dnssoftver: a tool to fingerprint DNS resolver software versions

Yevheniya Nosyk, Jan Bayer, Andrzej Duda, Maciej Korczyński

yevheniya.nosyk@univ-grenoble-alpes.fr
Université Grenoble Alpes (France)

OARC 42 (Charlotte, USA)
8 February 2024

version.bind

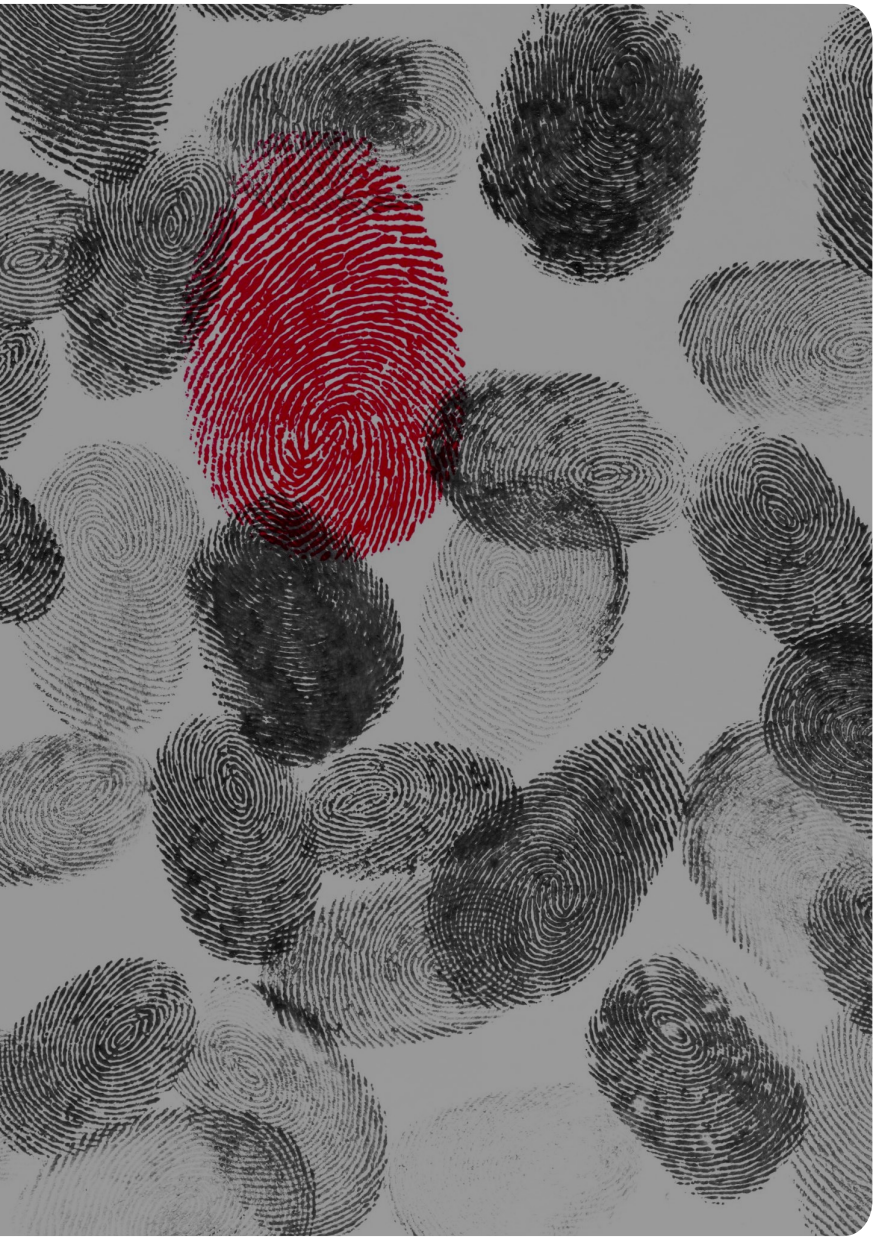
```
$ dig @1.2.3.4 CH TXT version.bind +short
unbound 1.16.2

$ dig @5.6.7.8 CH TXT version.bind +short
9.19.13

$ dig @9.10.11.12 CH TXT version.bind +short
PowerDNS Recursor 4.5.0-beta2 (built Dec 30 2023 11:22:09 by root@c828502c66b1)
```

version.bind in the wild

```
$ dig @1.2.3.4 CH TXT version.bind +short
…
My named is Bind, James BIND
…
None of your business
…
go to sleep
…
Its Secret, Don't try to fetch it
…
Hmmm... neh.
```

We need a tool that …

1. Covers various resolver software
2. Does not rely on specific configurations
3. Scales to support future software releases



Presenting today – dnssoftver

https://github.com/yevheniya-nosyk/dnssoftver

Overview

1. Choose and set up software
2. Design and run test cases
3. Generate the decision tree
4. Implement the scanner

Overview

1. Choose and set up software
2. Design and run test cases
3. Generate the decision tree
4. Implement the scanner

Vendors and versions: 674 in total

Vendor	Versions	Since	Count
BIND9	9.10.0 - 9.19.9	2014	293
Unbound	1.5.0 - 1.17.1rc2	2014	107
Knot Resolver	2.0.0 - 5.7.0	2018	35
PowerDNS Recursor	4.0.0 - 4.9.1	2015	122
Technitium DNS Server	1.0 - 11.4.1	2017	68
MaraDNS	2.0.11 - 3.5.0036	2015	46
Windows Server	2016 - 2022	2016	3

Vendors and versions: setup

641 custom Dockerfiles

30 Knot Resolver images pulled from cznic/knot-resolver

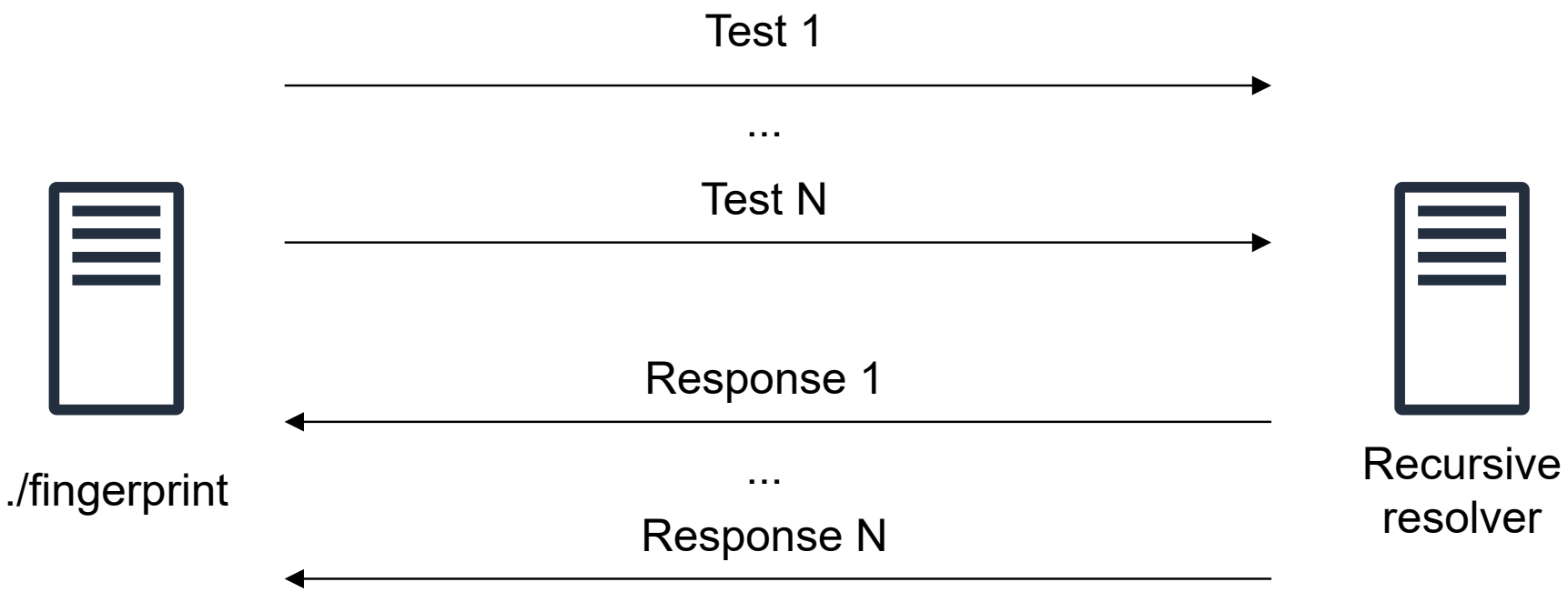
3 virtual private servers with Windows Server

Sources: https://www.docker.com/company/newsroom/media-resources/, https://nitin27may.medium.com/pushing-custom-docker-image-to-docker-hub-631dce3492f5, https://corporate.ovhcloud.com/en/newsroom/assets/

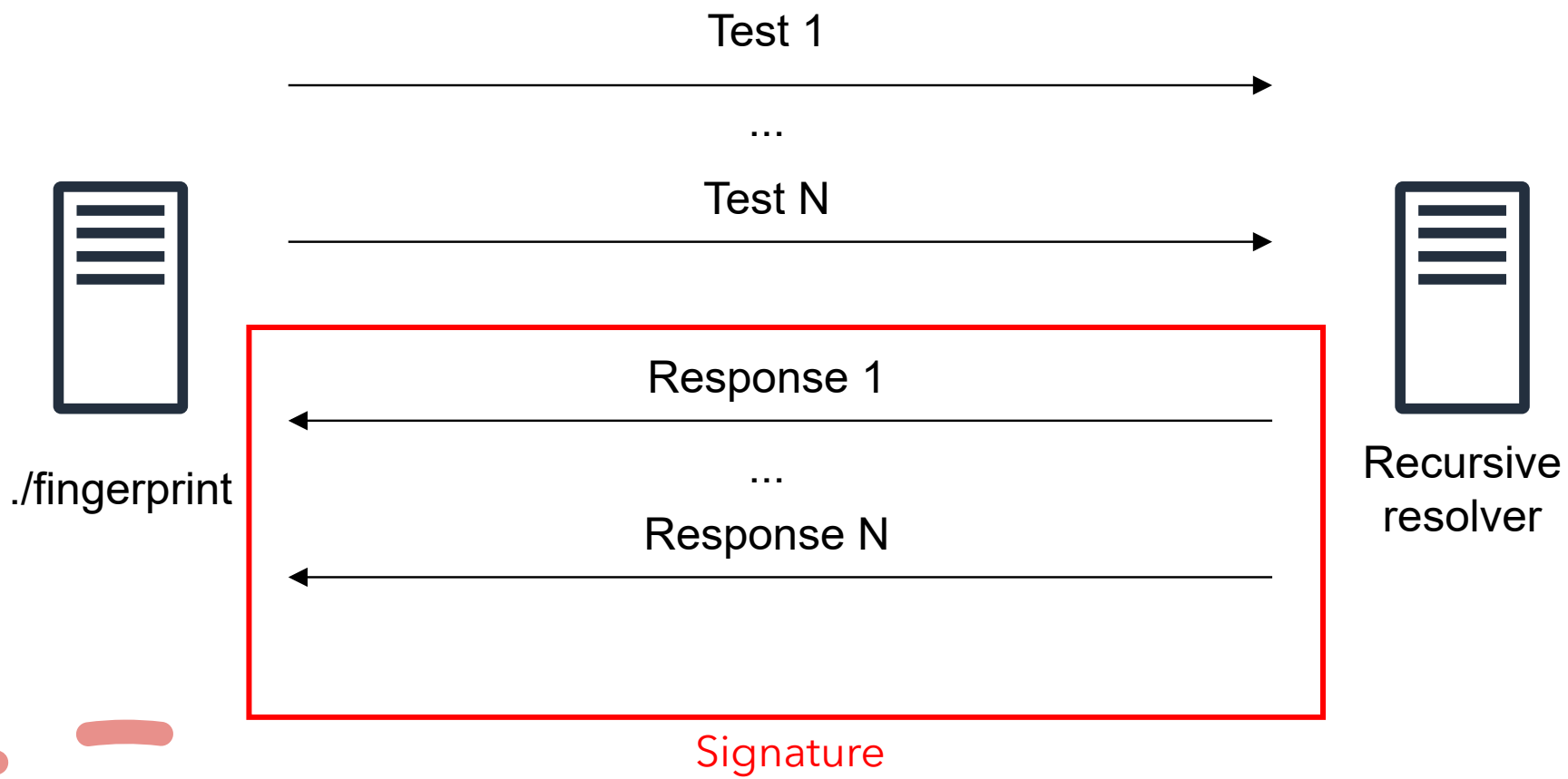
Overview

1. Choose and set up software
2. Design and run test cases
3. Generate the decision tree
4. Implement the scanner

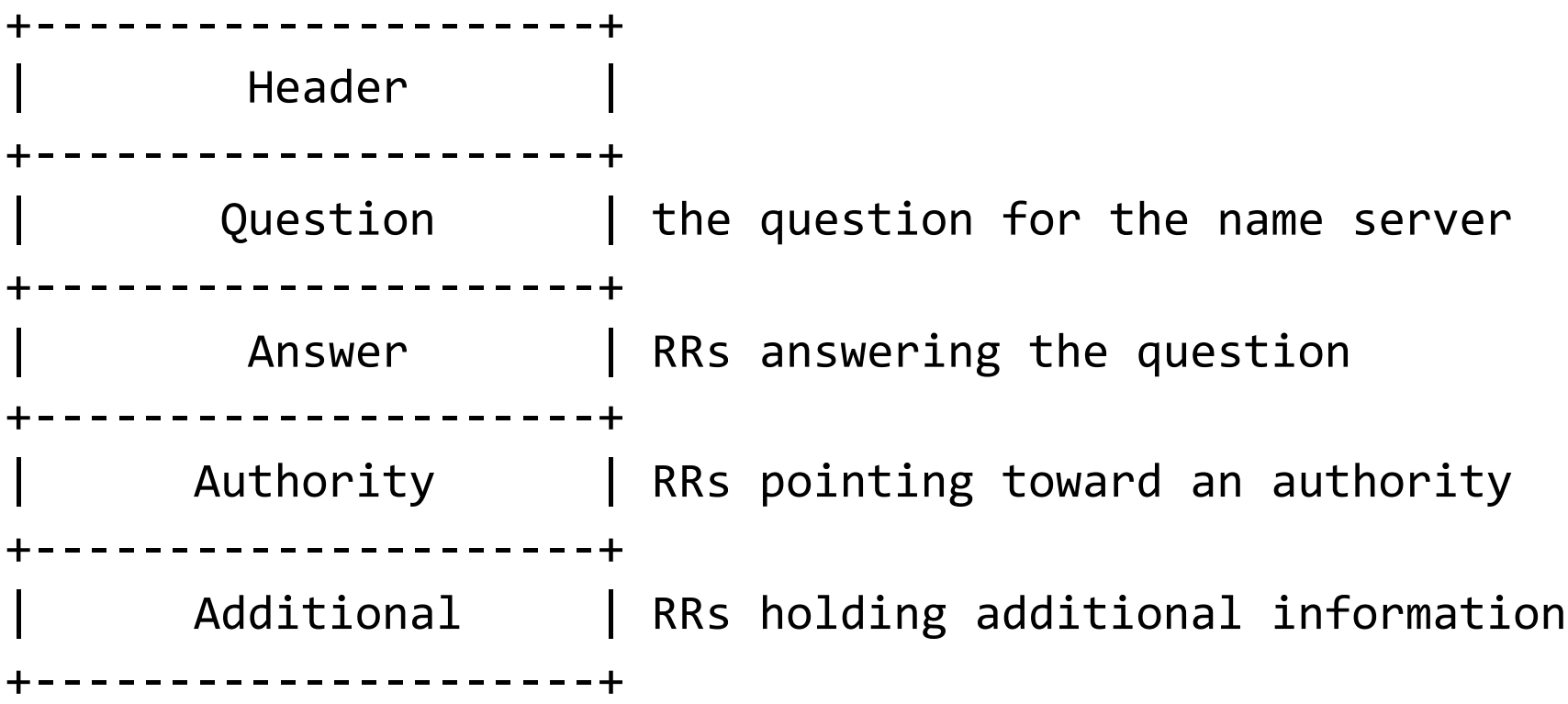
Test case == DNS request



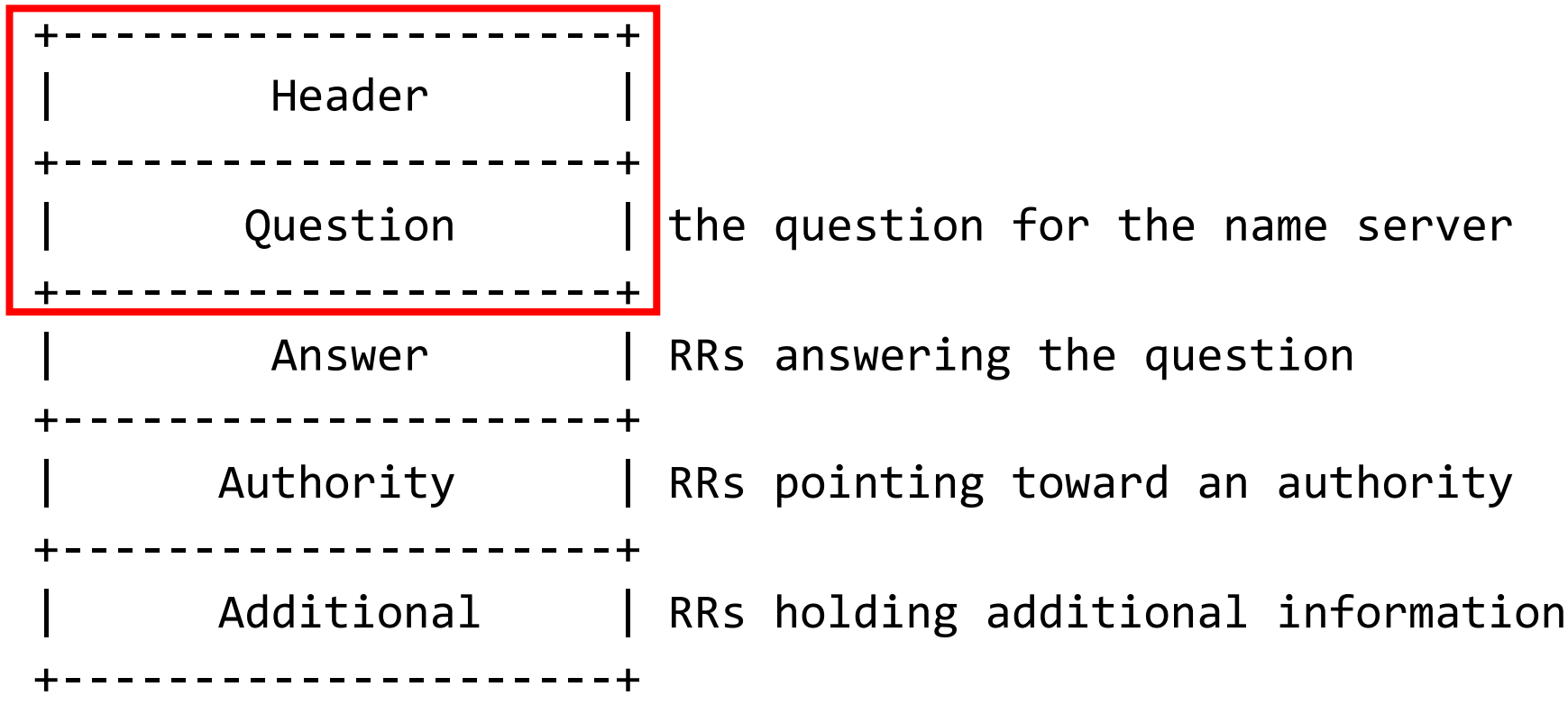
Signature == DNS responses



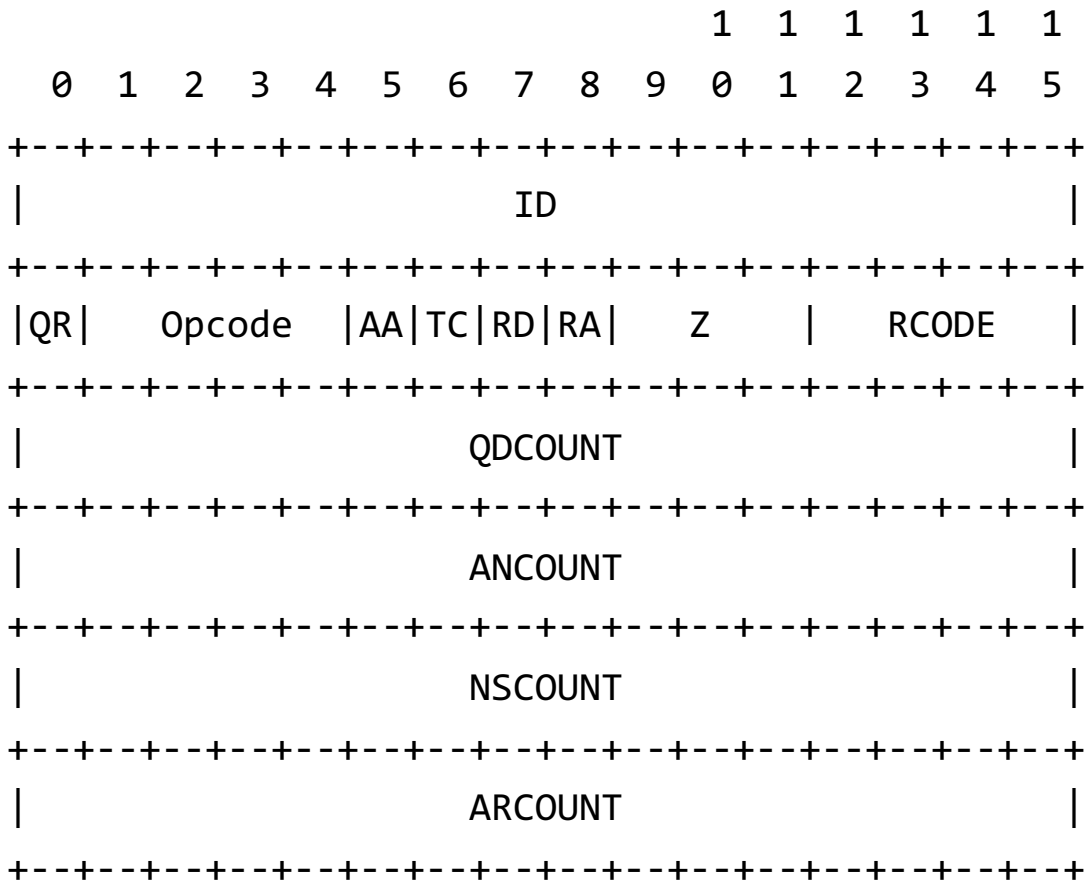
RFC-1035: DNS message



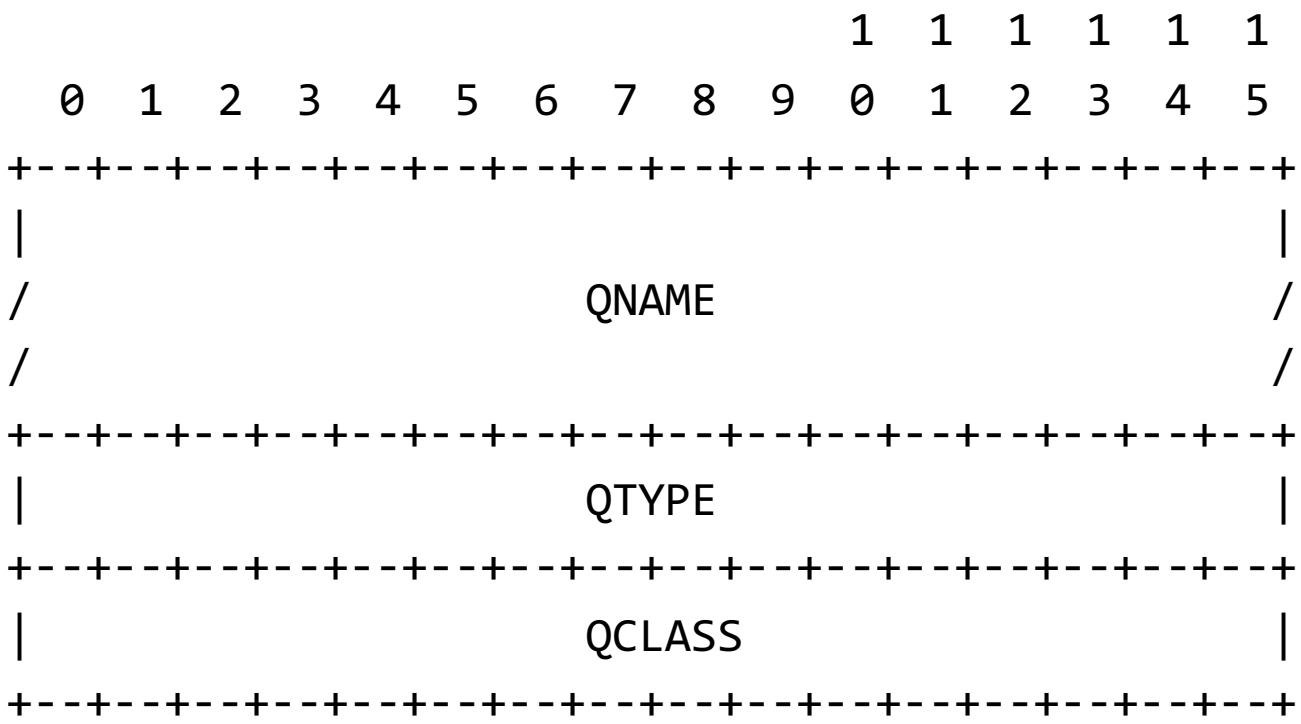
RFC-1035: DNS message



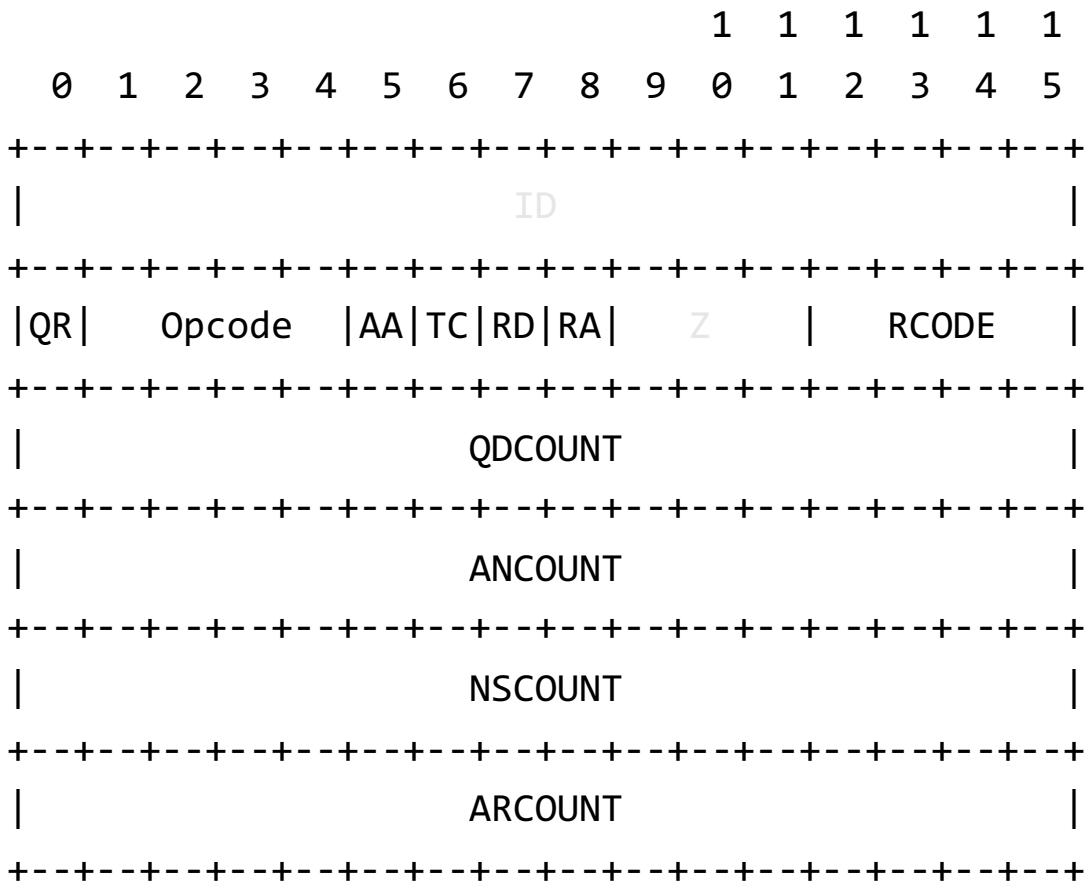
RFC-1035: DNS header



RFC-1035: DNS question

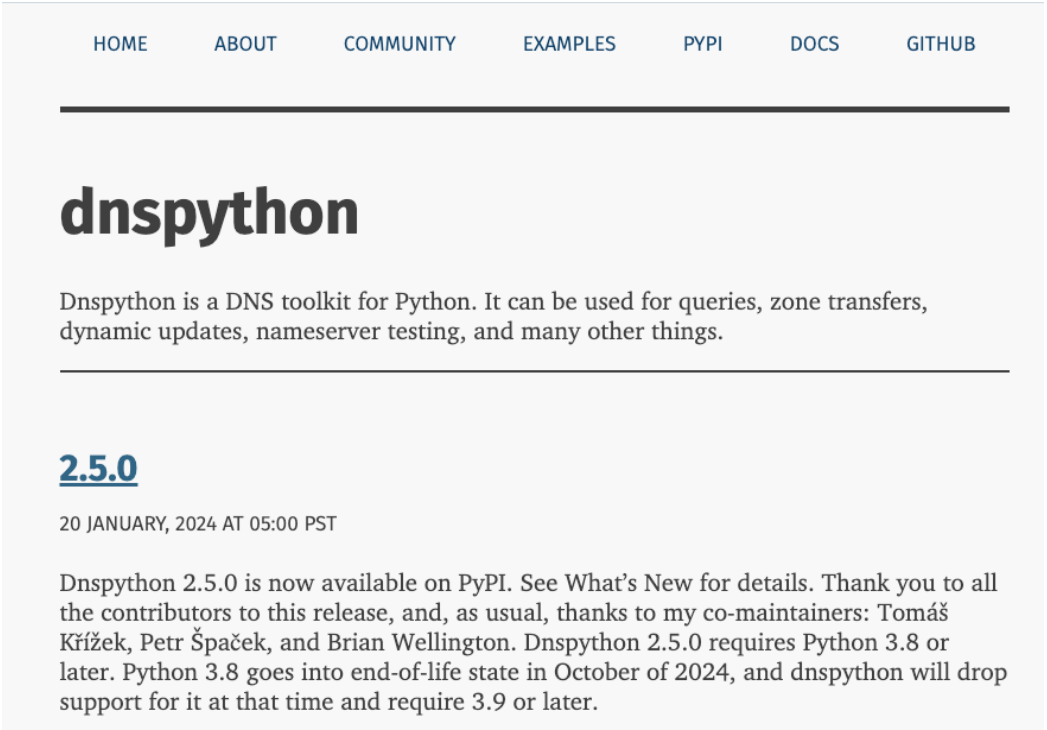


Testcase response



768 test cases

```
QR: [0,1]
AA: [0,1]
TC: [0,1]
RD: [0,1]
RA: [0,1]
Opcode: [QUERY, IQUERY, STATUS, NOTIFY]
Class: [RESERVED0, IN, CH, HS, NONE, ANY]
Resource record: [A]
Domain name: [baseline.dnssoftver.com]
```



Source: https://www.dnspython.org

Domain name configuration

```
;NS records
@               IN      NS      ns1
ns1             IN      A       65.21.51.117
ns1             IN      AAAA    2a01:4f9:c012:7407::1
@               IN      NS      ns2
ns2             IN      A       65.21.51.117
ns2             IN      AAAA    2a01:4f9:c012:7407::1

;Baseline test domain
@                IN      A       129.88.46.51
@                IN      A       129.88.68.231
@                IN      A       129.88.68.222
@                IN      AAAA    2001:660:5301:46::46:51
*                IN      A       129.88.46.51
*                IN      A       129.88.68.231
*                IN      A       129.88.68.222
*                IN      AAAA    2001:660:5301:46::46:51
```

Workflow

```
for _ in range(30):
        for software in software_all:
                start_docker_container()
                issue_768_test_cases()
                kill_docker_container()
                write_intermediary_result()
```

Multiple rounds

1. For the training data
2. For inconsistencies:
    - Network failures (Timeouts)
    - Different responses (NOERROR vs. SERVFAIL, etc.)

Overview

1. Choose and set up software
2. Design and run test cases
3. Generate the decision tree
4. Implement the scanner

Input: 15 million DNS requests/responses


```
"Opcode": "QUERY", "AA": 0, "TC": 0, "RD": 1, "RA": 0, "RCODE": "REFUSED", "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0},
"baseline_A_NONE_IQUERY_AA_RD": {"QR": 1, "Opcode": "IQUERY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0,
"ARCOUNT": 0}, "baseline_A_NONE_IQUERY_RD": {"QR": 1, "Opcode": "IQUERY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0,
"NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_ANY_QUERY_AA_RD_RA": {"QR": 1, "Opcode": "QUERY", "AA": 0, "TC": 0, "RD": 1, "RA": 1, "RCODE": "NOERROR", "QDCOUNT": 1,
"ANCOUNT": 3, "NSCOUNT": 2, "ARCOUNT": 0}, "baseline_A_NONE_NOTIFY_QR_AA_TC_RD": {"error": "Timeout after 5 seconds"}, "baseline_A_HS_QUERY_AA_TC_RD_RA": {"QR":
1, "Opcode": "QUERY", "AA": 0, "TC": 0, "RD": 1, "RA": 0, "RCODE": "REFUSED", "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0},
"baseline_A_NONE_STATUS_QR_RD_RA": {"error": "Timeout after 5 seconds"}, "baseline_A_CH_IQUERY_AA_RA": {"QR": 1, "Opcode": "IQUERY", "AA": 0, "TC": 0, "RD": 0,
"RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_NONE_QUERY_AA_RA": {"QR": 1, "Opcode": "QUERY", "AA": 0, "TC":
0, "RD": 0, "RA": 0, "RCODE": "REFUSED", "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_NONE_STATUS_TC_RA": {"QR": 1, "Opcode": "STATUS",
"AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_CH_NOTIFY_QR_TC": {"error": "Timeout
after 5 seconds"}, "baseline_A_ANY_NOTIFY_AA_TC_RD": {"QR": 1, "Opcode": "NOTIFY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "FORMERR", "QDCOUNT": 1,
"ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_ANY_QUERY_QR_TC_RA": {"error": "Timeout after 5 seconds"}, "baseline_A_HS_IQUERY_AA": {"QR": 1, "Opcode":
"IQUERY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_HS_NOTIFY_AA": {"QR": 1,
"Opcode": "NOTIFY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "REFUSED", "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0},
"baseline_A_NONE_NOTIFY_AA_TC": {"QR": 1, "Opcode": "NOTIFY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "REFUSED", "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0,
"ARCOUNT": 0}, "baseline_A_ANY_STATUS_QR_AA_TC": {"error": "Timeout after 5 seconds"}, "baseline_A_IN_STATUS_AA_RA": {"QR": 1, "Opcode": "STATUS", "AA": 0,
"TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_RESERVED0_IQUERY_AA_TC_RD": {"QR": 1,
"Opcode": "IQUERY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0},
"baseline_A_ANY_NOTIFY_AA_RA": {"QR": 1, "Opcode": "NOTIFY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "FORMERR", "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0,
"ARCOUNT": 0}, "baseline_A_IN_NOTIFY_QR_TC": {"error": "Timeout after 5 seconds"}, "baseline_A_CH_STATUS_AA_TC_RD_RA": {"QR": 1, "Opcode": "STATUS", "AA": 0,
"TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_RESERVED0_QUERY_AA_TC_RD_RA": {"QR": 1,
"Opcode": "QUERY", "AA": 0, "TC": 0, "RD": 1, "RA": 0, "RCODE": "FORMERR", "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0},
"baseline_A_ANY_QUERY_QR_AA_TC_RD": {"error": "Timeout after 5 seconds"}, "baseline_A_CH_IQUERY_QR_RA": {"error": "Timeout after 5 seconds"},
"baseline_A_IN_IQUERY_TC": {"QR": 1, "Opcode": "IQUERY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0,
"ARCOUNT": 0}, "baseline_A_CH_STATUS": {"QR": 1, "Opcode": "STATUS", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0,
"NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_HS_IQUERY_TC_RD": {"QR": 1, "Opcode": "IQUERY", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0,
"ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_RESERVED0_STATUS_AA": {"QR": 1, "Opcode": "STATUS", "AA": 0, "TC": 0, "RD": 0, "RA": 0, "RCODE":
"NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_CH_IQUERY_QR_AA": {"error": "Timeout after 5 seconds"},
"baseline_A_CH_STATUS_QR_AA_RA": {"error": "Timeout after 5 seconds"}, "baseline_A_ANY_STATUS_AA_TC_RD_RA": {"QR": 1, "Opcode": "STATUS", "AA": 0, "TC": 0,
"RD": 0, "RA": 0, "RCODE": "NOTIMP", "QDCOUNT": 0, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0}, "baseline_A_HS_STATUS_QR": {"error": "Timeout after 5 seconds"},
```

Output: scikit-learn Decision Tree

scikit learn Install User Guide API Examples Community More Go

Prev Up Next

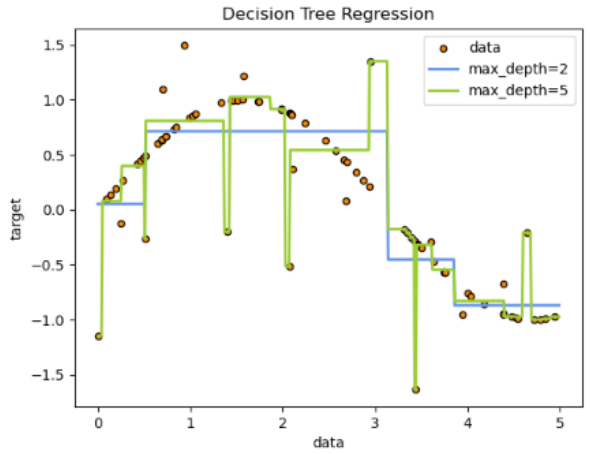
scikit-learn 1.3.2
Other versions

Please cite us if you use the software.

1.10. Decision Trees

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.



Fingerprinting granularity

bind	9.16.11
unbound	1.17.0
unbound	1.17.1
knot-resolver	2.1.1
knot-resolver	4.2.2
knot-resolver	5.5.3

Fingerprinting granularity: vendor

bind	9.16.11
unbound	1.17.0
unbound	1.17.1
knot-resolver	2.1.1
knot-resolver	4.2.2
knot-resolver	5.5.3

Decision tree: vendor

```
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} <= 0.50
|   |--- Q: {class: ANY, opcode: QUERY, flags: [AA,TC,RA]} and A: {RCODE: NOERROR, QDCOUNT: 1, flags: [QR,RA]} <= 0.50
|   |   |--- Q: {class: ANY, opcode: STATUS, flags : [AA,TC,RA]} and A: {RCODE: FORMERR, QDCOUNT: 1, flags: [AA,QR,RA]} <= 0.50
|   |   |   |--- Q: {class: IN, opcode: QUERY, flags: [TC,RA]} and A: {Timeout} <= 0.50
|   |   |   |   |--- Q: {class: IN, opcode: NOTIFY, flags: [AA,RD,RA]} and A: {RCODE: NOERROR, QDCOUNT: 1, ANCOUNT: 3, flags: [QR,RA,RD]} <= 0.50
|   |   |   |   |   |--- Q: {class: RESERVED0, opcode: STATUS, flags: [AA,RA]} and A: {Timeout} <= 0.50
|   |   |   |   |   |   |--- class: technitium
|   |   |   |   |   |--- Q: {class: RESERVED0, opcode: STATUS, flags: [AA,RA]} and A: {Timeout} >  0.50
|   |   |   |   |   |   |--- class: windows
|   |   |   |   |--- Q: {class: IN, opcode: NOTIFY, flags: [AA,RD,RA]} and A: {RCODE: NOERROR, QDCOUNT: 1, ANCOUNT: 3, flags: [QR,RA,RD]} >  0.50
|   |   |   |   |   |--- class: knot
|   |   |   |--- Q: {class: IN, opcode: QUERY, flags: [TC,RA]} and A: {Timeout} >  0.50
|   |   |   |   |--- class: maradns
|   |   |--- Q: {class: ANY, opcode: STATUS, flags: [AA,TC,RA]} and A: {RCODE: FORMERR, QDCOUNT: 1, flags: [AA,QR,RA]} >  0.50
|   |   |   |--- class: unbound
|   |--- Q: {class: ANY, opcode: QUERY, flags: [AA,TC,RA]} and A: {RCODE: NOERROR, QDCOUNT: 1, flags: [QR,RA]} >  0.50
|   |   |--- class: pdns
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} >  0.50
|   |--- class: bind9
```

Decision tree: vendor

```
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} <= 0.50
|    |--- Q: {class: ANY, opcode: QUERY, flags: [AA,TC,RA]} and A: {RCODE: NOERROR, QDCOUNT: 1, flags: [QR,RA]} <= 0.50
|    |    |--- Q: {class: ANY, opcode: STATUS, flags : [AA,TC,RA]} and A: {RCODE: FORMERR, QDCOUNT: 1, flags: [AA,QR,RA]} <= 0.50
|    |    |    |--- Q: {class: IN, opcode: QUERY, flags: [TC,RA]} and A: {Timeout} <= 0.50
|    |    |    |    |--- Q: {class: IN, opcode: NOTIFY, flags: [AA,RD,RA]} and A: {RCODE: NOERROR, QDCOUNT: 1, ANCOUNT: 3, flags: [QR,RA,RD]} <= 0.50
|    |    |    |    |    |--- Q: {class: RESERVED0, opcode: STATUS, flags: [AA,RA]} and A: {Timeout} <= 0.50
|    |    |    |    |    |    |--- class: technitium
|    |    |    |    |    |--- Q: {class: RESERVED0, opcode: STATUS, flags: [AA,RA]} and A: {Timeout} >  0.50
|    |    |    |    |    |    |--- class: windows
|    |    |    |    |--- Q: {class: IN, opcode: NOTIFY, flags: [AA,RD,RA]} and A: {RCODE: NOERROR, QDCOUNT: 1, ANCOUNT: 3, flags: [QR,RA,RD]} >  0.50
|    |    |    |    |    |--- class: knot
|    |    |    |--- Q: {class: IN, opcode: QUERY, flags: [TC,RA]} and A: {Timeout} >  0.50
|    |    |    |    |--- class: maradns
|    |    |--- Q: {class: ANY, opcode: STATUS, flags: [AA,TC,RA]} and A: {RCODE: FORMERR, QDCOUNT: 1, flags: [AA,QR,RA]} >  0.50
|    |    |    |--- class: unbound
|    |--- Q: {class: ANY, opcode: QUERY, flags: [AA,TC,RA]} and A: {RCODE: NOERROR, QDCOUNT: 1, flags: [QR,RA]} >  0.50
|    |    |--- class: pdns
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} >  0.50
|    |--- class: bind9
```

Decision tree: vendor

```
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} <= 0.50
|    ...
|    (issue more queries)
|    ...
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} >  0.50
|   |--- class: bind9
```

Decision tree: vendor

```
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} <= 0.50
|    ...
|    (issue more queries)
|    ...
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} >  0.50
|   |--- class: bind9
```

Decision tree: vendor

```
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} <= 0.50
|    ...
|    (issue more queries)
|    ...
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} >  0.50
|   |--- class: bind9
```

Decision tree: vendor

```
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} <= 0.50
|    ...
|    (issue more queries)
|    ...
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} >  0.50
|   |--- class: bind9
```

Decision tree: vendor

```
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} <= 0.50
|    ...
|    (issue more queries)
|    ...
|--- Q: {class: RESERVED0, opcode: STATUS, flags: [TC]} and A: {RCODE: NOTIMP, flags: [QR]} >  0.50
|   |--- class: bind9
```

Decision tree: vendor

- 768 testcases issued in total
- 6 testcases are significant
- 7 unique labels out of 7

Fingerprinting granularity: major

bind	9.16.11
unbound	1.17.0
unbound	1.17.1
knot-resolver	2.1.1
knot-resolver	4.2.2
knot-resolver	5.5.3

Decision tree: major

- 768 testcases issued in total
- 48 testcases are significant
- 19 unique labels out of 23
- Full tree on GitHub

Fingerprinting granularity: minor

bind	9.16.11
unbound	1.17.0
unbound	1.17.1
knot-resolver	2.1.1
knot-resolver	4.2.2
knot-resolver	5.5.3

Decision tree: minor

- 768 testcases issued in total
- 117 testcases are significant
- 51 unique labels out of 96
- Full tree on GitHub

Fingerprinting granularity: build

bind	9.16.11
unbound	1.17.0
unbound	1.17.1
knot-resolver	2.1.1
knot-resolver	4.2.2
knot-resolver	5.5.3

Decision tree: build

- 768 testcases issued in total
- 149 testcases are significant
- 382 unique labels out of 674
- Full tree on GitHub

Overview

1. Choose and set up software
2. Design and run test cases
3. Generate the decision tree
4. Implement the scanner

The scanner

```
$ python3 scan.py --input_file <input_file> -output_file <output_file> --
granularity [vendor,major,minor,build] --threads <num_of_threads>
```

The scanner

```
$ python3 scan.py --input_file <input_file> -output_file <output_file> --
granularity [vendor,major,minor,build] --threads <num_of_threads>
```

```
{"ip": "1.2.3.4","versions": ["unbound"]}
```

The scanner

```
$ python3 scan.py --input_file <input_file> -output_file <output_file> --
granularity [vendor,major,minor,build] --threads <num_of_threads>
```

```
{"ip": "1.2.3.4","versions": ["unbound"]}
{"ip": "1.2.3.4","versions": ["technitium-2"]}
```

The scanner

```
$ python3 scan.py --input_file <input_file> -output_file <output_file> --
granularity [vendor,major,minor,build] --threads <num_of_threads>
```

```
{"ip": "1.2.3.4","versions": ["unbound"]}
{"ip": "1.2.3.4","versions": ["technitium-2"]}
{"ip": "1.2.3.4","versions": ["bind-9.16"]}
```

The scanner

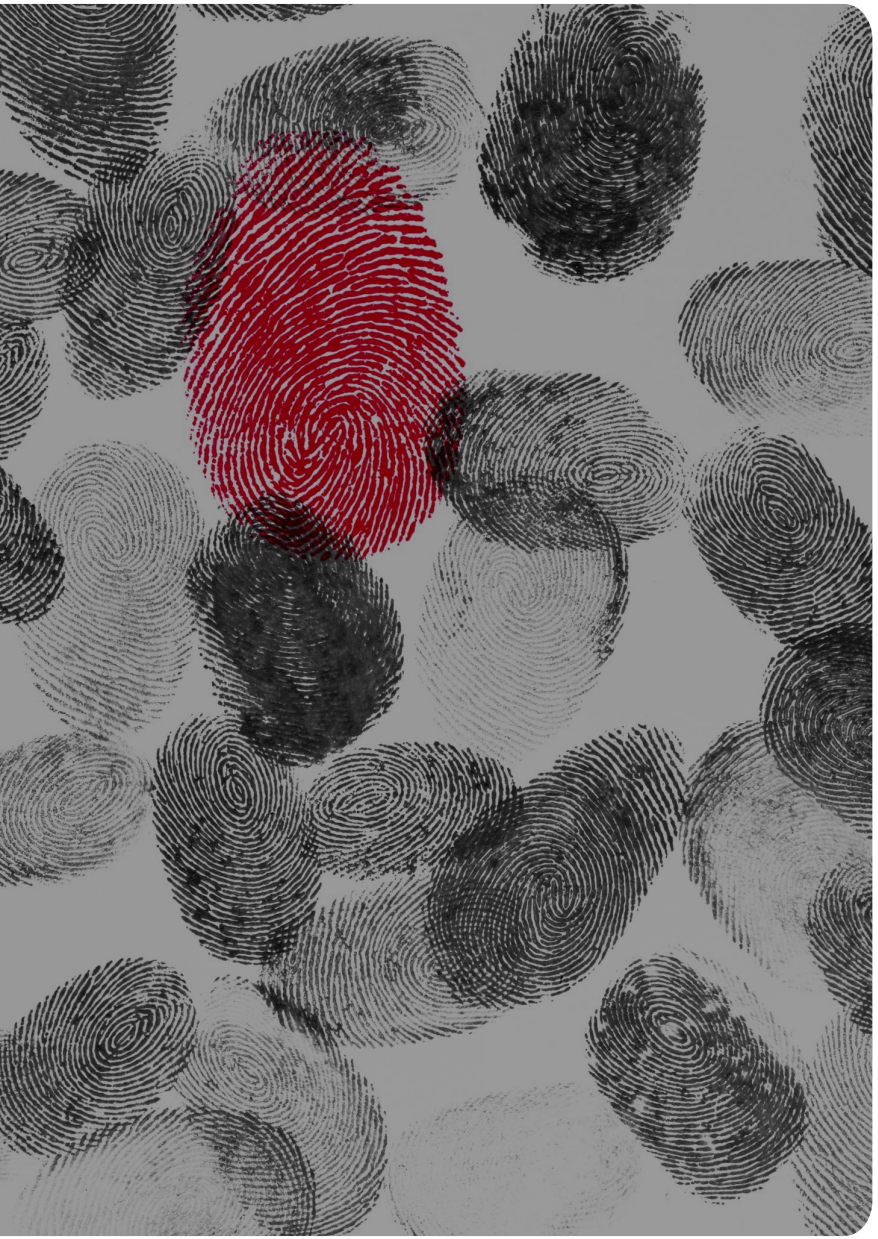
```
$ python3 scan.py --input_file <input_file> -output_file <output_file> --
granularity [vendor,major,minor,build] --threads <num_of_threads>
```

```
{"ip": "1.2.3.4","versions": ["unbound"]}
{"ip": "1.2.3.4","versions": ["technitium-2"]}
{"ip": "1.2.3.4","versions": ["bind-9.16"]}
{"ip": "1.2.3.4","versions": ["pdns-recuror-4.1.6"]}
```

The scanner

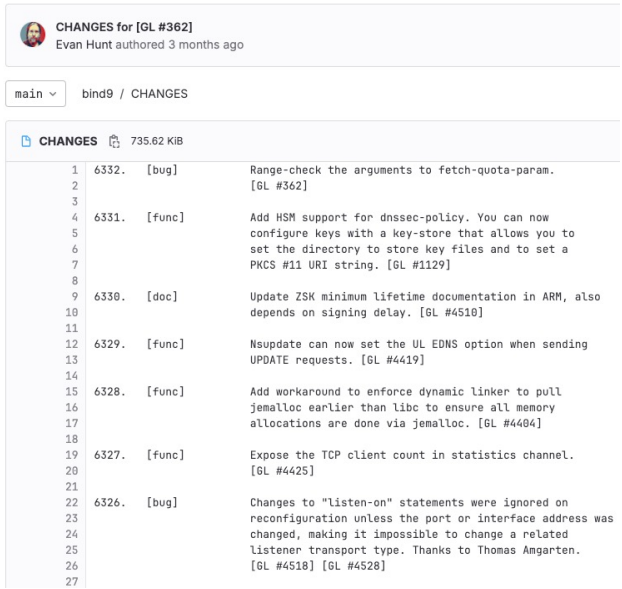
```
$ python3 scan.py --input_file <input_file> -output_file <output_file> --
granularity [vendor,major,minor,build] --threads <num_of_threads>
```

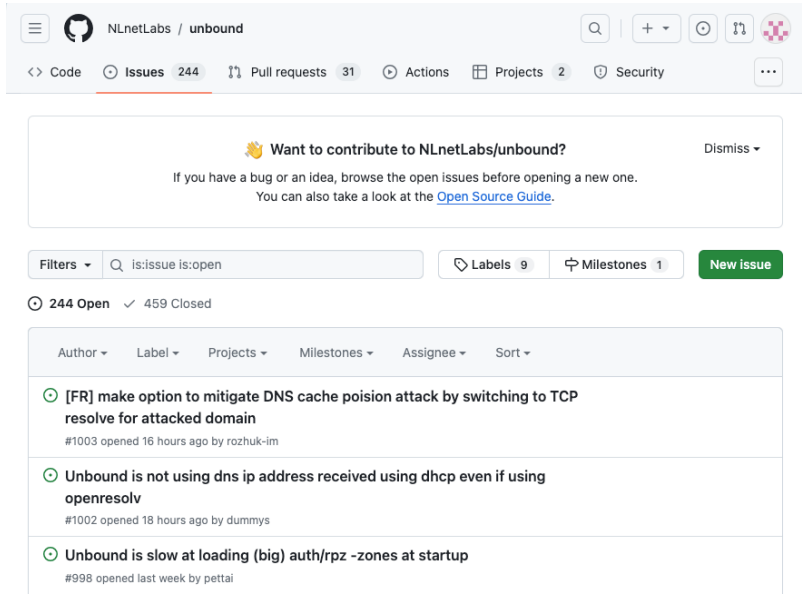
```
{"ip": "1.2.3.4","versions": ["unbound"]}
{"ip": "1.2.3.4","versions": ["technitium-2", "technitium-3"]}
{"ip": "1.2.3.4","versions": ["bind-9.16"]}
{"ip": "1.2.3.4","versions": ["pdns-recuror-4.1.6"]}
```

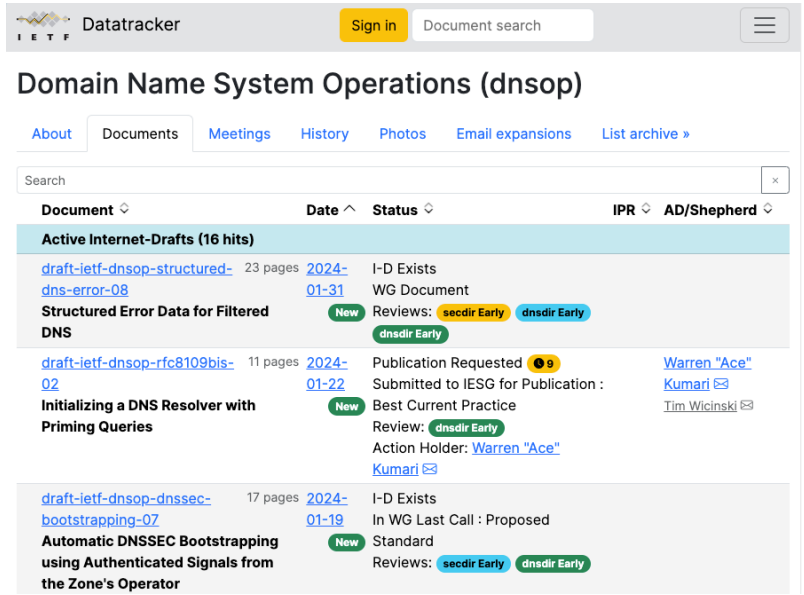


Next steps

Next steps: more test cases







Sources: https://gitlab.isc.org/isc-projects/bind9/-/blob/main/CHANGES?ref_type=heads, https://datatracker.ietf.org/wg/dnsop/documents/, https://github.com/NLnetLabs/unbound/issues

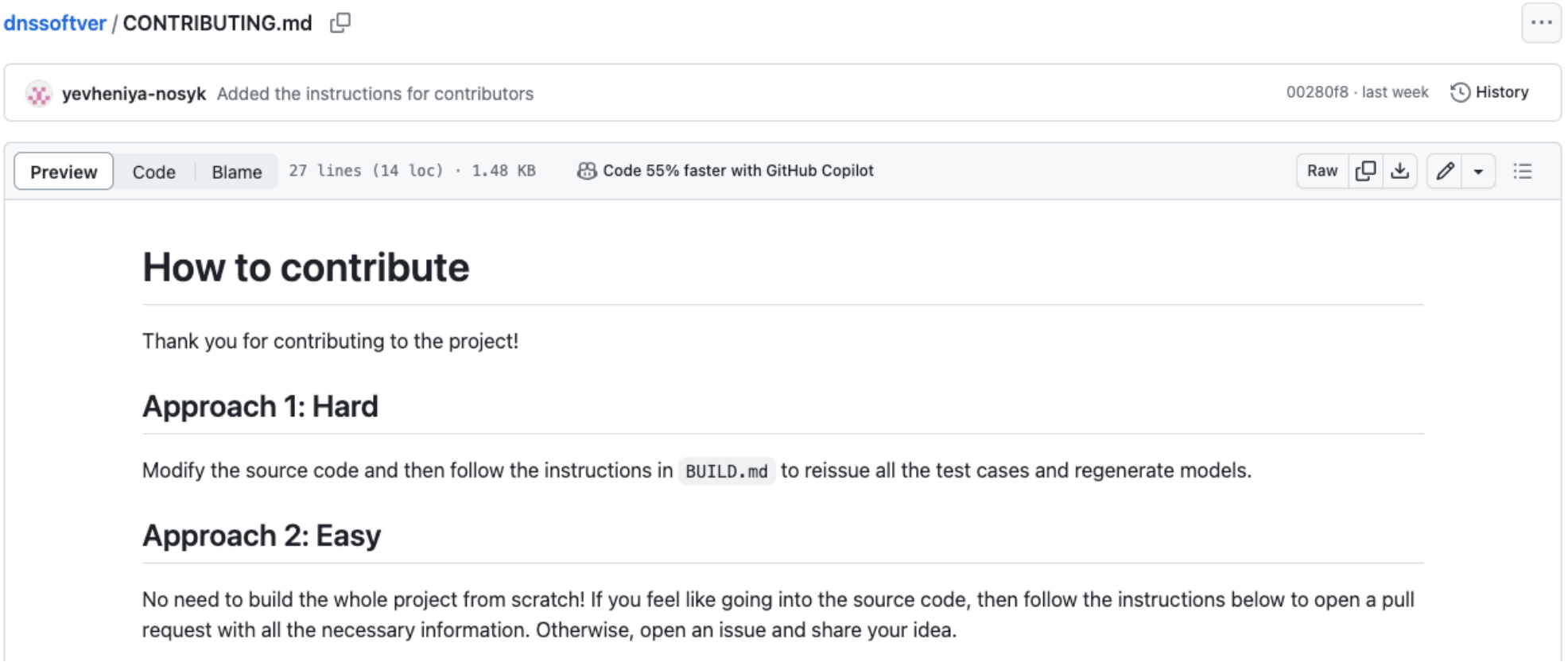
Next steps: less test cases

1. Timeouts are not deterministic
2. Inconsistencies are unpredictable

Next steps: testing in the wild

1. No ground truth data available
2. Need to test on real systems:
    - To confirm the accuracy
    - To catch weird behaviours
    - To adapt the models

Next steps: maintain and update



Next steps: extensions

1. Authoritative nameservers?
2. Forwarders?
3. Forwarders + resolvers?

Acknowledgements

Many thanks to RIPE NCC Community Projects Fund and Université Grenoble Alpes for supporting this project.

Thank you!

https://github.com/yevheniya-nosyk/dnssoftver/

yevheniya.nosyk@univ-grenoble-alpes.fr