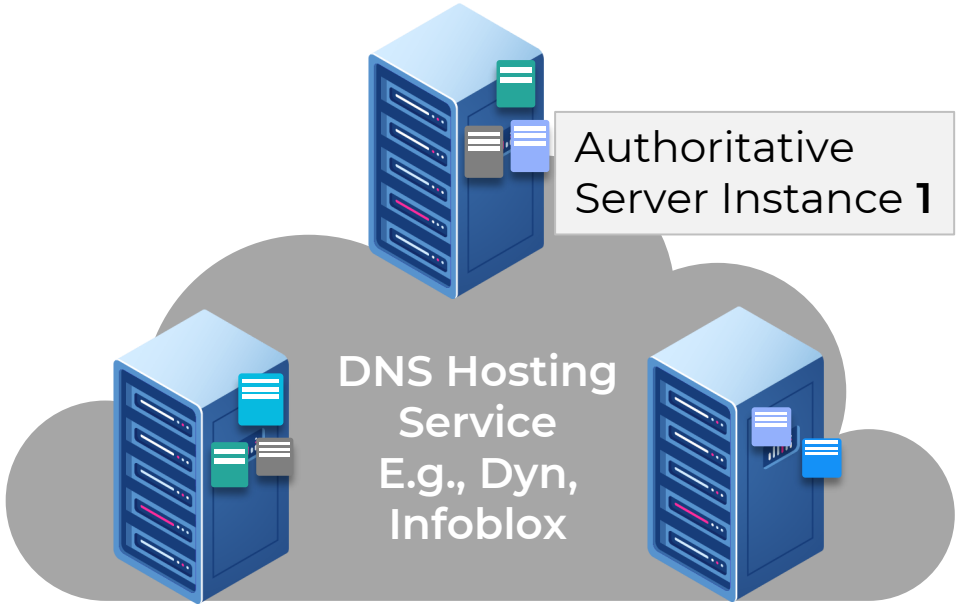
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

Attacker

1 Host attack.com zone file

2 Query for:

⟨foo.attack.foo.attack.com.,DNAME⟩



Authoritative Server Instance **1**

Authoritative Server Instance **2**

Authoritative Server Instance **3**

BIND Crash Remote Exploitation

Scenario 1: Attack on a DNS hosting service that uses BIND

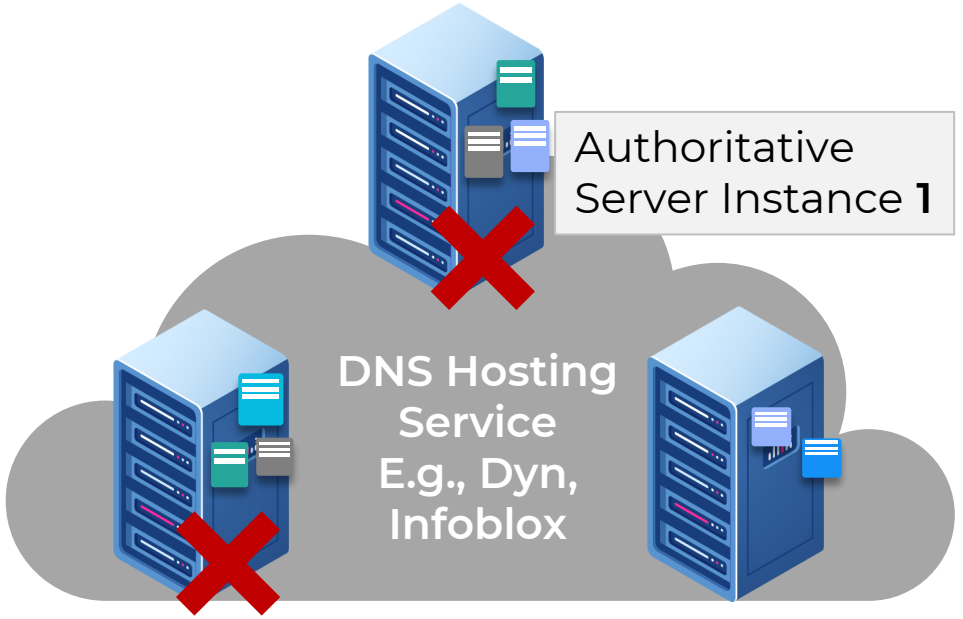
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

Attacker

1 Host attack.com zone file

2 Query for:

⟨foo.attack.foo.attack.com.,DNAME⟩



Crashes and takes down other customer zone files – Remote DoS Attack

BIND Crash Remote Exploitation

Scenario 2: Attack on a public BIND DNS Resolver

Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

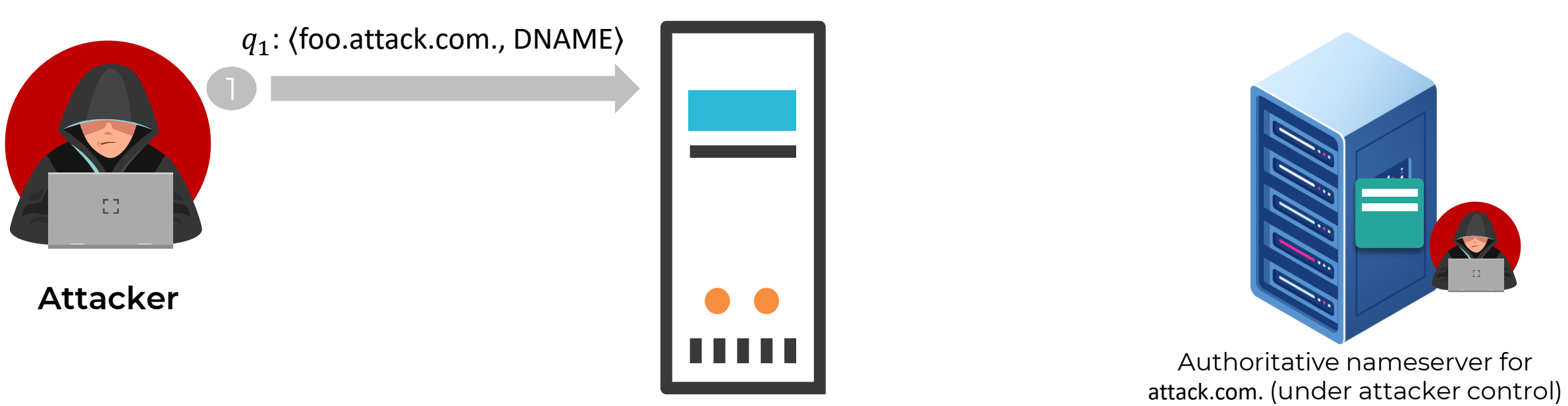
Attacker

Authoritative nameserver for attack.com. (under attacker control)

BIND Crash Remote Exploitation

Scenario 2: Attack on a public BIND DNS Resolver

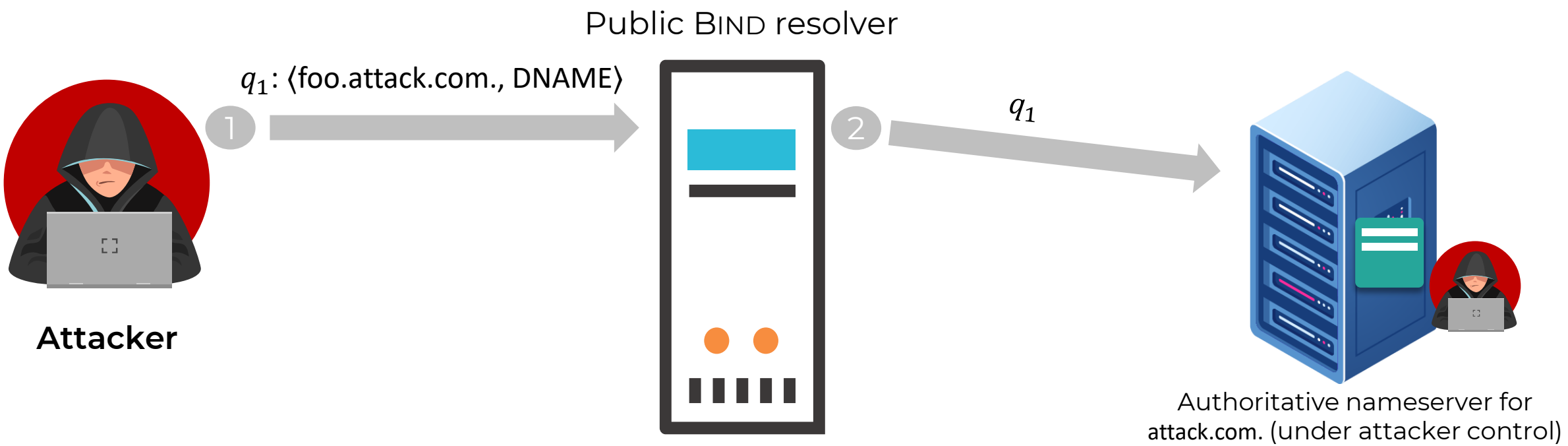
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.



BIND Crash Remote Exploitation

Scenario 2: Attack on a public BIND DNS Resolver

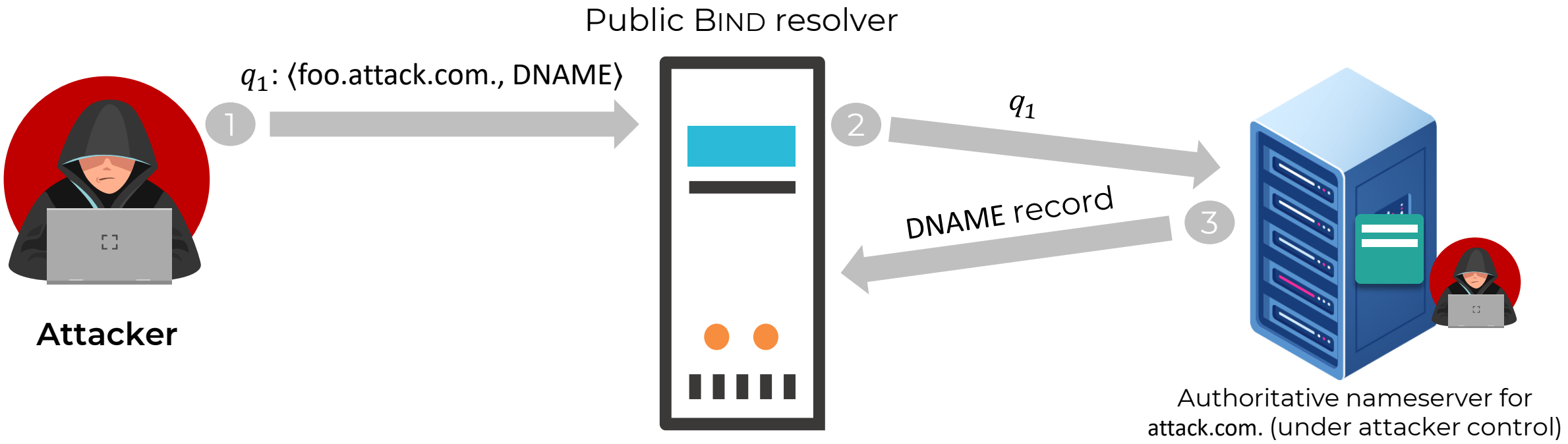
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.



BIND Crash Remote Exploitation

Scenario 2: Attack on a public BIND DNS Resolver

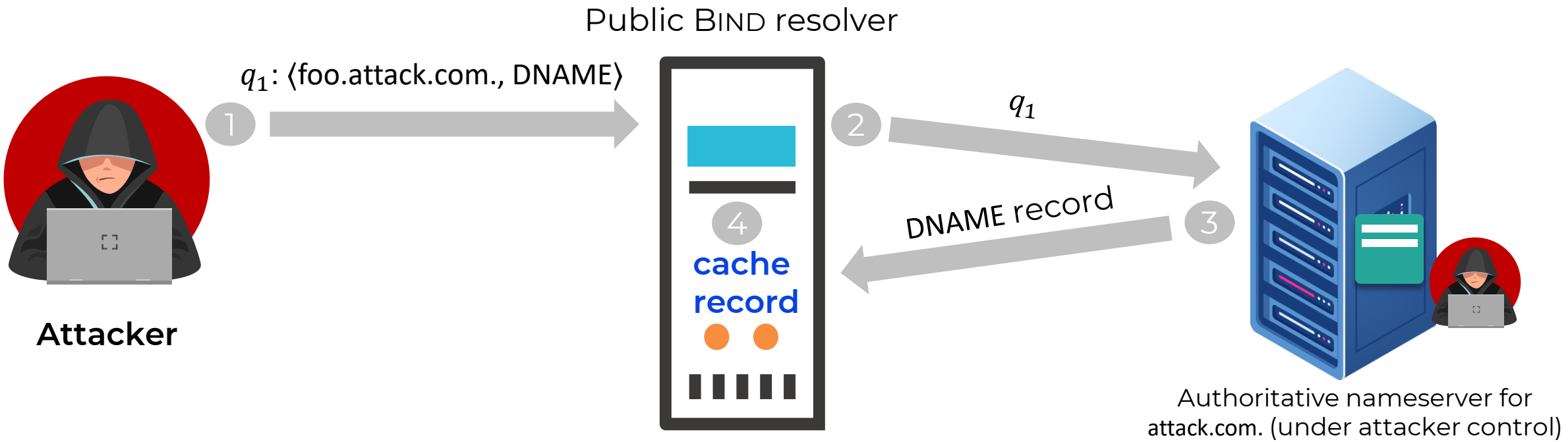
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.



BIND Crash Remote Exploitation

Scenario 2: Attack on a public BIND DNS Resolver

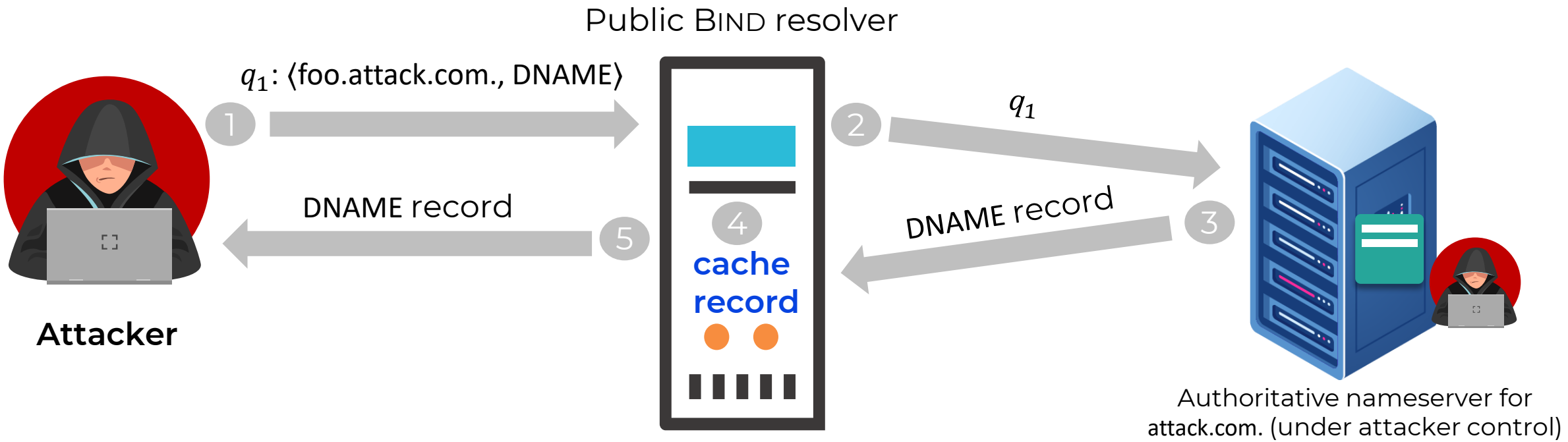
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.



BIND Crash Remote Exploitation

Scenario 2: Attack on a public BIND DNS Resolver

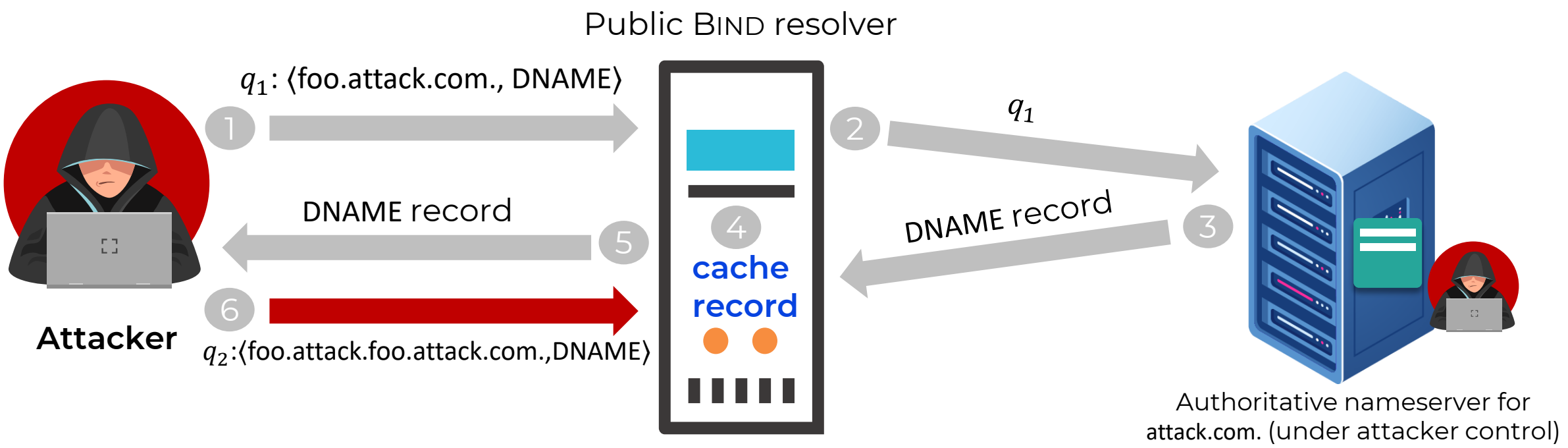
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.



BIND Crash Remote Exploitation

Scenario 2: Attack on a public BIND DNS Resolver

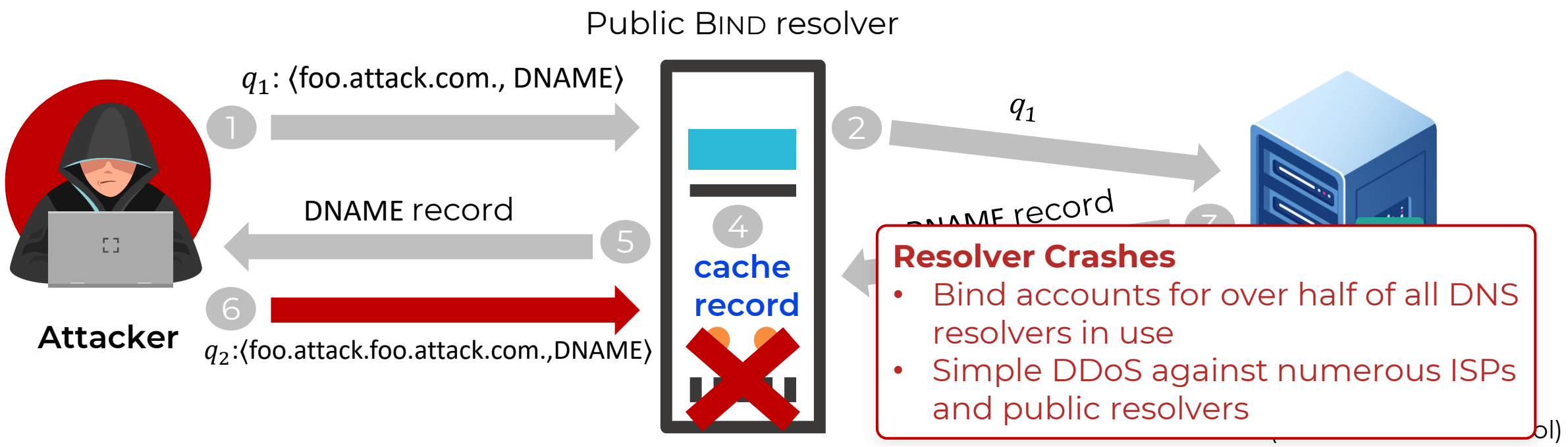
Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.



BIND Crash Remote Exploitation

Scenario 2: Attack on a public BIND DNS Resolver

Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.



BIND Crash Remote Exploitation

Scenario 1: Attack on a DNS hosting service that uses BIND

Scenario 2: Attack on a public BIND DNS Resolver

BIND Crash Disclosure

Scenario 1: Attack on a DNS hosting service that uses BIND

Initiated a responsible disclosure with BIND

Scenario 2: Attack on a public BIND DNS Resolver

BIND Crash Disclosure

Scenario 1: Attack on a DNS hosting service that uses BIND

Initiated a responsible disclosure with BIND

Scenario 2: Attack on a public BIND DNS Resolver

CVE: CVE-2021-25215

Document version: 2.0

Posting date: 28 April 2021

Program impacted: BIND

Versions affected: BIND 9.0.0 -> 9.11.2
BIND Supported Preview Edition, as well

Severity: High

Exploitable: Remotely

Description:

DNAME records, described in RFC 6672,

BIND Crash Disclosure

Scenario 1: Attack on a DNS hosting service that uses BIND

Initiated a responsible disclosure with BIND

Scenario 2: Attack on a public BIND DNS Resolver

Affected all maintained BIND versions affecting NetApp, Ubuntu, Infoblox, and Red Hat.

CVE: CVE-2021-25215

Document version: 2.0

Posting date: 28 April 2021

Program impacted: BIND

Versions affected: BIND 9.0.0 -> 9.11.2
BIND Supported Preview Edition, as well

Severity: High

Exploitable: Remotely

Description:

DNAME records, described in RFC 6672,

Previously Unknown BIND Crash Bug

Tool Generated Test Case

1. Zone file

Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

⟨foo.attack.foo.attack.com.,DNAME⟩ **2. Query**

⟨foo.attack.foo.

attack.com.,DNAME⟩

BIND Server

Previously Unknown BIND Crash Bug

Tool Generated Test Case

1. Zone file

Domain Name	Type	Data
attack.com.	SOA	ns1.exm. …
foo.attack.com.	DNAME	com.

⟨foo.attack.foo.attack.com.,DNAME⟩ **2. Query**

⟨foo.attack.foo.

attack.com.,DNAME⟩

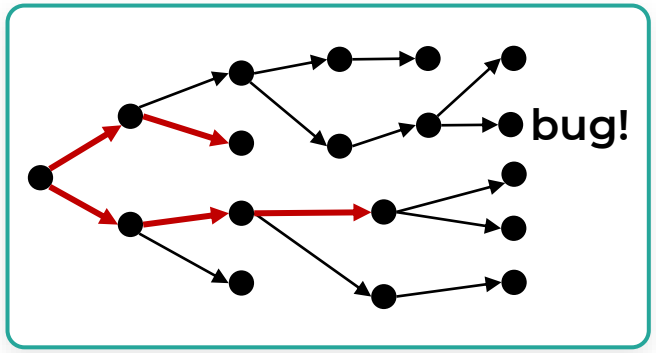
BIND Server

Joint auto generation of query and zone file is required

Standard Automated Testers are Insufficient

Fuzz testing for DNS Implementations

- Scalable to large codebases
- Can't navigate complex semantic requirements and dependencies to generate zone files
- **Generates queries only to check zone file parsers**
- No coverage guarantees



DNS Nameserver Implementation (BIND)

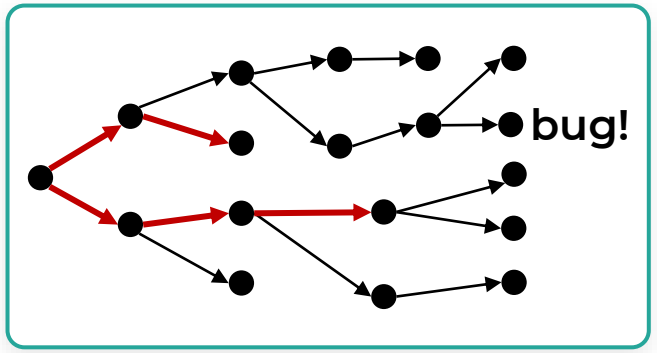
Standard Automated Testers are Insufficient

Fuzz testing for DNS Implementations

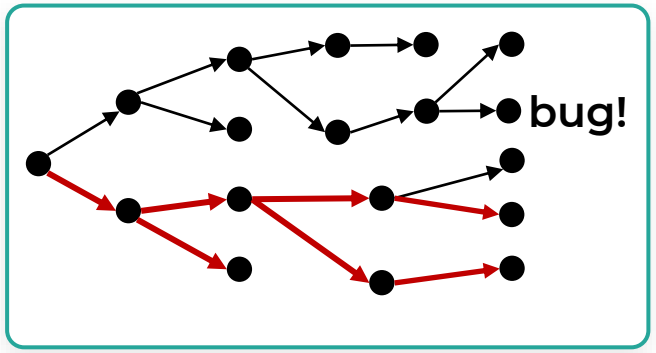
- ✓ Scalable to large codebases
- ✗ Can't navigate complex semantic requirements and dependencies to generate zone files
- ✗ **Generates queries only to check zone file parsers**
- ✗ No coverage guarantees

Symbolic execution for DNS Implementations

- ✓ Solves for path input conditions
- ✗ Path explosion and difficulty with complex data structures
- ✗ Explores a subset of implementation paths
- ✗ **Coverage guarantees in theory**



DNS Nameserver Implementation (BIND)



Standard Automated Testers are Insufficient

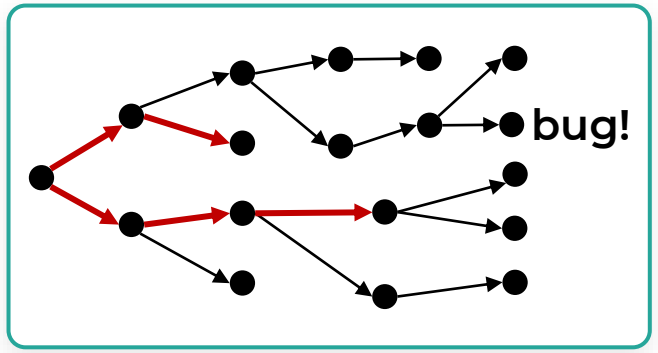
Fuzz testing for DNS Implementations

- ✓ Scalable to large codebases
- ✗ Can't navigate complex semantic requirements and dependencies to generate zone files
- ✗ **Generates queries only to check zone file parsers**
- ✗ No coverage guarantees

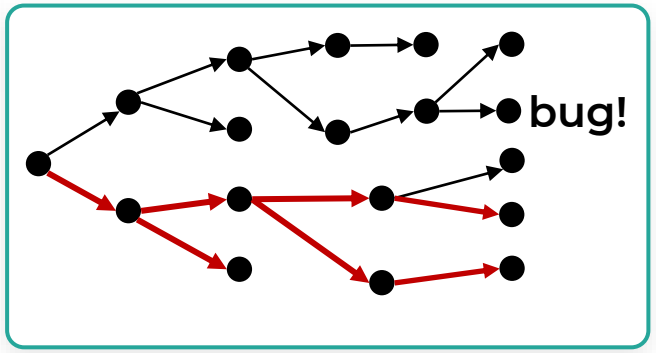
Symbolic execution for DNS Implementations

- ✓ Solves for path input conditions
- ✗ Path explosion and difficulty with complex data structures
- ✗ Explores a subset of implementation paths
- ✗ **Coverage guarantees in theory**

Current automated testers for DNS do not generate zone files and hence do not find RFC violations



DNS Nameserver Implementation (BIND)



Our Approach

- **S** Small-scope
- **C** Constraint-driven
- **A** Automated
- **L** Logical
- **E** Execution

Our Approach

S Small-scope

C Constraint-driven

A Automated

L Logical

E Execution

- **Jointly** generates zone files & queries
- Covers many **different** RFC behaviors
- Applicable to **black-box** implementations

Our Approach

S Small-scope

C Constraint-driven

A Automated

L Logical

E Execution

- **Jointly** generates zone files & queries
- Covers many **different** RFC behaviors
- Applicable to **black-box** implementations

Specification of DNS
RFCs 1034, 4592, 6672, ...

Our Insight

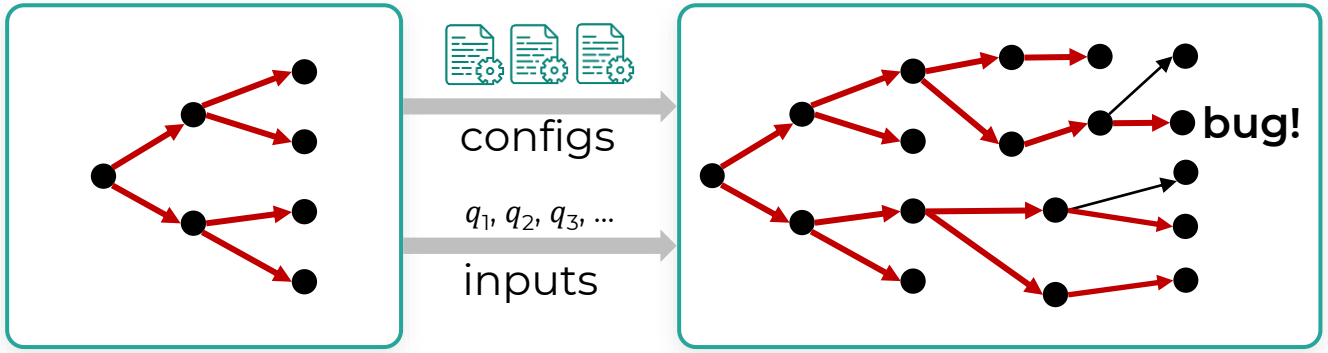
- **S** Small-scope
- **C** Constraint-driven
- **A** Automated
- **L** Logical
- **E** Execution

Use DNS formal model to guide test generation

Our Insight

- **S** Small-scope
- **C** Constraint-driven
- **A** Automated
- **L** Logical
- **E** Execution

Use DNS formal model to guide test generation



DNS logical model from RFCs

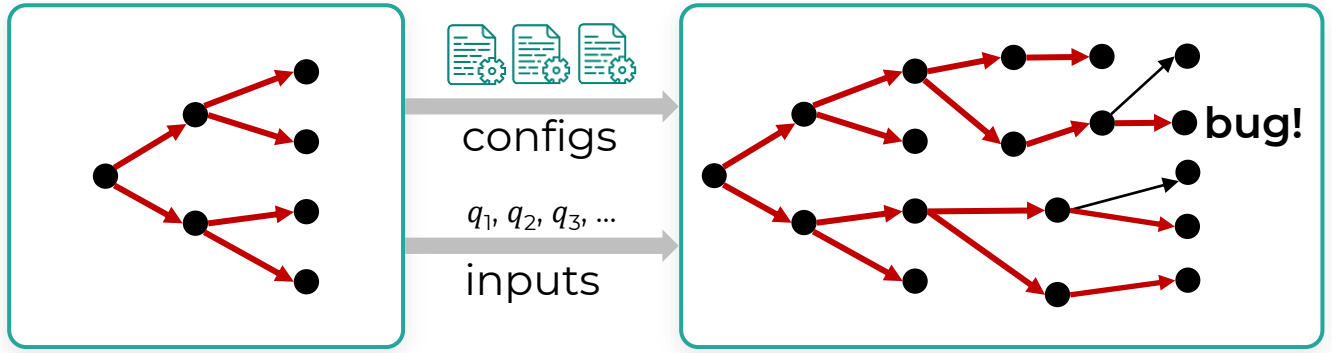
DNS Nameserver Implementation (BIND)

Our Insight

- **S** Small-scope
- **C** Constraint-driven
- **A** Automated
- **L** Logical
- **E** Execution

Use DNS formal model to guide test generation

High RFC behavior coverage – Tests cover all return points (different RFC scenarios) in the logical model

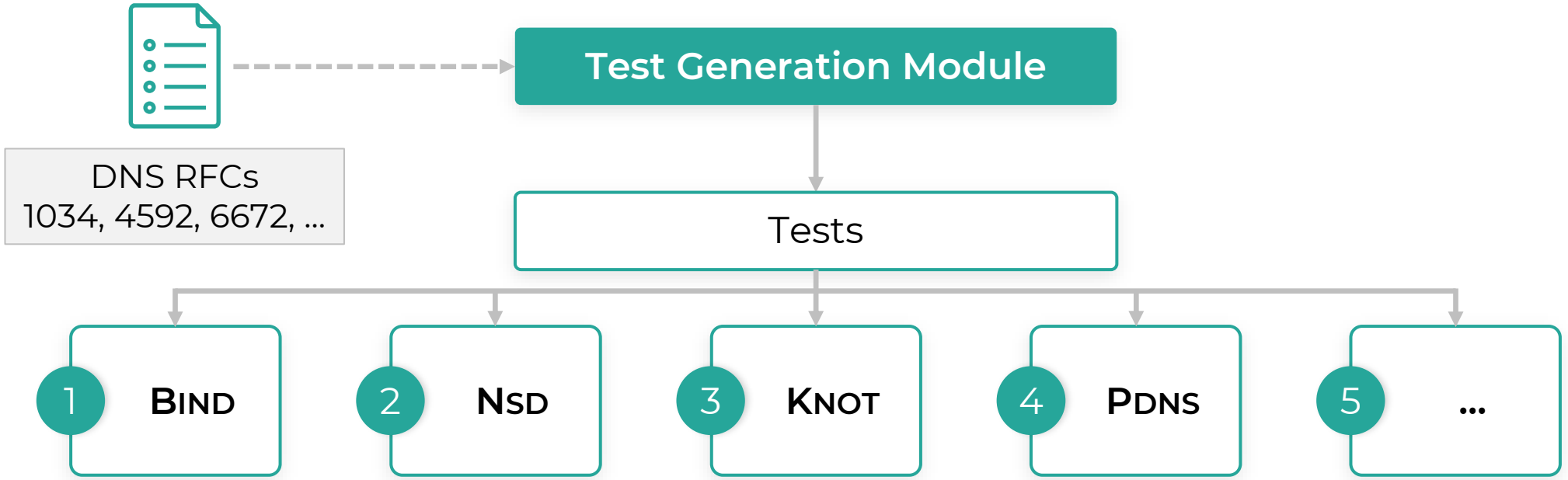


DNS logical model from RFCs

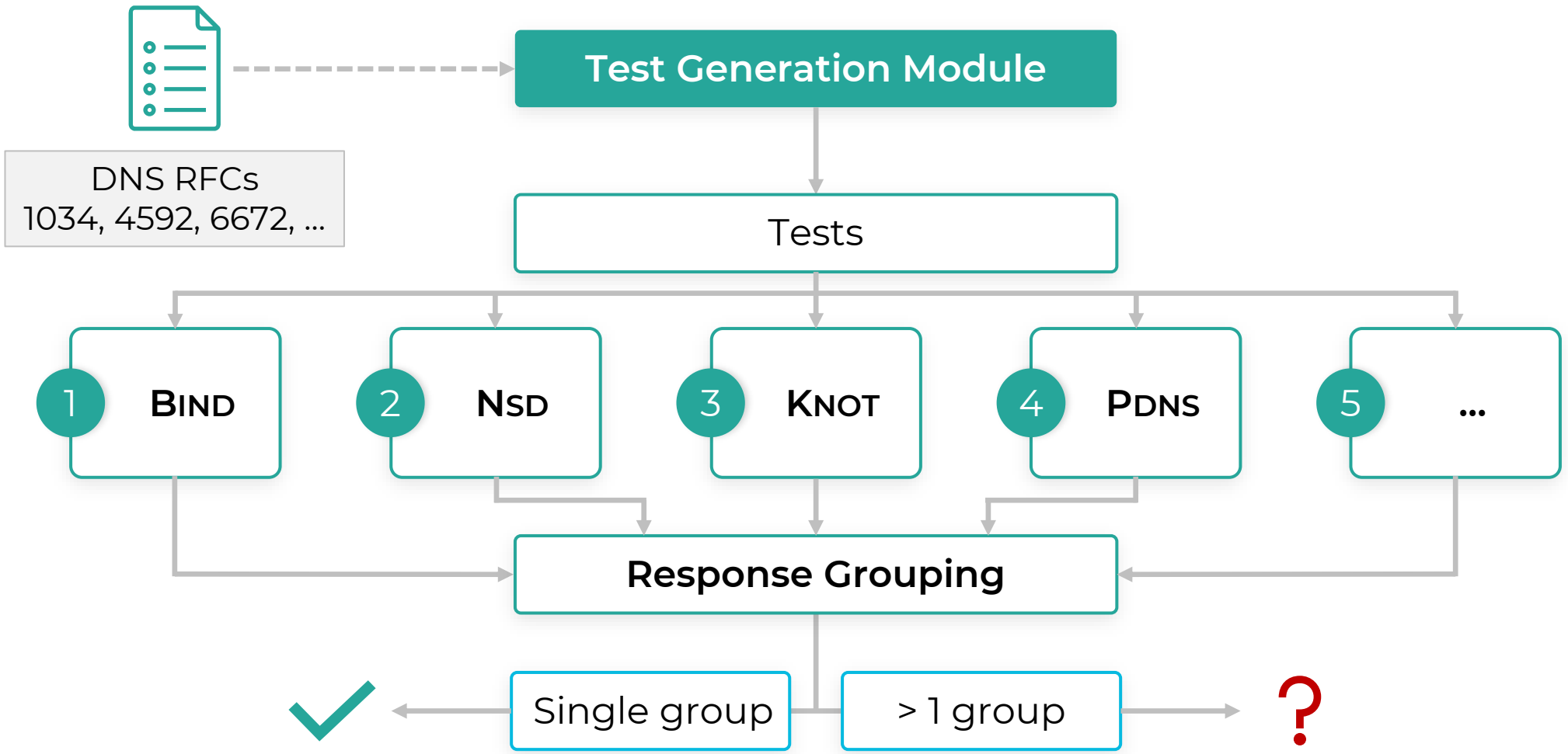
DNS Nameserver Implementation (BIND)

FERRET: Tool based on SCALE for DNS

FERRET: Tool based on SCALE for DNS

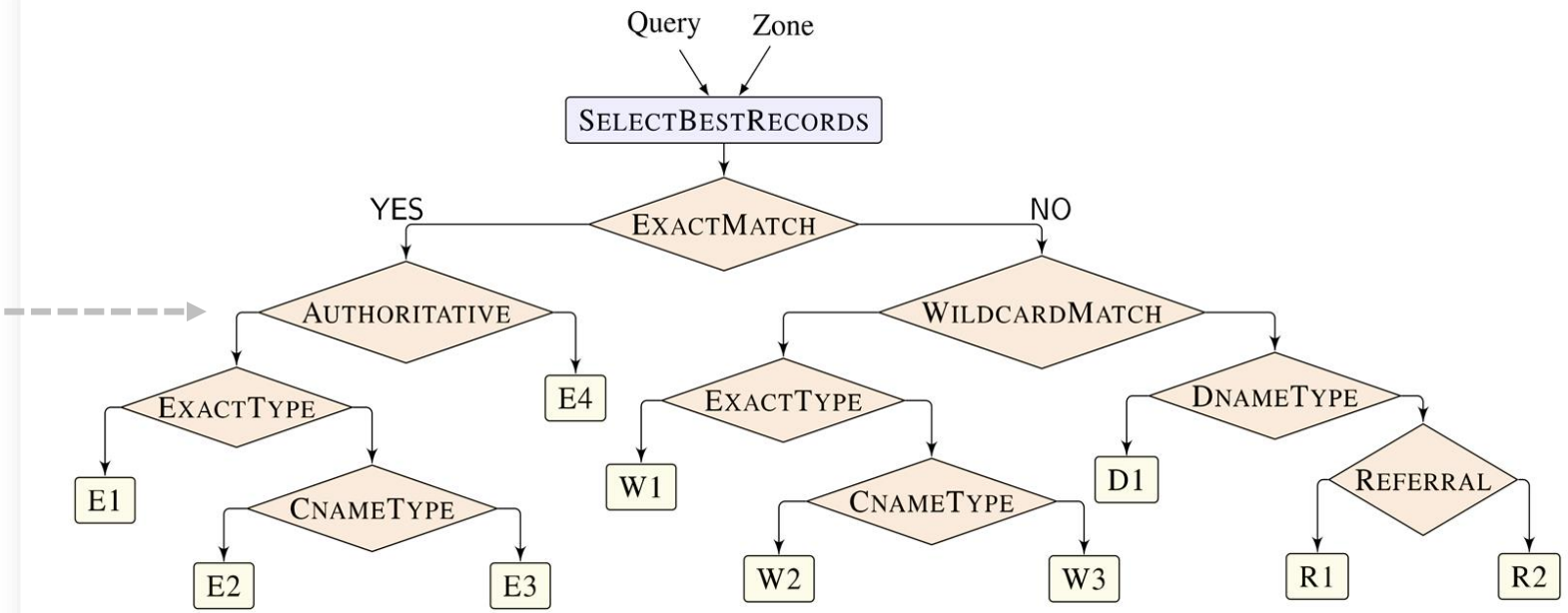


FERRET: Tool based on SCALE for DNS



Test Generation Module

Formal Model †



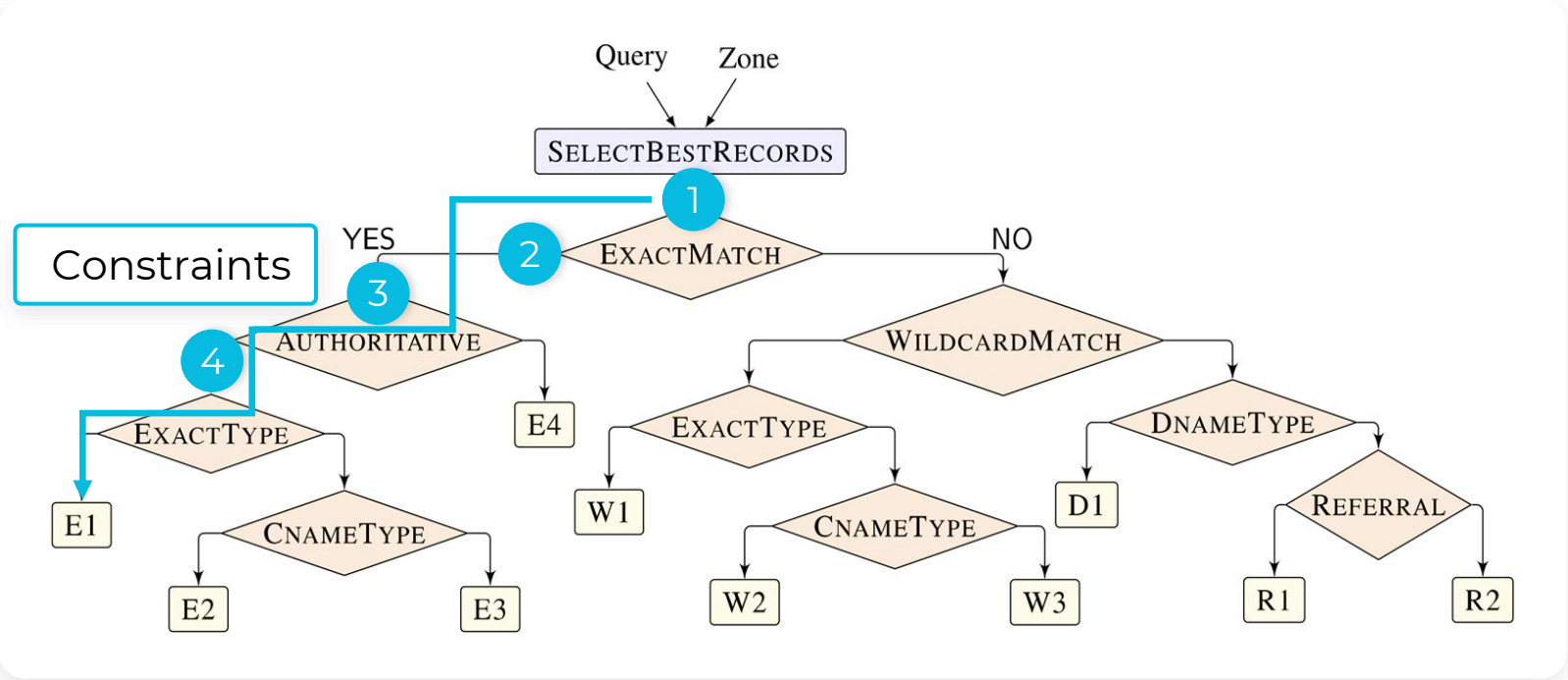
RFCs
1034, 4592, 6672, ...

English → Declarative (Mathematical) specification of the nameserver logic

† GROOT: Proactive Verification of DNS Configurations – Siva Kakarla et al., SIGCOMM 2020

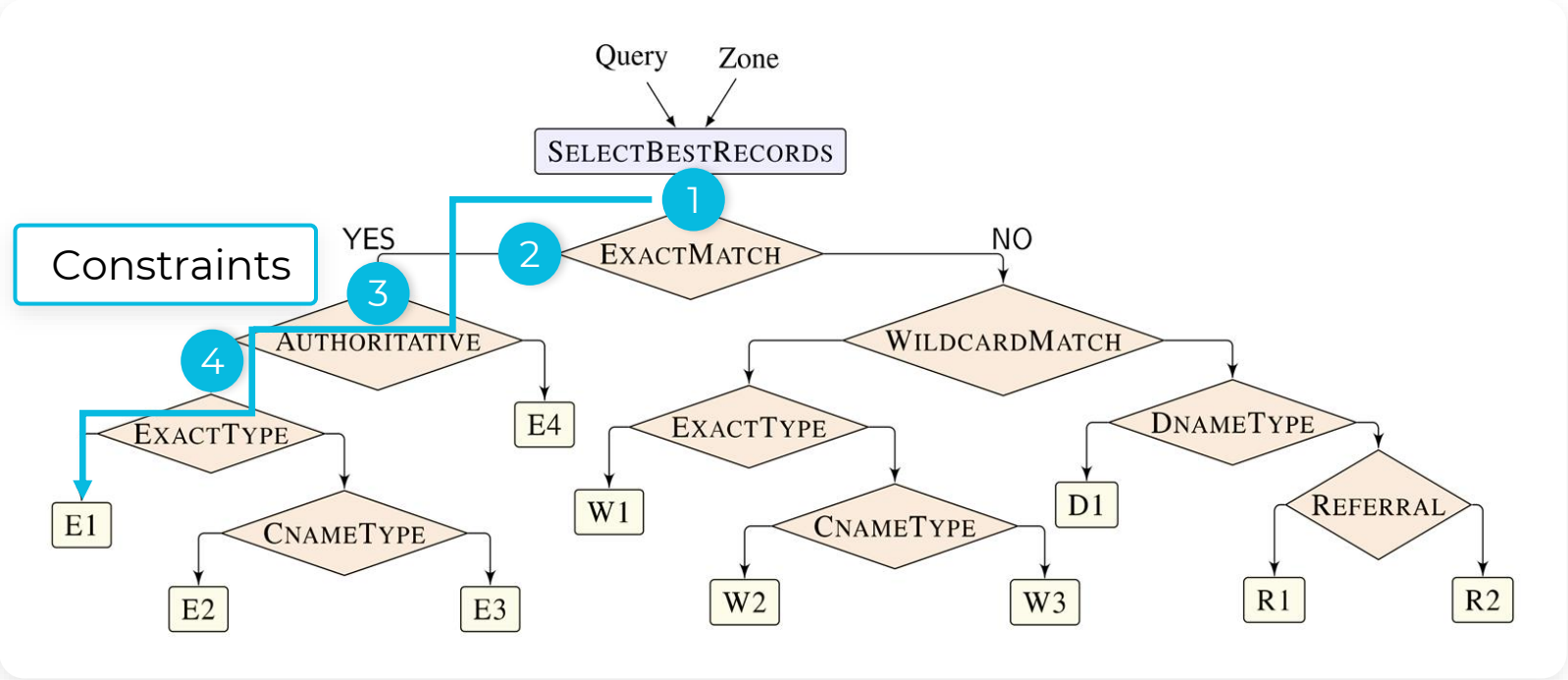
Test Generation Module

Formal Model



Test Generation Module

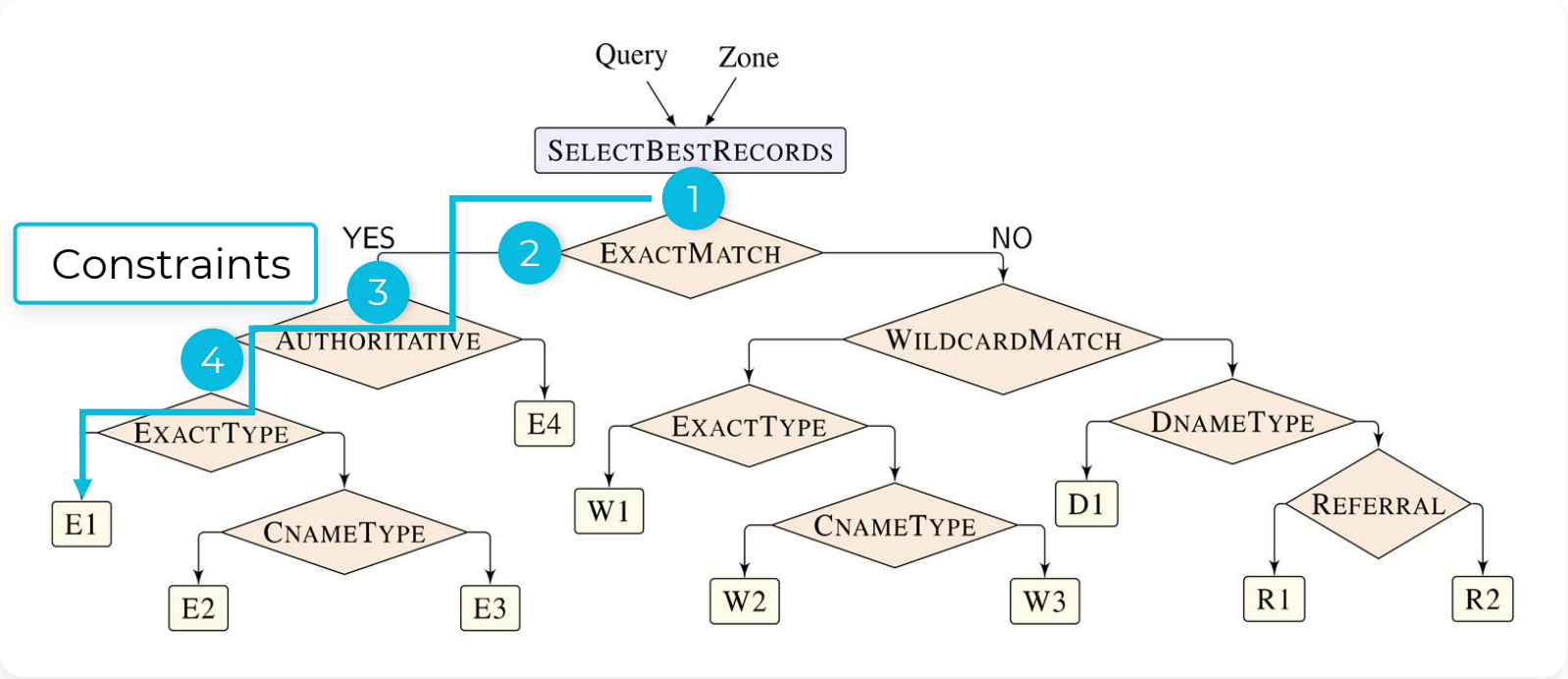
Formal Model



Solve (1,2,3,4) for inputs → (z, q)

Test Generation Module

Formal Model



Constraints

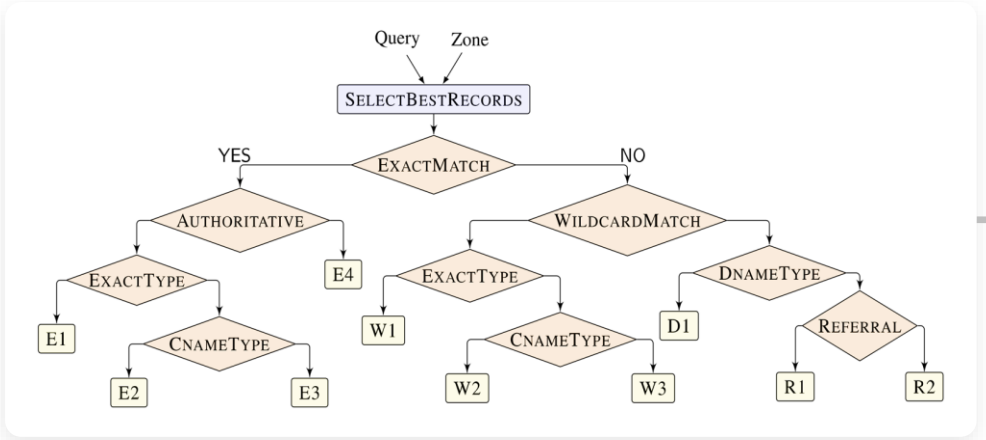
Symbolic Execution

Solve (1,2,3,4) for inputs → (z, q)

Test Generation Module

Formal Model

Executable version in Zen



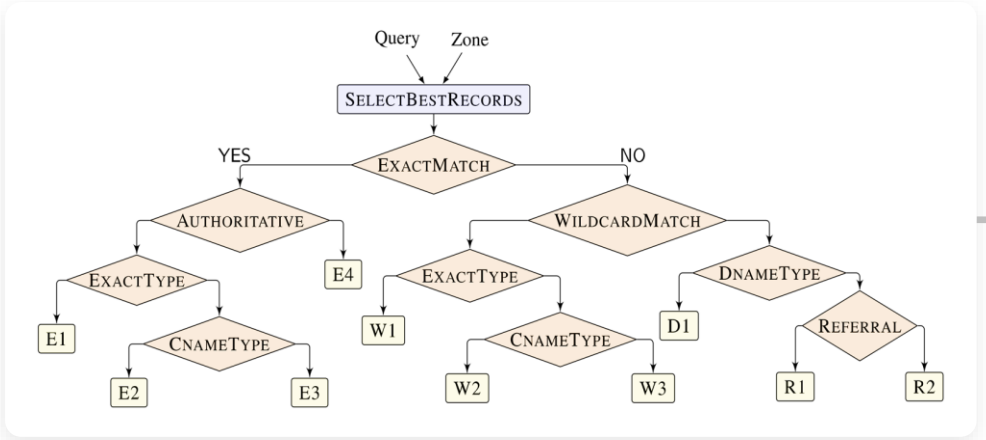

```
Zen<Response> QueryLookup(
  Zen<Query> q,
  Zen<Zone> z)
{
  var records = SelectBestRecords(q, z);
  var rname = records.At(0).Value().Name();
  var types = records.Select(r => r.Type());

  return If(
    rname == q.Name(),
    ExactMatch(records, q, z),
    If(
      IsWildcardMatch(q.Name(), rname),
      WildcardMatch(records, q, z),
      If(
        types.Any(t => t == RType.DNAME),
        Rewrite(records, q),
        If(
          And(types.Any(t => t == RType.NS),
              Not(types.Any(t => t == RType.SOA))),
          Response(Tag.R1,
            Delegation(records, z), Null<Query>()),
          Response(Tag.R2, empty, Null<Query>())
    ))));
}
```

Test Generation Module

Formal Model

Executable version in Zen




```
Zen<Response> QueryLookup(
  Zen<Query> q,
  Zen<Zone> z)
{
  var records = SelectBestRecords(q, z);
  var rname = records.At(0).Value().Name();
  var types = records.Select(r => r.Type());

  return If(
    rname == q.Name(),
    ExactMatch(records, q, z),
    If(
      IsWildcardMatch(q.Name(), rname),
      WildcardMatch(records, q, z),
      If(
        types.Any(t => t == RType.DNAME),
        Rewrite(records, q),
        If(
          And(types.Any(t => t == RType.NS),
              Not(types.Any(t => t == RType.SOA))),
          Response(Tag.R1,
            Delegation(records, z), Null<Query>()),
          Response(Tag.R2, empty, Null<Query>())
    ))));
}
```

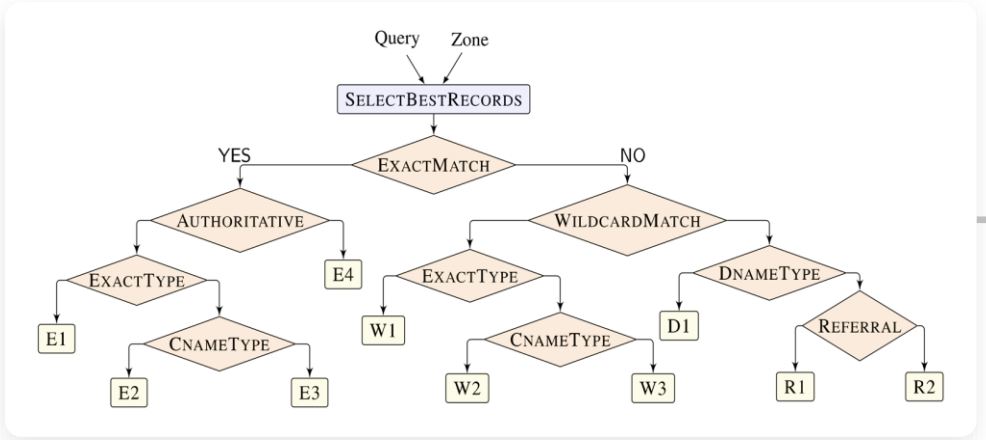
An **executable version** of formal model is implemented in **Zen**, a domain-specific modeling language embedded in **C#** with built-in support for **symbolic execution**

Test Generation Module

Formal Model

Executable version in Zen

Tests




```
Zen<Response> QueryLookup(
  Zen<Query> q,
  Zen<Zone> z)
{
  var records = SelectBestRecords(q, z);
  var rname = records.At(0).Value().Name();
  var types = records.Select(r => r.Type());

  return If(
    rname == q.Name(),
    ExactMatch(records, q, z),
    If(
      IsWildcardMatch(q.Name(), rname),
      WildcardMatch(records, q, z),
      If(
        types.Any(t => t == RType.DNAME),
        Rewrite(records, q),
        If(
          And(types.Any(t => t == RType.NS),
              Not(types.Any(t => t == RType.SOA))),
          Response(Tag.R1,
            Delegation(records, z), Null<Query>()),
          Response(Tag.R2, empty, Null<Query>())
    ))));
}
```

Symbolic Execution

$\langle$zone file$_1$, query$_1\rangle$

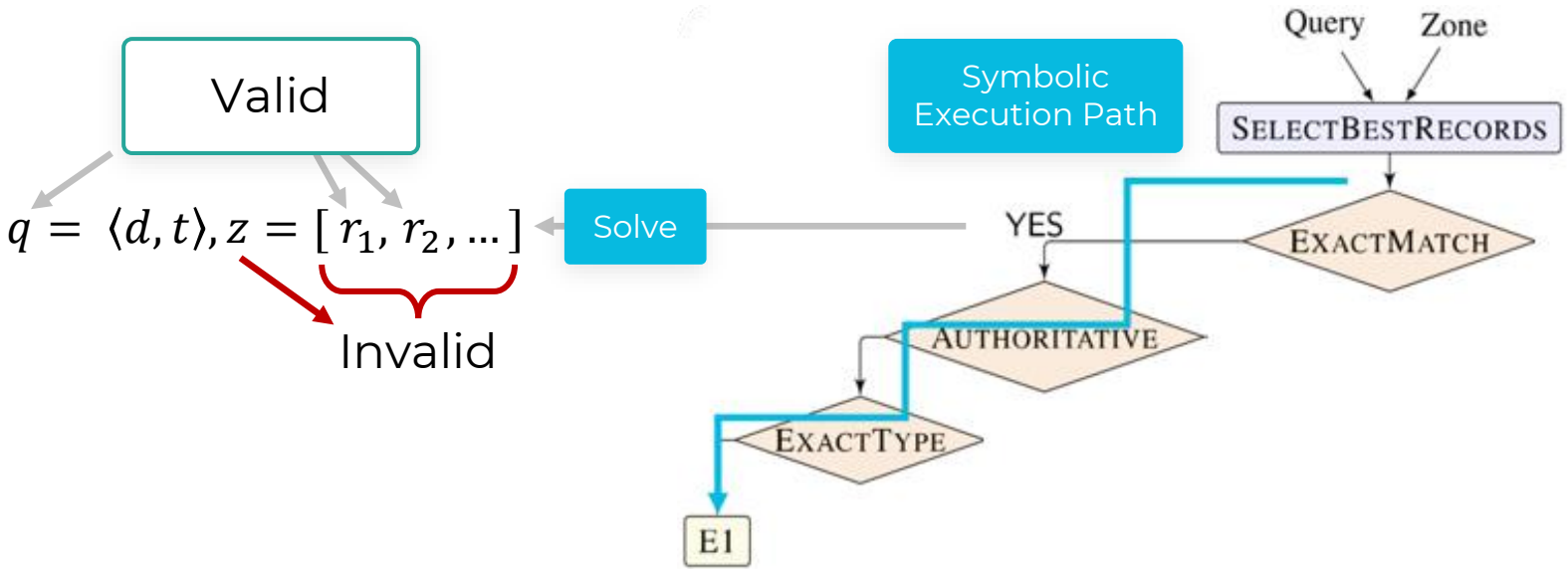
$\langle$zone file$_2$, query$_2\rangle$

...

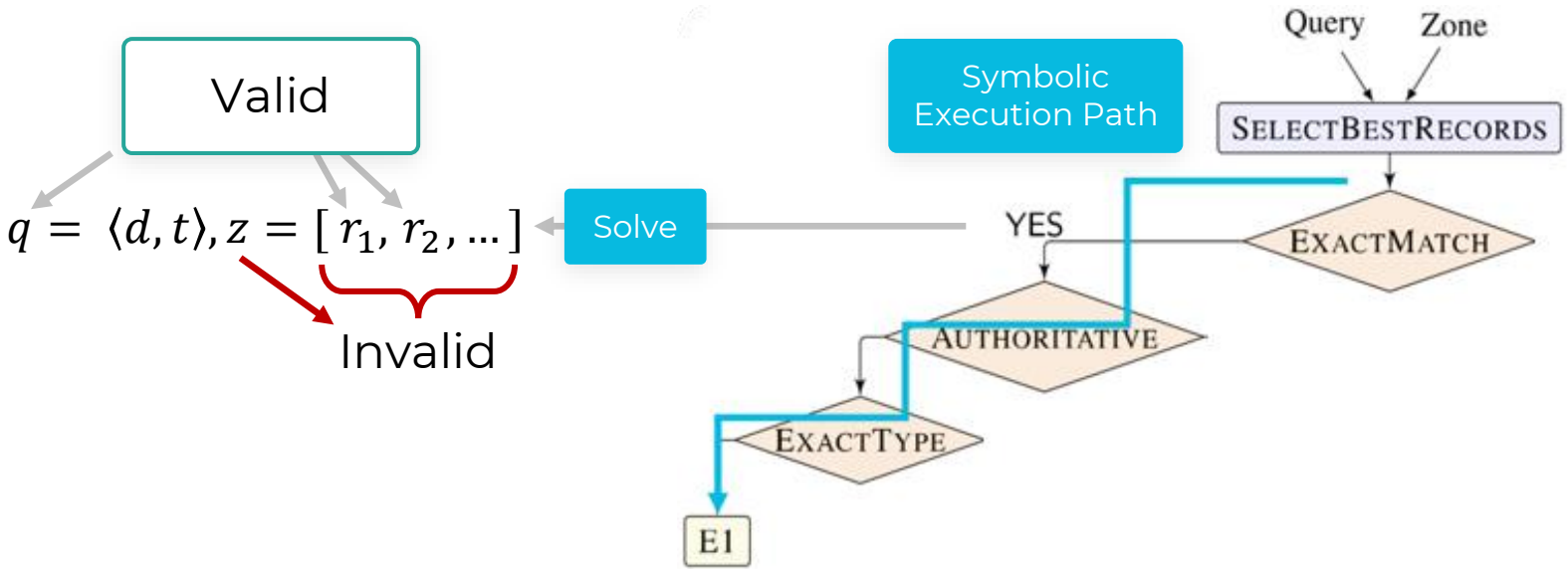
$\langle$zone file$_n$, query$_n\rangle$

An **executable version** of formal model is implemented in **Zen**, a domain-specific modeling language embedded in **C#** with built-in support for **symbolic execution**

Challenge – Generating Valid Zones

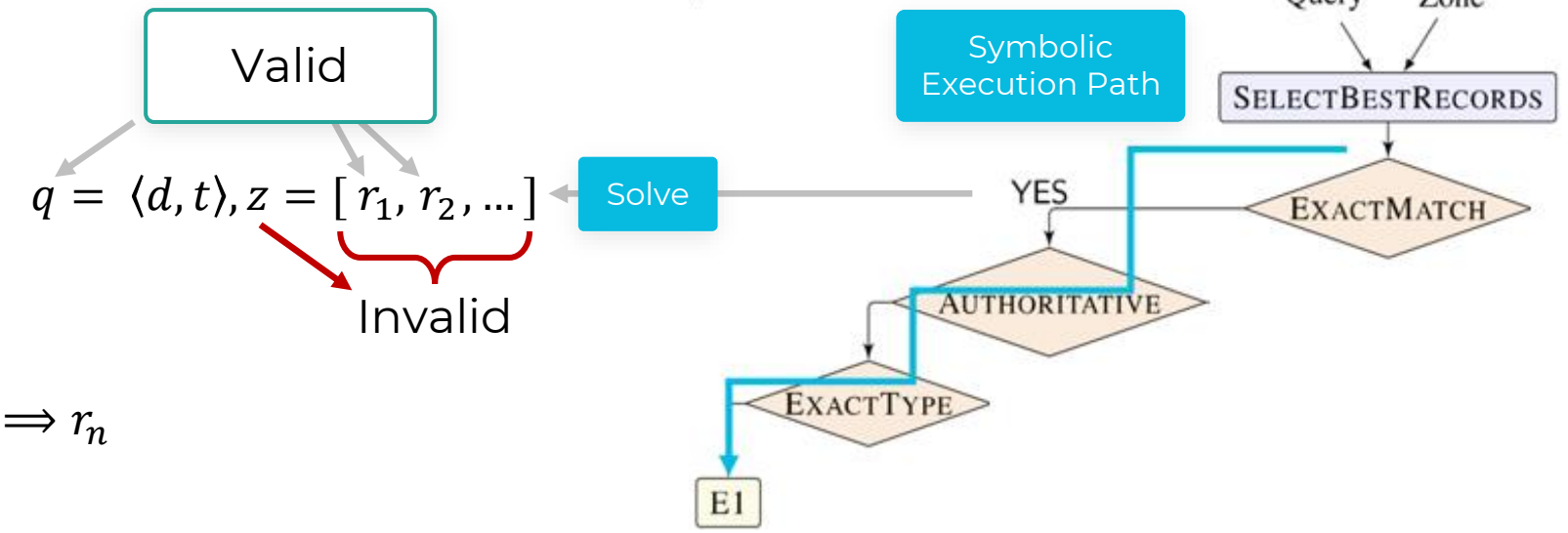


Challenge – Generating Valid Zones



- Zone must satisfy several conditions to be valid
- Example condition C_1 - There can be only one DNAME record for a domain name

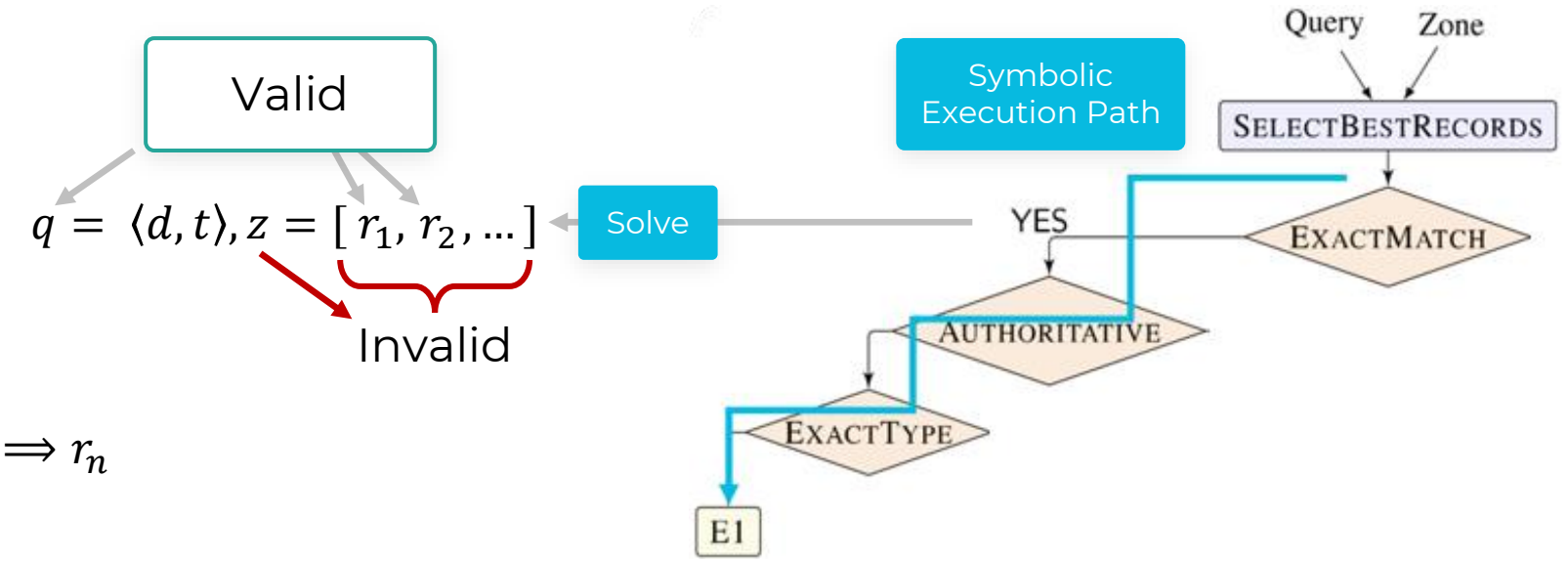
Challenge – Generating Valid Zones



r_1 = {**foo.com.**,DNAME,bar.edu.} $\Longrightarrow r_n$
$\neq$ {**foo.com.**,DNAME,web.in.}

- Zone must satisfy several conditions to be valid
- Example condition C_1 - There can be only one DNAME record for a domain name

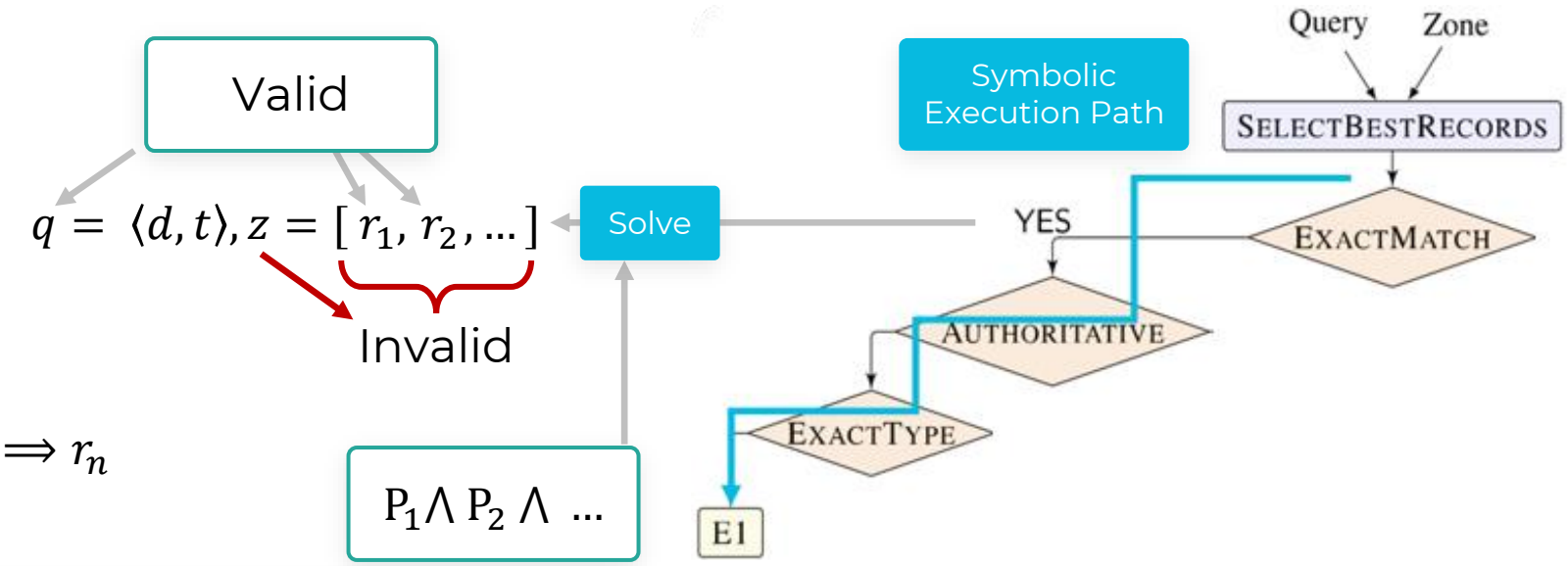
Challenge – Generating Valid Zones



r_1 = {**foo.com.**,DNAME,bar.edu.} $\Longrightarrow r_n$
$\neq$ {**foo.com.**,DNAME,web.in.}

- Zone must satisfy several conditions to be valid
- Example condition C_1 - There can be only one DNAME record for a domain name
- Conditions C_1, C_2, … → Zen predicates P_1, P_2, …

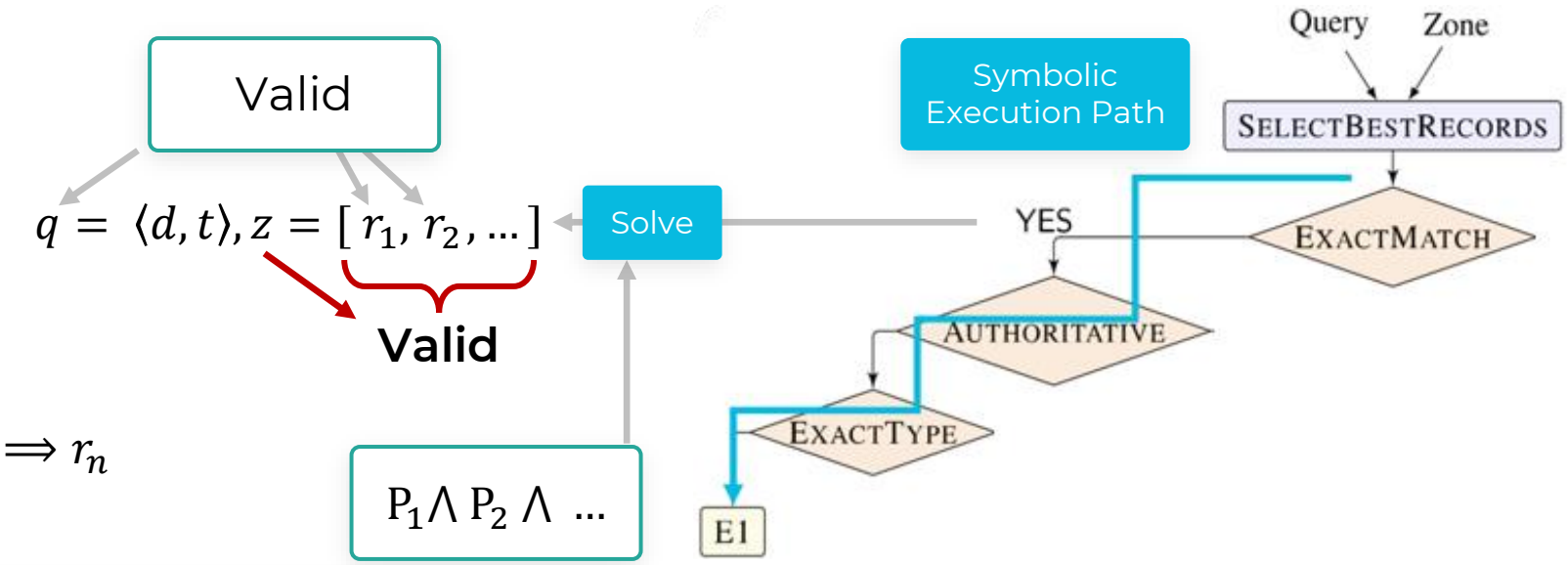
Challenge – Generating Valid Zones



r_1 = {**foo.com.**,DNAME,bar.edu.} $\Longrightarrow r_n$
$\neq$ {**foo.com.**,DNAME,web.in.}

- Zone must satisfy several conditions to be valid
- Example condition C_1 - There can be only one DNAME record for a domain name
- Conditions C_1, C_2, … → Zen predicates P_1, P_2, …

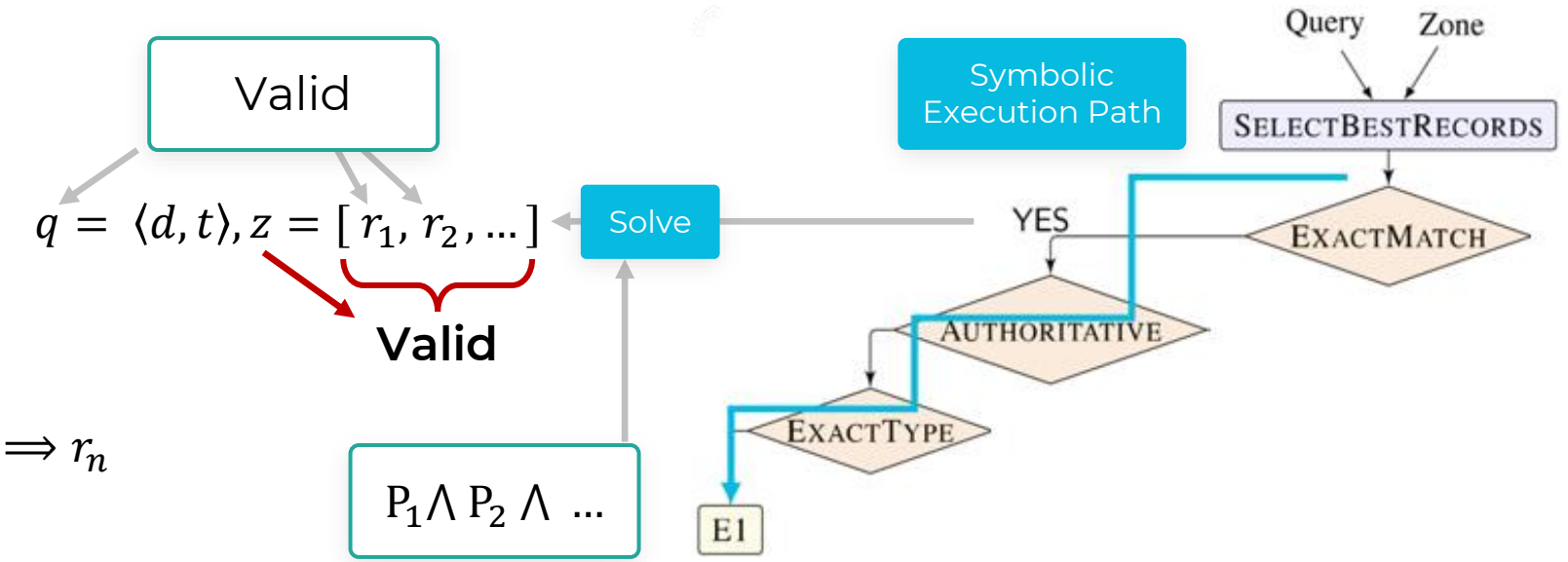
Challenge – Generating Valid Zones



r_1 = {foo.com.,DNAME,bar.edu.} $\Longrightarrow r_n$
$\neq$ {foo.com.,DNAME,web.in.}

- Zone must satisfy several conditions to be valid
- Example condition C_1 - There can be only one DNAME record for a domain name
- Conditions C_1, C_2, … → Zen predicates P_1, P_2, …

Challenge – Generating Valid Zones

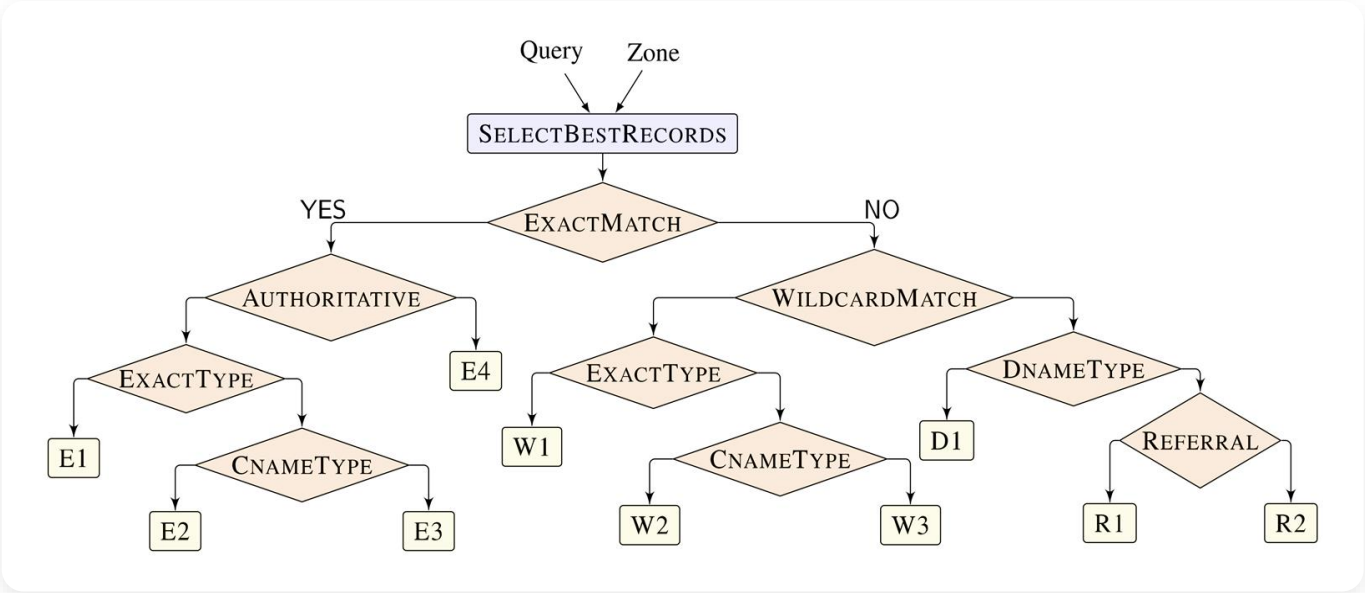


r_1 = {foo.com.,DNAME,bar.edu.} $\Longrightarrow r_n$
$\neq$ {foo.com.,DNAME,web.in.}

- Zone must satisfy several conditions to be valid
- Example condition C_1 - There can be only one DNAME record for a domain name
- Conditions C_1, C_2, … → Zen predicates P_1, P_2, …

We also generate invalid zone files using Zen predicates

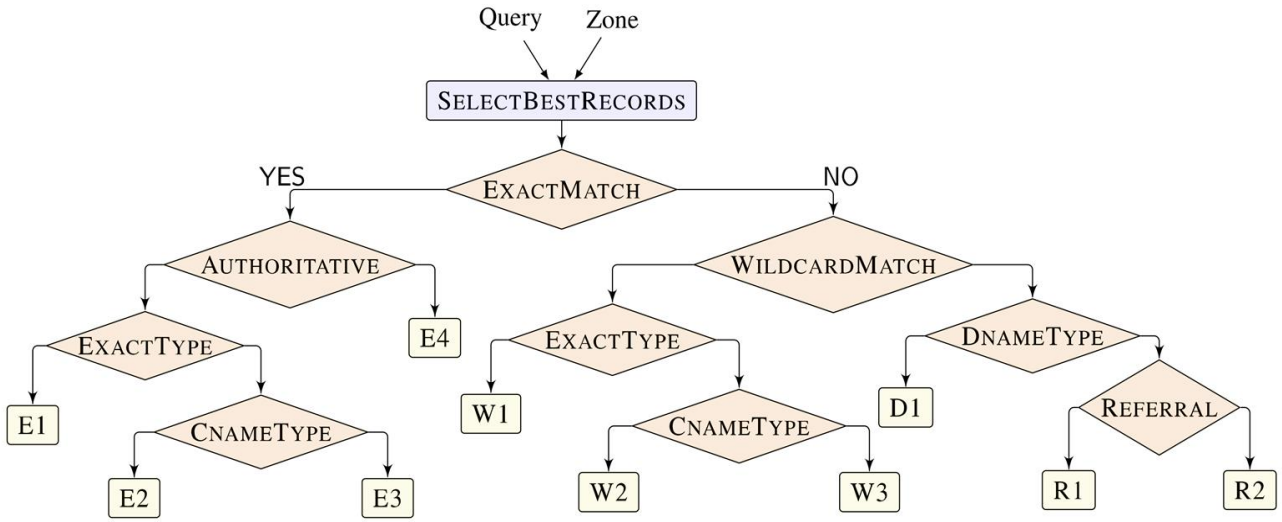
Exhaustive Model Coverage with Test Generation



Using small-scope property of DNS we limit the length of each domain name & the number of records in the zone ≤ 4

Exhaustive Model Coverage with Test Generation

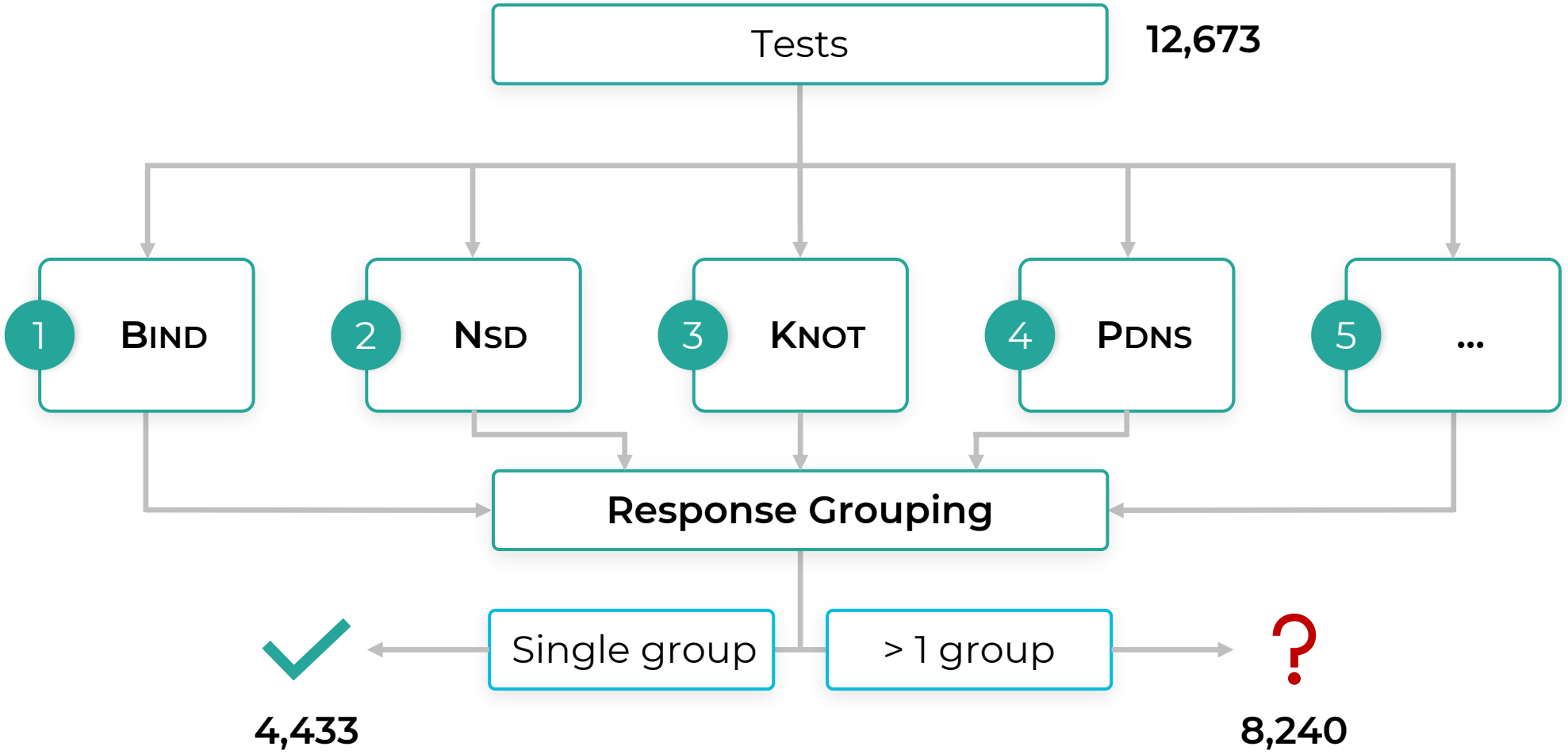
All model leaves are covered



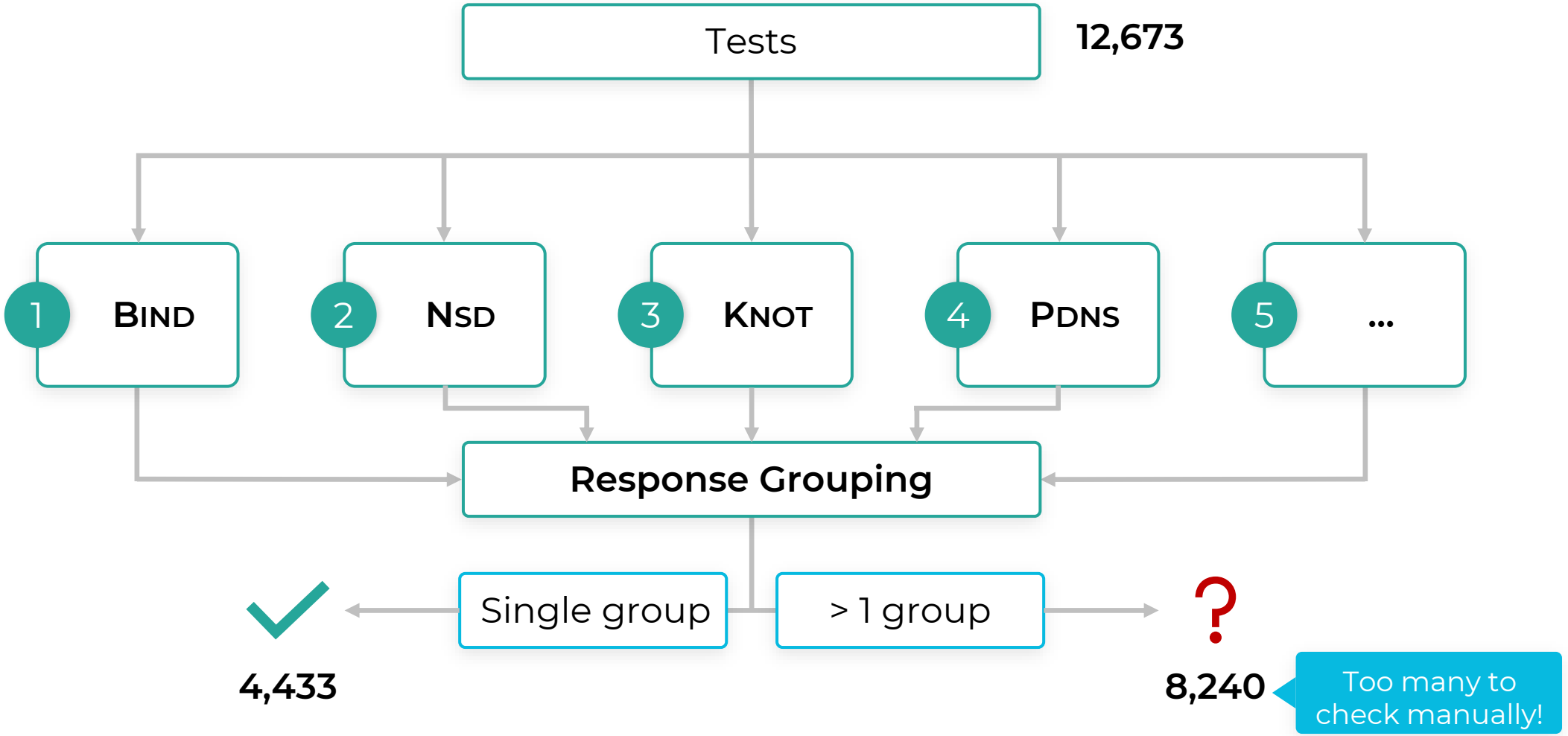
Using small-scope property of DNS we limit the length of each domain name & the number of records in the zone ≤ 4

Model Case	Number of Tests
E1	3180
E2	12
E4	96
W1	6036
W2	60
W3	24
D1	18
R1	230
R2	2980
Total	**12,673**

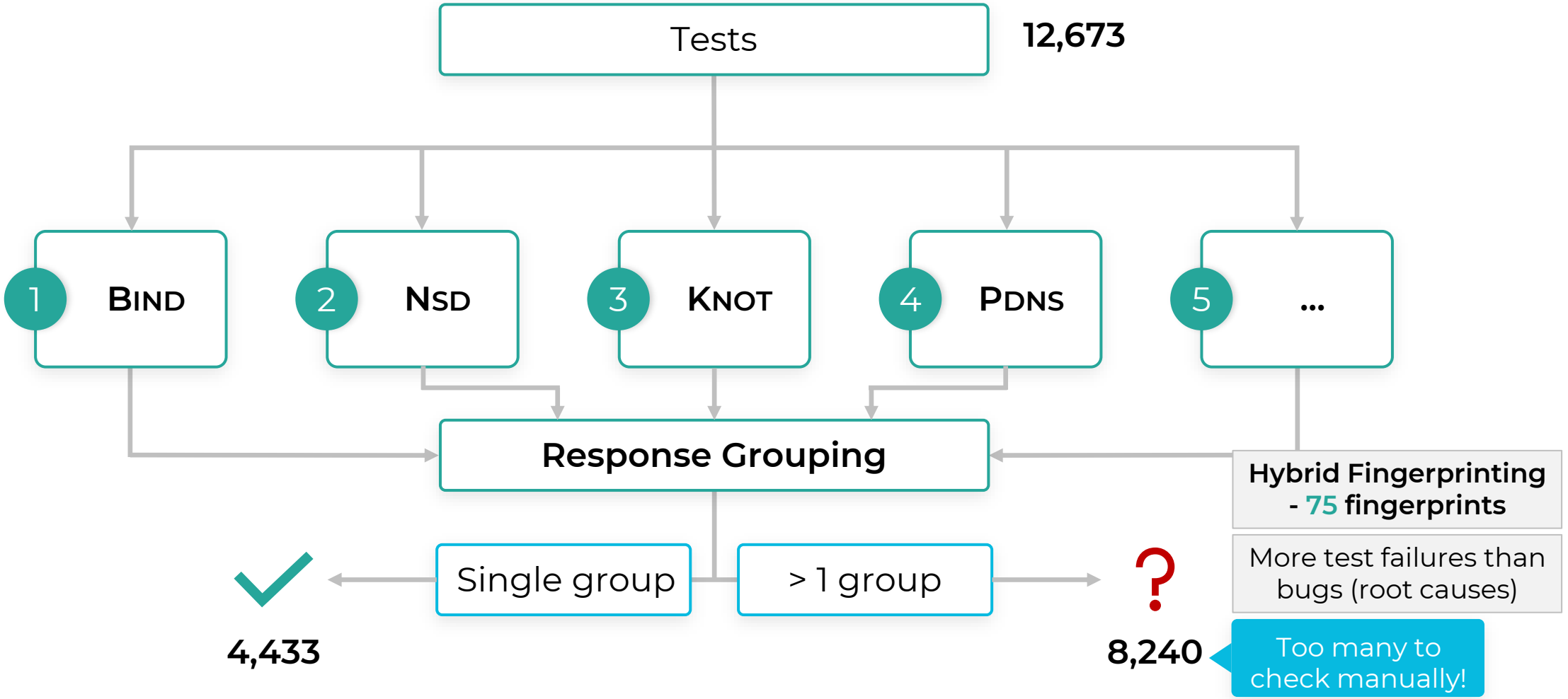
DNS Differential Testing



DNS Differential Testing



DNS Differential Testing



Hybrid Fingerprinting

Model Case	Number of Tests	Number of Tests Failing
E1	3180	239
E2	12	10
E3	96	12
E4	6036	5312
W1	60	33
W2	24	21
W3	18	16
D1	230	65
R1	2980	2529
R2	37	3

- **Fingerprint** failed tests
- Based on model case and the unique implementations in each group from the responses
- Example fingerprint – ⟨R1, {Nsd, Knot, PowerDns, Yadifa}, {Bind, Coredns}, {TrustDns, Maradns}⟩
- Unlikely for different unique bugs to have the same fingerprint

Hybrid Fingerprinting

Model Case	Number of Tests	Number of Tests Failing
E1	3180	239
W3	18	16
D1	230	65
R1	2980	2529
R2	37	3

Hybrid Fingerprint:

- Signature + Behavior
- Commonly used in IDS but not for triaging bugs

- **Fingerprint** failed tests
- the unique group from

Structural Signature (specification-based)

- **Example fingerprint** – ⟨R1, {Nsd, Knot, PowerDns, Yadifa}, {Bind, Coredns}, {TrustDns, Maradns}⟩
- Unlikely for have the sa

Behavioral Differences (differential testing)

Hybrid Fingerprinting

Model Case	Number of Tests	Number of Tests Failing	Number of Fingerprints
E1	3180	239	7
E2	12	10	5
E3	96	12	3
E4	6036	5312	11
W1	60	33	8
W2	24	21	9
W3	18	16	1
D1	230	65	4
R1	2980	2529	27
R2	37	3	1

- **Fingerprint** failed tests
- Based on model case and the unique implementations in each group from the responses
- Example fingerprint – ⟨R1, {Nsd, Knot, PowerDns, Yadifa}, {Bind, Coredns}, {TrustDns, Maradns}⟩
- Unlikely for different unique bugs to have the same fingerprint

Bugs Found and Confirmed in Open-source DNS Implementations

Implementation	Language	Description	Bugs found	Crashes
BIND	**C**	De facto standard	**4**	**1**
POWERDNS	**C++**	Popular in North Europe	**2**	**--**
NSD	**C**	Hosts several TLDs	**4**	**--**
KNOT	**C**	Hosts several TLDs	**5**	**--**
COREDNS	**Go**	Used in Kubernetes	**6**	**1**
YADIFA	**C**	Created by EURid (.eu)	**3**	**--**
TRUSTDNS	**Rust**	Security, safety focused	**4**	**1**
MARADNS	**C**	Lightweight server	**2**	**--**

Bugs Found and Confirmed in Open-source DNS Implementations

Implementation	Language	Description	Bugs found	Crashes
BIND	**C**	De facto standard	**4**	**1**
POWERDNS	**C++**	Popular in North Europe	**2**	**--**
NSD	**C**	Hosts several TLDs	**4**	**--**
KNOT	**C**	Hosts several TLDs	**5**	**--**
COREDNS	**Go**	Used in Kubernetes	**6**	**1**
YADIFA	**C**	Created by EURid (.eu)	**3**	**--**
TRUSTDNS	**Rust**	Security, safety focused	**4**	**1**
MARADNS	**C**	Lightweight server	**2**	**--**

Tests part of CI/CD pipeline in **Amazon Route 53** DNS

Example Bug – CoreDNS Crash

FERRET Generated Test Case

Domain Name	Type	Data
example.	SOA	ns1.exm. …
*.example.	CNAME	foo.example.

Query: ⟨"baz.bar.example., A"⟩

- Query is rewritten using CNAME to: "baz.bar.example. CNAME foo.example."
- The rewritten query will match the wildcard again !

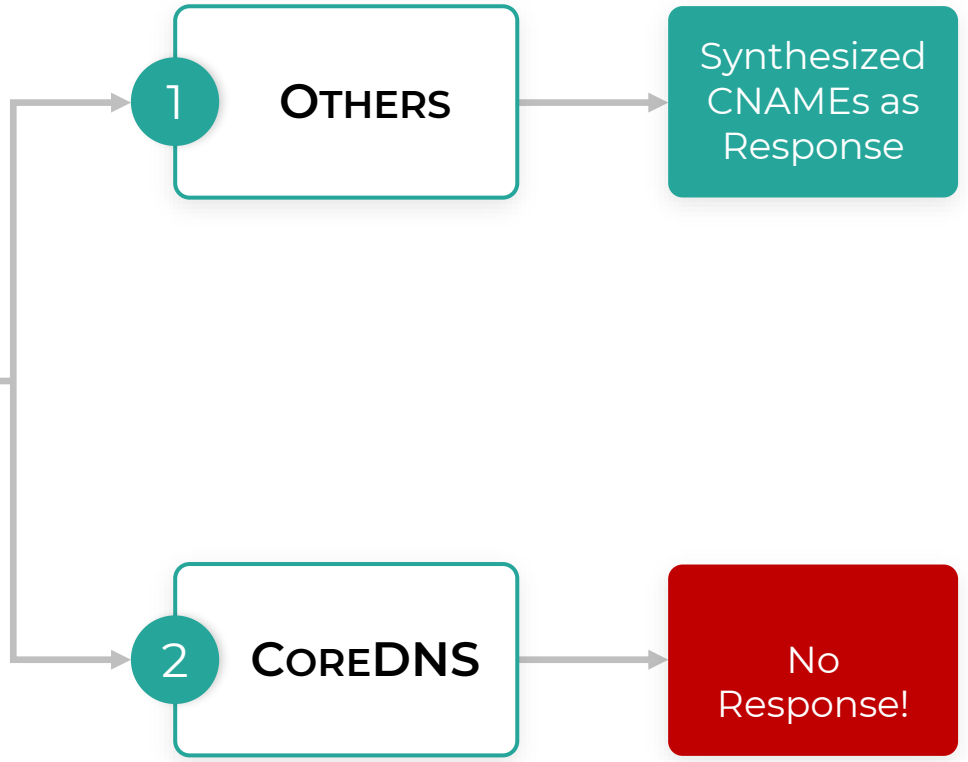
Example Bug – CoreDNS Crash

FERRET Generated Test Case

Domain Name	Type	Data
example.	SOA	ns1.exm. ...
*.example.	CNAME	foo.example.

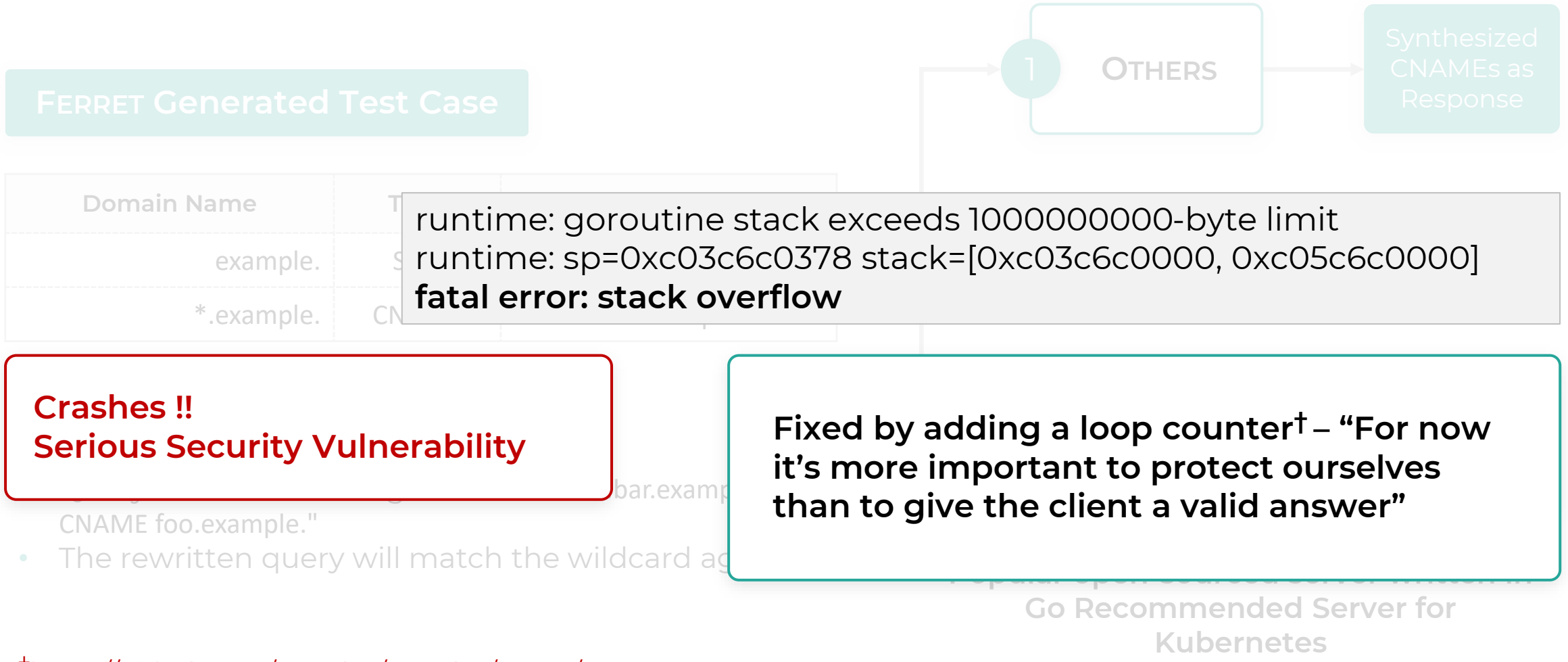
Query: ⟨"baz.bar.example., A"⟩

- Query is rewritten using CNAME to: "baz.bar.example. CNAME foo.example."
- The rewritten query will match the wildcard again !



Popular open-sourced server written in Go Recommended Server for Kubernetes

Example Bug – CoreDNS Crash



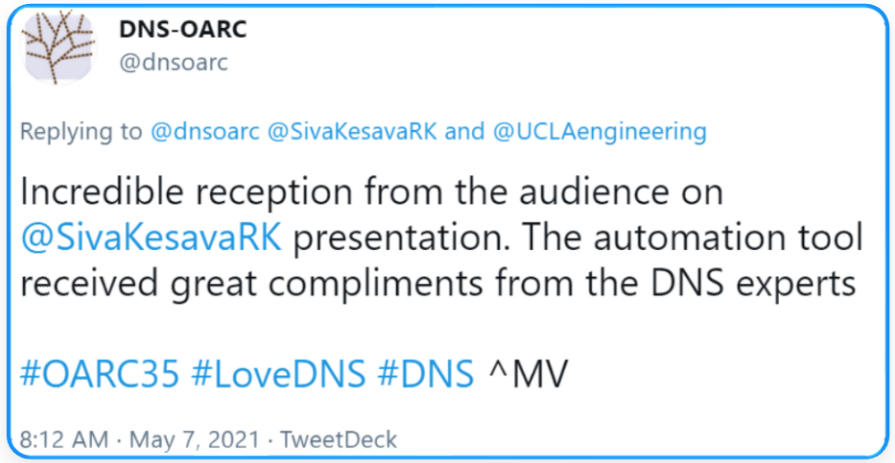
†https://github.com/coredns/coredns/issues/4378

Comments from DNS Community

"**This is awesome**, thank you for this work, and thank you for your very clear bug reports, both to us (PowerDNS) and to other projects."

"I was not kidding about the **excellent** bug reports, by the way.."

— Peter Van Dijik
(Senior PowerDNS Developer)



"I was skeptical because I thought – why should I believe his tests, but **he proved them** by running against so many DNS servers through them"

"So, possibly new RFCs should come with their **own logic diagram** which can be used to generate the tests"

— Vicky Risk
(Director of Marketing, ISC Bind)
And
Pauel Hauffman
(IETF & ICANN)

Summary

Technical Challenge

Must *jointly* generate structured zone files and queries in order to check RFC behavior compliance of DNS nameserver implementations

Key Idea

Leverages the small-scope property to build an executable model of DNS resolution and symbolically execute it to generate high-coverage tests that cover all paths in the model

Impact

Found dozens of bugs across 8 nameserver implementations, including 3 critical security vulnerabilities

FERRET: github.com/dns-groot/Ferret
Dataset: github.com/dns-groot/FerretDataset

Generating Invalid Zone Files

- Bugs also occur when handling of ill-formed zones
- Fuzzing (previous work) → syntactically invalid inputs
- Zen → semantically invalid zone files
- GROOT → Test queries for invalid zone files