ResolverFuzz: Automated Discovery of DNS Resolver Vulnerabilities with Query-Response Fuzzing

Qifan Zhang, Xuesong Bai, Xiang Li, Haixin Duan, Qi Li and Zhou Li

Accepted by USENIX Security 2024

Feel free to visit my homepage (qifanz.com) for slides

Oct 23, 2023

UCI Samueli School of Engineering
University of California, Irvine
Tsinghua University 1911

Short Bio

- **4th-year Ph.D. student of Department of EECS**
  - Advisor: Prof. Dr. Zhou Li
- **Field of Research:**
  - **Domain Name System (DNS)**
    - [Security'24] **Zhang, Q.,** Bai, X., Li, X., Duan, H., Li, Q. and Li, Z., *ResolverFuzz: Automated Discovery of DNS Resolver Vulnerabilities with Query-Response Fuzzing.*
    - [NDSS'23] Li, X., Liu, B., Bai, X., Zhang, M., **Zhang, Q.,** Li, Z., Duan, H. and Li, Q., *Ghost Domain Reloaded: Vulnerable Links in Domain Name Delegation and Revocation.*
    - [Security'23] Li, X., Lu, C., Liu, B., **Zhang, Q.,** Li, Z., Duan, H. and Li, Q., *The Maginot Line: Attacking the Boundary of DNS Caching Protection.*
    - [IEEE Access'22] Liao, X., Xu, J., **Zhang, Q.** and Li, Z., *A Comprehensive Study of DNS Operational Issues by Mining DNS Forums.*

Short Bio

- **4th-year Ph.D. student of Department of EECS**
  - Advisor: Prof. Dr. Zhou Li
- **Field of Research:**
  - **Machine Learning and Security**
    - [ACSAC'22] **Zhang, Q.,** Shen, J., Tan, M., Zhou, Z., Li, Z., Chen, Q.A. and Zhang, H., *Play the Imitation Game: Model Extraction Attack against Autonomous Driving Localization.*
    - [Under review in ICLR'24] Han, S., Buyukates, B., Hu, Z., Jin, H., Jin, W., Sun, L., Wang, X., Xie, C., Zhang, K., **Zhang, Q.** and Zhang, Y., 2023. *FedMLSecurity: A Benchmark for Attacks and Defenses in Federated Learning and Federated LLMs.*
    - [Under review in ICLR'24] Han, S., Wu, W., Buyukates, B., Jin, W., Yao, Y., **Zhang, Q.**, Avestimehr, S. and He, C., 2023. *Kick Bad Guys Out! Zero-Knowledge-Proof-Based Anomaly Detection in Federated Learning.*

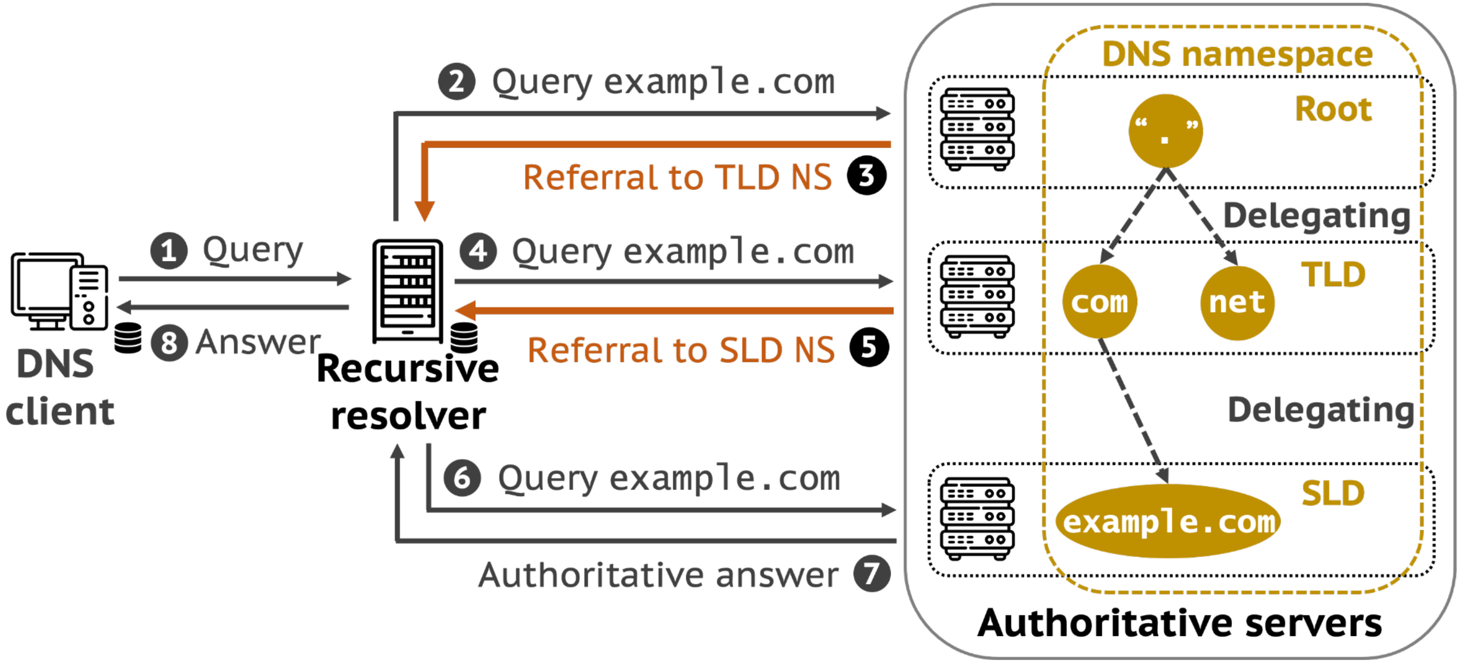
Domain Name System

- **Domain Name System (DNS)**
  - Entry point of many Internet activities
    - Interpret domain names into network addresses (IPs)
    - E.g., translate uci.edu into 128.200.151.40
  - Security guarantee of multiple application services
  - Domain names are widely registered
- **Fundamental for other apps**
  - Web, CDN, Email, Certificate Authentication, etc.

DNS Resolution

➢Recursive/Iterative process

- ➢Multiple roles
- ➢Forwarder, recursive resolver, authoritative server



DNS Resolution

➢Cache Mechanism

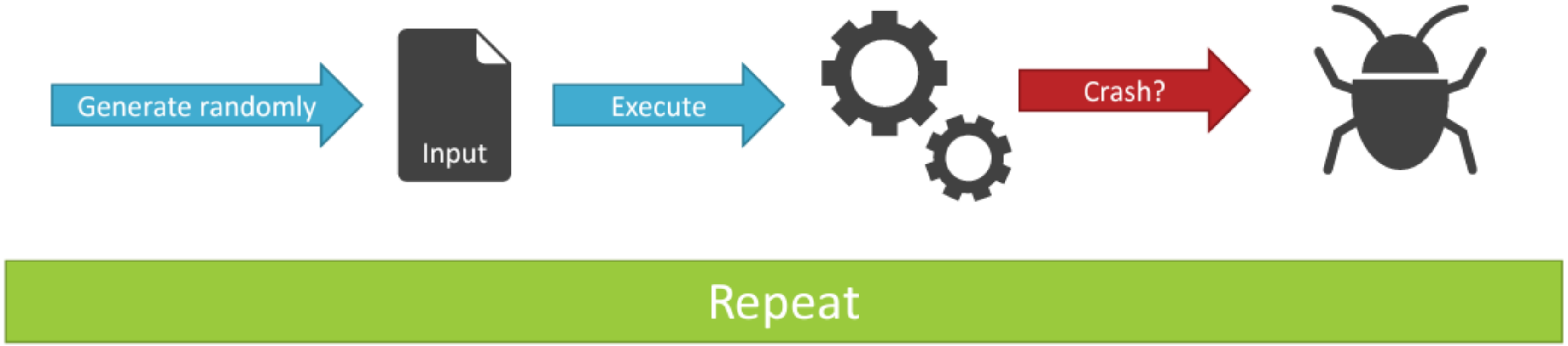
- ➢Cache DNS recourse records (RRs) for future references
- ➢One of the **most vulnerable** parts in DNS
  - ➢Cache poisoning, e.g., MaginotDNS [Security'23], SAD DNS [CCS'20&21]
  - ➢Domain delegation (Ghost Domain), e.g., Phoenix Domain [NDSS'23]
- ➢Only involved for recursive resolvers
  - ➢Focus on **recursive resolvers**

DNS Vulnerability Detection

- **How to find vulnerabilities automatically?**
  - Formal analysis
    - Already applied to nameservers: SCALE [SIGCOMM'22], G-Root [NSDI'20]
    - Lack **rigorous specifications** as references for vulnerability detection
  - **Fuzzing**

Fuzzing

- **Suitable for testing large-size software in large scale**
- **Flexible for multiple scenarios**
  - Lexical-based: Blackbox/Graybox/Whitebox fuzzing
  - Syntactic-based: (Probalistic) Grammar-based fuzzing
  - Semantic-based: Concolic/Symbolic fuzzing



Fuzzing on DNS

- **Previous works**
  - AFL++/AFLNet
  - SnapFuzz [ISSTA'22], DNS Fuzzer (a github repo)
  - Focus on **memory vulnerabilities**
    - Could **only** detect crashes
  - But cache poisoning is <u>**semantic vulnerabilities**</u>
    - Traditional memory-based fuzzers does not work
- **Need to design a fuzzer to detect <u>semantic bugs</u> in DNS**

Which part is more vulnerable?
Where should we focus on?

Check vulnerabilities which **have been** identified
Focus on where they were **most** spotted

Comprehensive Study of CVEs

➢ **Understand the distribution and root causes of DNS-related vulnerabilities**

Table 1: Study results of DNS CVEs for mainstream DNS software.

Software*	# CVE								
	Semantic					Memory			Total
	Cache poison.[1]	Res. consumpt.[2]	Serv. crash[3]	Others	Total	Corrupt.[4]	Others	Total	
BIND	18	17	73	10	118	22	1	23	141
Unbound	4	5	5	3	17	8	1	9	26
Knot Resolver	6	3	2	0	11	0	0	0	11
PowerDNS Recursor	13	7	7	9	36	6	0	6	42
MaraDNS	2	3	3	0	8	7	0	7	15
Technitium	3	1	0	0	4	0	0	0	4
Total	46	36	90	22	194	43	2	45	239

*: Recursive or forwarding modes. [1]: Cache poisoning. [2]: Resource consumption. [3]: Service crash. [4]: Corruption.
CVE of the forwarding mode only: Total (7), BIND (5), Unbound (0), Knot (1), PowerDNS (0), MaraDNS (0), and Technitium (1).
CVE of the authoritative mode only: Total (45), BIND (19), Unbound (4), Knot (2), PowerDNS (19), MaraDNS (1), and Technitium (0).
CVE of other software: Total (131), Microsoft DNS (90), Simple DNS Plus (1), Dnsmasq (33), CoreDNS (1), NSD (4), Yadifa (1), and TrustDNS (1).

Comprehensive Study of CVEs

- **Findings:**
  - Most of the CVEs are about resolvers
    - 284 CVEs, only 45 related to nameservers

Table 1: Study results of DNS CVEs for mainstream DNS software.

Software*	# CVE								
	Semantic					Memory			Total
	Cache poison.[1]	Res. consumpt.[2]	Serv. crash[3]	Others	Total	Corrupt.[4]	Others	Total	
BIND	18	17	73	10	118	22	1	23	141
Unbound	4	5	5	3	17	8	1	9	26
Knot Resolver	6	3	2	0	11	0	0	0	11
PowerDNS Recursor	13	7	7	9	36	6	0	6	42
MaraDNS	2	3	3	0	8	7	0	7	15
Technitium	3	1	0	0	4	0	0	0	4
Total	46	36	90	22	194	43	2	45	239

*: Recursive or forwarding modes. [1]: Cache poisoning. [2]: Resource consumption. [3]: Service crash. [4]: Corruption.
CVE of the forwarding mode only: Total (7), BIND (5), Unbound (0), Knot (1), PowerDNS (0), MaraDNS (0), and Technitium (1).
CVE of the authoritative mode only: Total (45), BIND (19), Unbound (4), Knot (2), PowerDNS (19), MaraDNS (1), and Technitium (0).
CVE of other software: Total (131), Microsoft DNS (90), Simple DNS Plus (1), Dnsmasq (33), CoreDNS (1), NSD (4), Yadifa (1), and TrustDNS (1).

Comprehensive Study of CVEs

- **Findings:**
  - Diversified CVEs among DNS software
    - BIND has the most CVEs
    - Only 13 out of 239 CVEs affect all software

Table 1: Study results of DNS CVEs for mainstream DNS software.

Software*	# CVE								
	Semantic					Memory			Total
	Cache poison.[1]	Res. consumpt.[2]	Serv. crash[3]	Others	Total	Corrupt.[4]	Others	Total	
BIND	18	17	73	10	118	22	1	23	141
Unbound	4	5	5	3	17	8	1	9	26
Knot Resolver	6	3	2	0	11	0	0	0	11
PowerDNS Recursor	13	7	7	9	36	6	0	6	42
MaraDNS	2	3	3	0	8	7	0	7	15
Technitium	3	1	0	0	4	0	0	0	4
Total	46	36	90	22	194	43	2	45	239

*: Recursive or forwarding modes. [1]: Cache poisoning. [2]: Resource consumption. [3]: Service crash. [4]: Corruption.
CVE of the forwarding mode only: Total (7), BIND (5), Unbound (0), Knot (1), PowerDNS (0), MaraDNS (0), and Technitium (1).
CVE of the authoritative mode only: Total (45), BIND (19), Unbound (4), Knot (2), PowerDNS (19), MaraDNS (1), and Technitium (0).
CVE of other software: Total (131), Microsoft DNS (90), Simple DNS Plus (1), Dnsmasq (33), CoreDNS (1), NSD (4), Yadifa (1), and TrustDNS (1).

Comprehensive Study of CVEs

➢Findings:

- ➢Most of the CVEs are semantic bugs
  - ➢Cache poisoning, resource consumption and service crash

Table 1: Study results of DNS CVEs for mainstream DNS software.

Software*	# CVE								
	Semantic					Memory			Total
	Cache poison.[1]	Res. consumpt.[2]	Serv. crash[3]	Others	Total	Corrupt.[4]	Others	Total	
BIND	18	17	73	10	118	22	1	23	141
Unbound	4	5	5	3	17	8	1	9	26
Knot Resolver	6	3	2	0	11	0	0	0	11
PowerDNS Recursor	13	7	7	9	36	6	0	6	42
MaraDNS	2	3	3	0	8	7	0	7	15
Technitium	3	1	0	0	4	0	0	0	4
Total	46	36	90	22	194	43	2	45	239

*: Recursive or forwarding modes. [1]: Cache poisoning. [2]: Resource consumption. [3]: Service crash. [4]: Corruption.
CVE of the forwarding mode only: Total (7), BIND (5), Unbound (0), Knot (1), PowerDNS (0), MaraDNS (0), and Technitium (1).
CVE of the authoritative mode only: Total (45), BIND (19), Unbound (4), Knot (2), PowerDNS (19), MaraDNS (1), and Technitium (0).
CVE of other software: Total (131), Microsoft DNS (90), Simple DNS Plus (1), Dnsmasq (33), CoreDNS (1), NSD (4), Yadifa (1), and TrustDNS (1).

Comprehensive Study of CVEs

- **Findings:**
  - Nearly every field of a DNS message has related CVEs
    - Query name, query type, query flag, RCODE, RDATA, TTL, etc.
  - Most of the CVEs are triggered with short message sequence

Table 1: Study results of DNS CVEs for mainstream DNS software.

Software*	# CVE								
	Semantic					Memory			Total
	Cache poison.[1]	Res. consumpt.[2]	Serv. crash[3]	Others	Total	Corrupt.[4]	Others	Total	
BIND	18	17	73	10	118	22	1	23	141
Unbound	4	5	5	3	17	8	1	9	26
Knot Resolver	6	3	2	0	11	0	0	0	11
PowerDNS Recursor	13	7	7	9	36	6	0	6	42
MaraDNS	2	3	3	0	8	7	0	7	15
Technitium	3	1	0	0	4	0	0	0	4
Total	46	36	90	22	194	43	2	45	239

*: Recursive or forwarding modes. [1]: Cache poisoning. [2]: Resource consumption. [3]: Service crash. [4]: Corruption.
CVE of the forwarding mode only: Total (7), BIND (5), Unbound (0), Knot (1), PowerDNS (0), MaraDNS (0), and Technitium (1).
CVE of the authoritative mode only: Total (45), BIND (19), Unbound (4), Knot (2), PowerDNS (19), MaraDNS (1), and Technitium (0).
CVE of other software: Total (131), Microsoft DNS (90), Simple DNS Plus (1), Dnsmasq (33), CoreDNS (1), NSD (4), Yadifa (1), and TrustDNS (1).

How should we design ResolverFuzz?

Black box, **Stateful** and **Grammar**-based fuzzing
Two input generators
Identify diff. vuln. by adapting diff. **oracles**

ResolverFuzz Infrastructure

- **Input:**
  - Query Generator
  - Response Generator

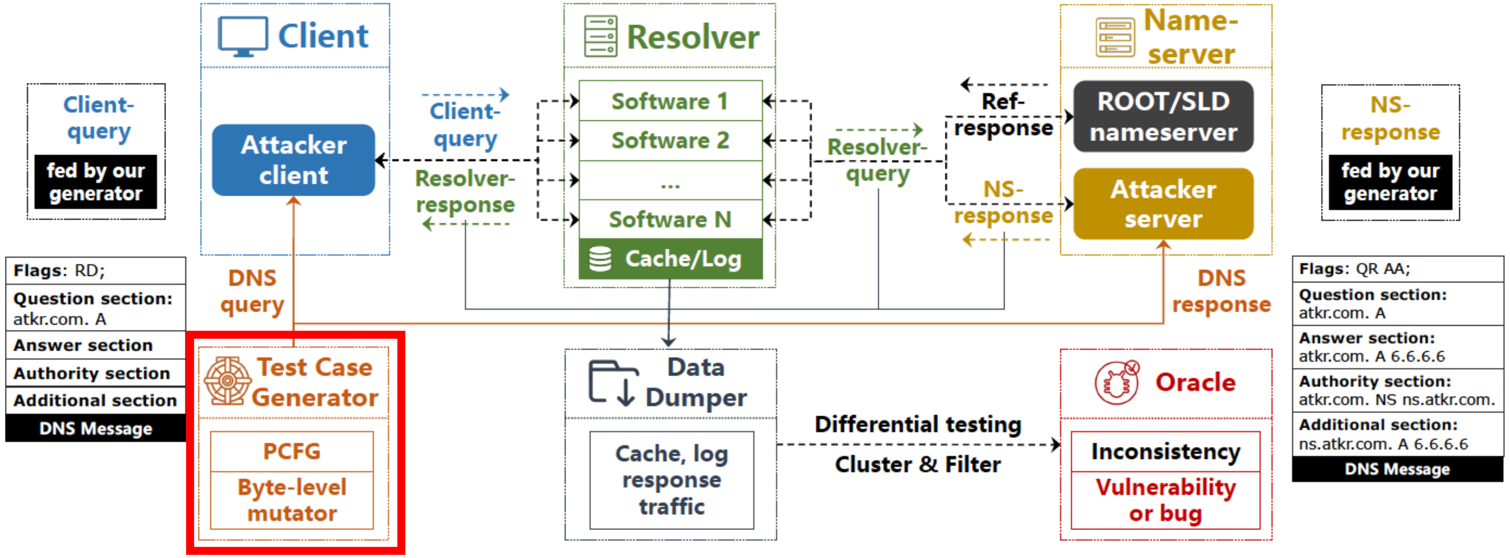


Figure 3: Workflow of RESOLVERFUZZ.

ResolverFuzz Infrastructure

- **Output:**
  - Response
  - Cache
  - System logs

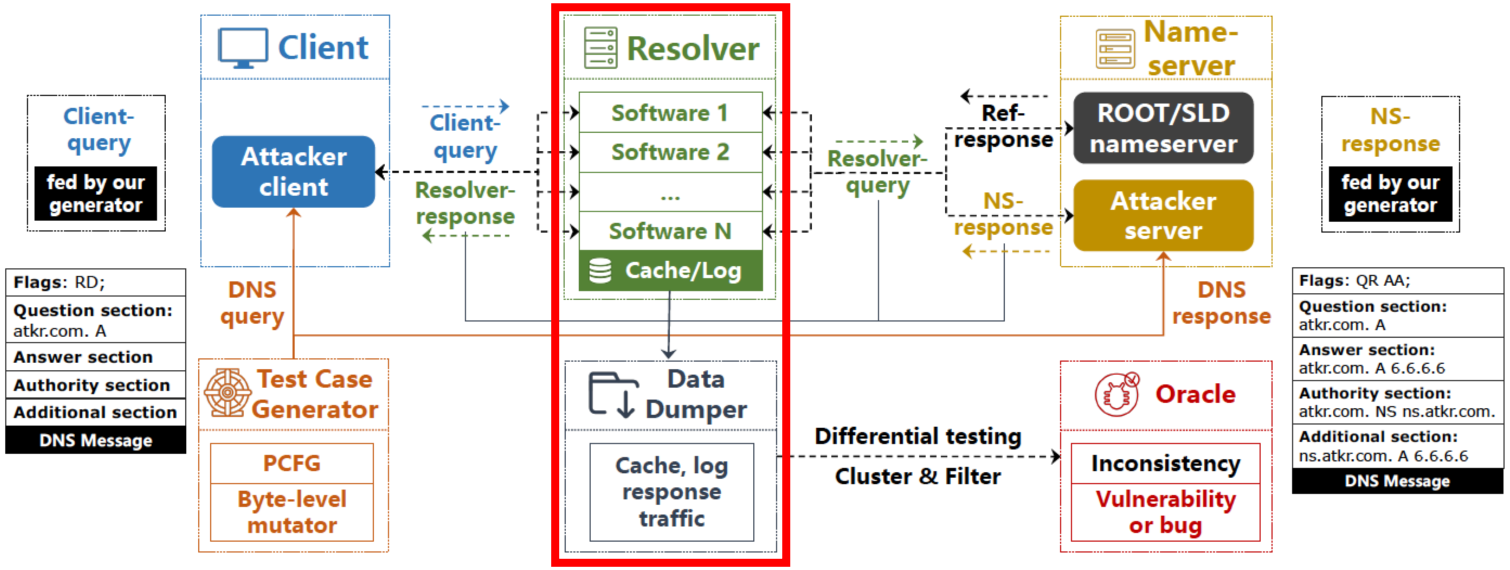


Figure 3: Workflow of RESOLVERFUZZ.

ResolverFuzz Infrastructure

➢Oracle:

- ➢Measure divergence
- ➢Bug/vuln. analysis

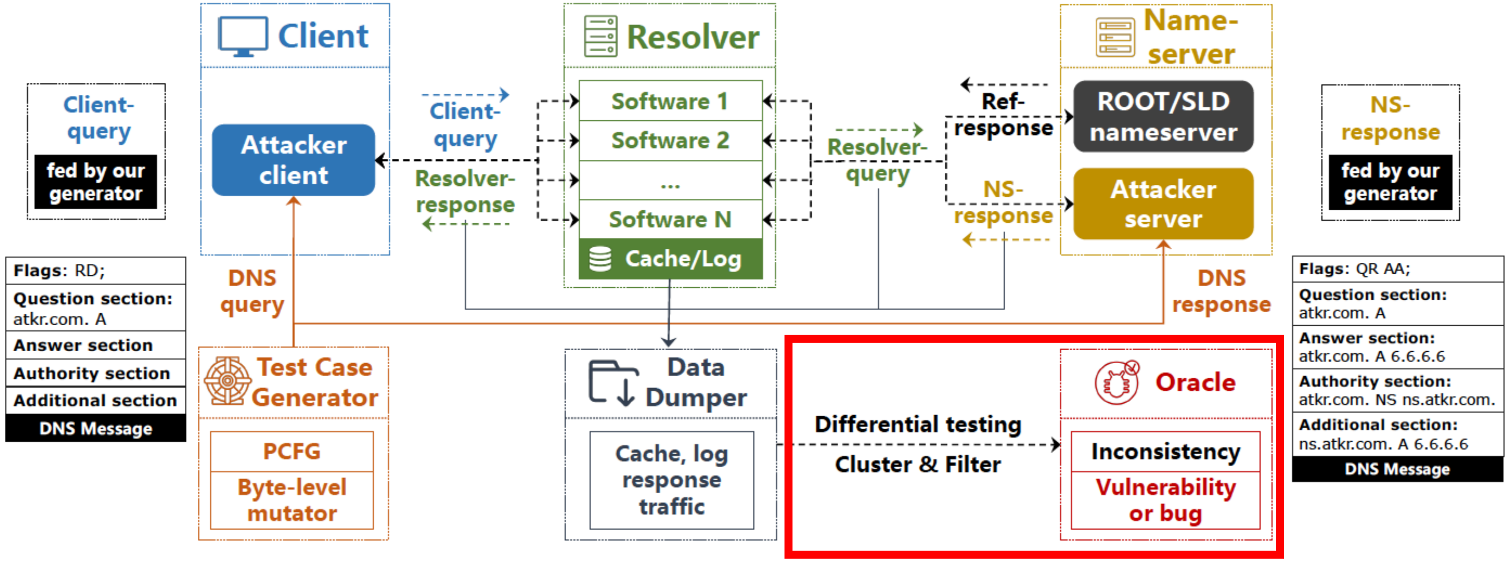


Figure 3: Workflow of RESOLVERFUZZ.

ResolverFuzz: Workflow

- ➢ **Initialize DNS Resolvers**
- ➢ **Test case generation**
  - ➢ Query & Responses
- ➢ **Test case execution**
  - ➢ Data dump
- ➢ **Reset for next round**
- ➢ **Differential analysis**

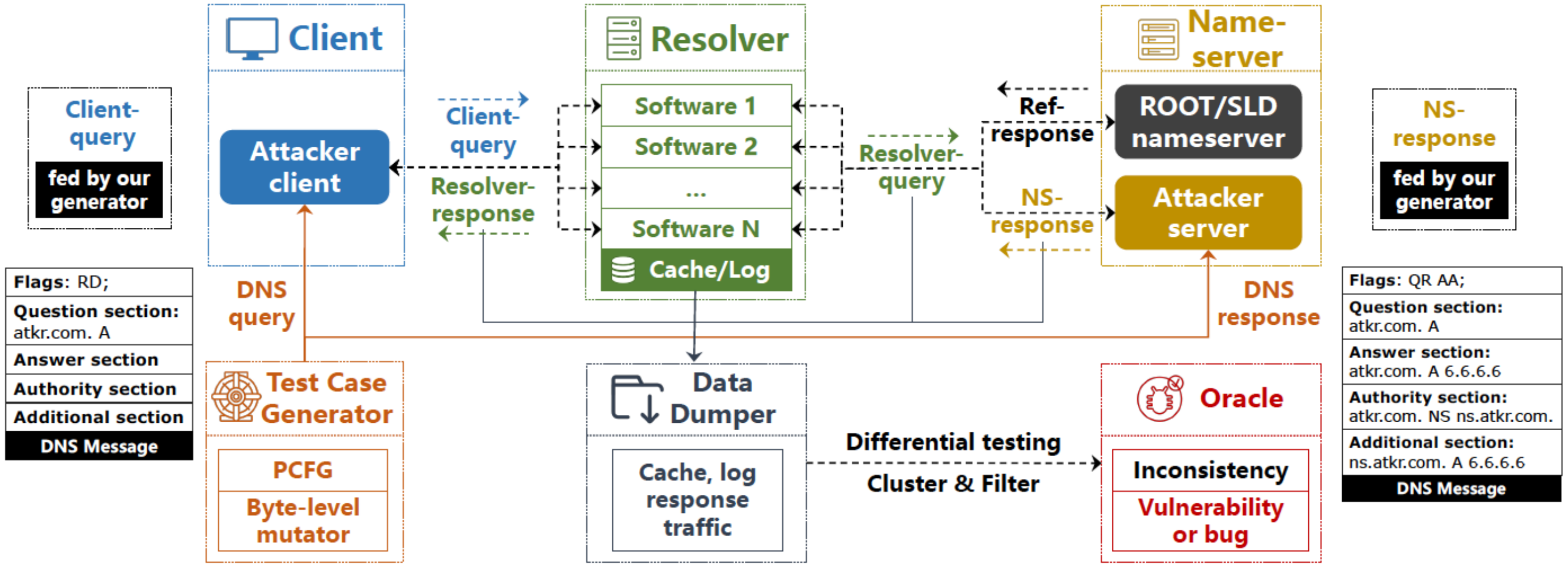


Figure 3: Workflow of RESOLVERFUZZ.

What are the challenges for ResolverFuzz?

Efficiency
Mutation
Stateful Fuzzing
Oracle

Efficiency

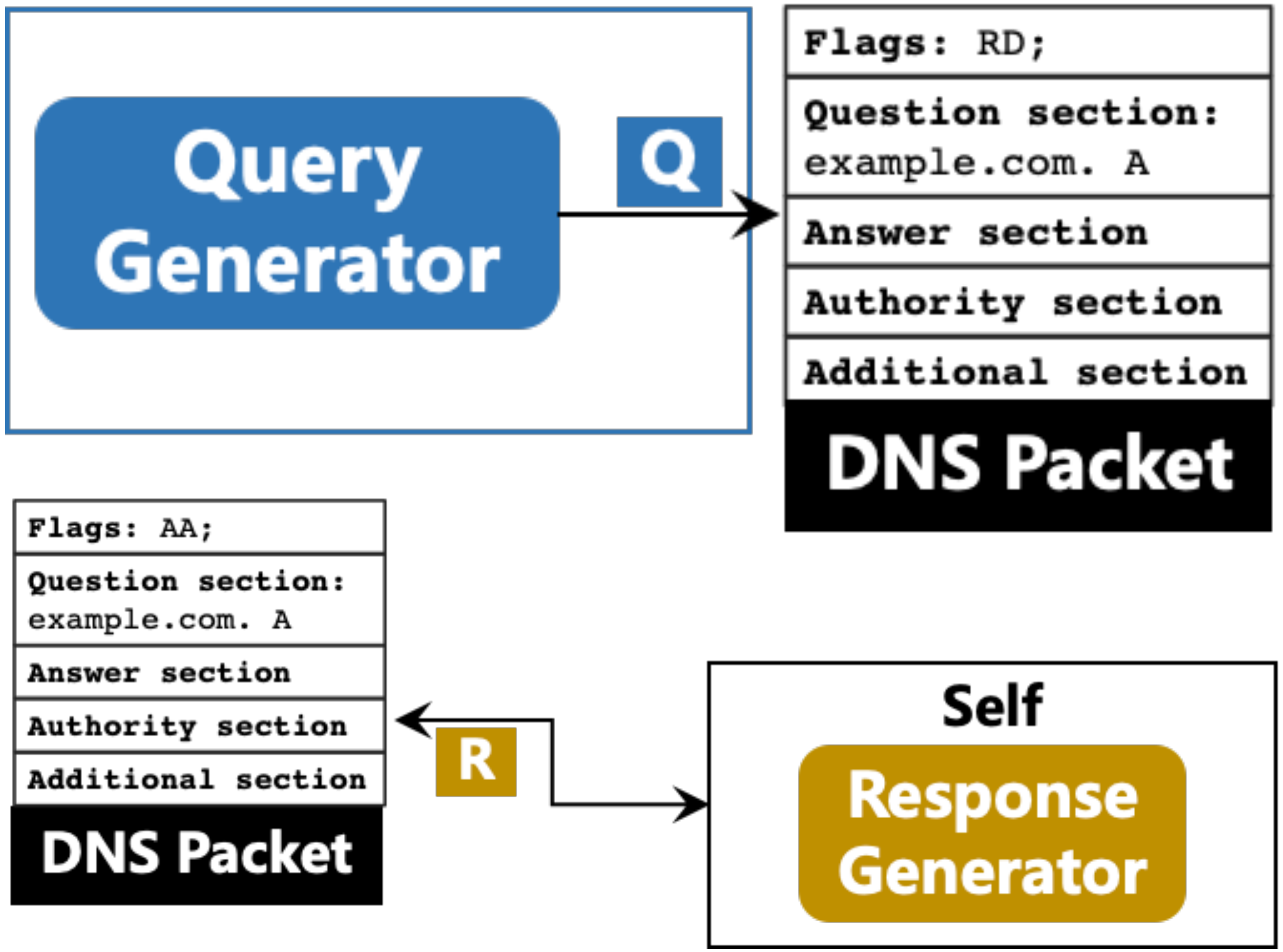
- **Some DNS software are slow**
  - E.g., BIND (~0.4s per query) v.s. PowerDNS (>1s per query)
- **Empty cache for each test**
- **Preset timeouts**
- **Pre- and post-processing**
  - NS initialization
  - Data collection
- **Solution: Run several test units in parallel**
  - "High efficiency via high **throughput**"

Mutation

- **Coverage-based fuzzers**
  - Fail to provide sufficient guidance
  - Poor on deciding which part should be mutated
  - **Reason**: no preliminary knowledge on DNS packets
- **Input dimension**
  - Only one dimension (query or NS response) leads to many invalid tests

Input Generation

- **Two dimensions**
  - Client-queries
    - For attacker clients
  - Nameserver (NS)-responses
    - For attacker NSes



Input Generation

- Grammar-based Fuzzing
  - Probabilistic context-free grammar (PCFG)
    - Queries and Responses
  - High prob. for certain fields
    - Guide fuzzing process

```
⟨start⟩ ::= ⟨query⟩
⟨query⟩ ::= ⟨Header⟩⟨Question⟩
⟨Header⟩ ::= ⟨TransactionID⟩⟨Flags⟩⟨RRs⟩
⟨TransactionID⟩ ::= (randomly generated 2-byte hex value)
⟨Flags⟩ ::= ⟨QR⟩⟨OPCODE⟩⟨AA⟩⟨TC⟩⟨RD⟩⟨RA⟩⟨Z⟩⟨AD⟩⟨CD⟩⟨RCODE⟩
⟨QR⟩ ::= 0
⟨OPCODE⟩ ::= QUERY[.80] | IQUERY[.04] | STATUS[.04] |
    NOTIFY[.04] | UPDATE[.04] | DSO[.04]
⟨AA⟩ ::= 0 | 1
⟨TC⟩ ::= 0 | 1
⟨RD⟩ ::= 0 | 1
⟨RA⟩ ::= 0 | 1
⟨Z⟩ ::= 0 | 1
⟨AD⟩ ::= 0 | 1
⟨CD⟩ ::= 0 | 1
⟨RCODE⟩ ::= NOERROR[.80] | FORMERR[.01] | SERVFAIL[.01] |
    NXDOMAIN[.01] | NOTIMP[.01] | REFUSED[.01] | YXDOMAIN
    [.01] | YXRRSET[.01] | NXRRSET[.01] | NOTAUTH[.01] |
    NOTZONE[.01] | DSOTYPENI[.01] | BADVERS[.01] | BADKEY
    [.01] | BADTIME[.01] | BADMODE[.01] | BADNAME[.01] |
    BADALG[.01] | BADTRUNC[.01] | BADCOOKIE[.01]
⟨RRs⟩ ::= ⟨QDCOUNT⟩⟨ANCOUNT⟩⟨NSCOUNT⟩⟨ARCOUNT⟩
⟨QDCOUNT⟩ ::= 1
⟨ANCOUNT⟩ ::= 0
⟨NSCOUNT⟩ ::= 0
⟨ARCOUNT⟩ ::= 0
⟨Question⟩ ::= ⟨QNAME⟩⟨QTYPE⟩⟨QCLASS⟩
⟨QNAME⟩ ::= (base domain)[.40] |
          (sub-domain)[.40] |
          (2-9th sub-domain)[.10] |
          (10-max sub-domain)[.10] |
⟨QTYPE⟩ ::= A | NS | CNAME | SOA | PTR | MX | TXT | AAAA |
     RRSIG | SPF | ANY
⟨QCLASS⟩ ::= IN
```

Listing 1: PCFG for DNS query.

Input Generation

- **Byte-level mutation**
  - Some DNS implementations fail to correctly decode strings with **special characters** embedded
    - E.g., \., \000, @, /, and \
    - Jeitner et al. [Security'21]
  - Addition, deletion, and replacement
    - After PCFG test generation

Stateful Fuzzing

- **DNS resolvers are <u>stateful</u>**
  - Depending on cache records, configurations, etc.
  - Major challenge for network fuzzing
    - Large search space of input sequences
- **Solution:**
  - Generate one pair of the query and (authoritative) response
    - Cover most vulnerable cases
  - Deploy the auth. response on the NS side
  - Start to test by sending the query
    - Communication between DNS resolvers and the NS
    - Preset timeout (5s) is deployed

Oracle

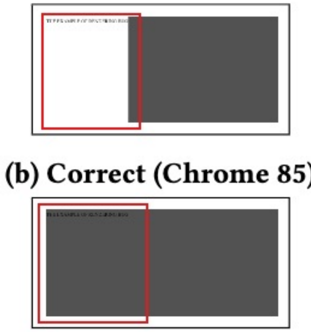
- **Lack an oracle to detect semantic bugs**
  - Memory bugs have their oracle
    - E.g., AddressSanitizer [USENIX ATC'12]
- **Differential testing**
  - Used for memory bugs, but none for DNS
- **How to connect inconsistency with vulnerabilities?**
  - Inconsistencies are common in DNS
  - Many of them do not indicate vulnerabilities

Differential Analysis

- **Runs multiple programs, comparing their outputs for the same input**
  - Detecting rendering regressions in browsers (e.g., R2Z2 [ICSE'22])
  - Comparing outputs from different versions
  - Efficient to find divergences

```html
<html>
  <style>
  .class7 {
    backdrop-filter: hue-rotate(1deg);
    filter: brightness(0.32269068);
    padding: 66%;
  }
  </style>
  <body>
  THE EXAMPLE OF RENDERING BUG
    <span class="class7" ></span>
  </body>
</html>
```

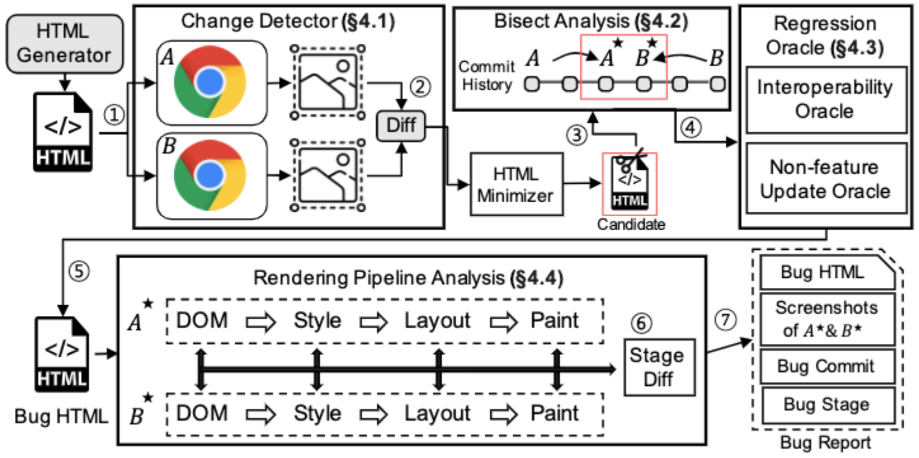
(a) PoC HTML code



(b) Correct (Chrome 85)

(c) Incorrect (Chrome 86)

Figure 2: A rendering bug example (Chrome Issue #1122021).



Oracle

- **Different DNS software**
  - Objects of differential analysis
- **Three Oracles**
  - Cache poisoning oracle
    - Semi-automatic, differential-analysis based
    - Record the max # different records of one software from the others
    - Cluster by Bisecting K-Means
    - Manually check each cluster to identify possible vulnerabilities

Oracle

Three Oracles

- Resource consumption oracle
  - 4 metrics:
    - # queries
    - Sizes of responses
    - Resolution timeout
    - Frequency of internal operations (e.g., cache search)
  - Compare metrics with the value distribution in normal cases

Oracle

- **Three Oracles**
  - Crash & Corruption oracle
    - Monitor DNS software processes
    - Check if the process is running after each test case

How does ResolverFuzz perform?

Tested in **6** popular DNS software and **4** popular modes
Good coverage of different field values
Efficient runtime performance

Evaluation

- **6 DNS software**
  - BIND 9, Unbound, PowerDNS, Knot, Technitium and MaraDNS
  - Docker-based
  - Schedulers and oracles implemented in Python

Evaluation

➢4 configurations:

- ➢Recur.-only, Fwd-only, CDNS w/ fallback and CDNS w/o fallback

```
options {
	recursion yes;
	// includes the entire namespace
}
```

(a)

```
options {
	recursion no;
	// disables recursive resolution
	forwarders {
		x.x.x.x port 53;
	}
	// forward the entire zone "." to an upstream server
}
```

(b)

```
options {
	recursion yes;
}
// create a forward zone for test-cdns.example.com
zone "test-cdns.example.com" {
	type forward;
	forwarders { x.x.x.x port 53; };
	forward only; // fallback mode disabled
}
```

(c)

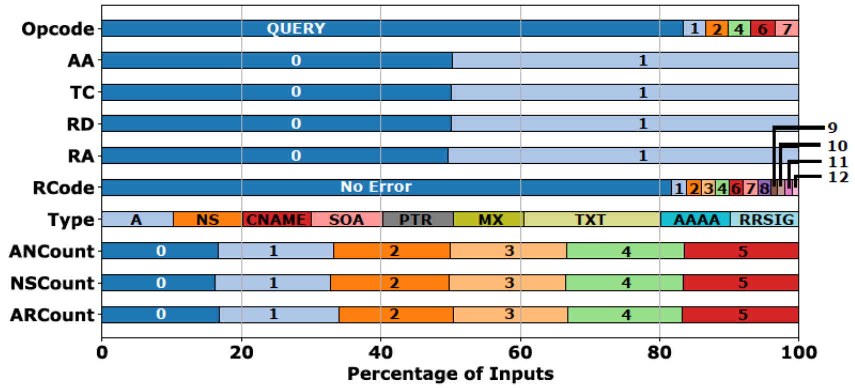
```
options {
	recursion yes;
}
// create a forward zone for test-cdns.example.com
zone "test-cdns.example.com" {
	type forward;
	forwarders { x.x.x.x port 53; };
	forward first; // fallback mode enabled
}
```

(d)

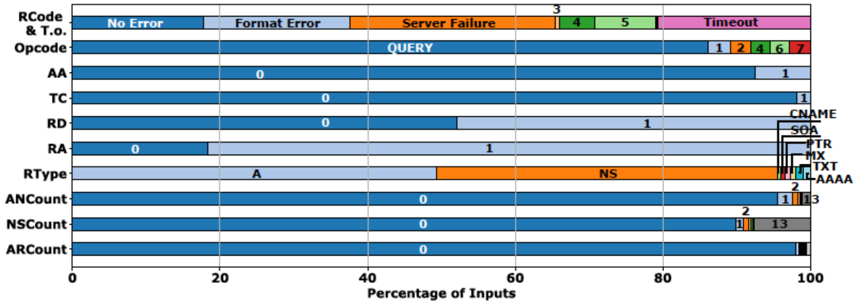
Figure 11: Example BIND configs of a) recursive-only, b) forward-only, c) CDNS without fallback, and d) CDNS with fallback.

Evaluation

- **Analysis of tests generation**
  - Good coverage of different field values
  - Rule probabilities of PCFG
    - Test certain code logic more intensively
  - Test cases prone to trigger errors
    - Potentially bugs
    - Only 17.8% have RCODE=NOERROR



(a) Client-queries and NS-responses.



(b) Resolver-responses. "*RCode & T.o.*" refers to "RCODE and Timeouts".

Figure 6: Input coverage analysis on: a) client-queries and ns-responses; b) resolver-responses. The client-query and ns-response have the similar distribution for fields from `OPCODE` to `TYPE`. `AN/NS/ARCOUNT` applies to ns-responses. The values marked on bars are standard DNS values from [78].

Evaluation

➢Runtime performance

- ➢Use concurrency to speed up
  - ➢5.9 QPS (CDNS w/ f.b.)
    - ➢ BIND and Unbound only
  - ➢2.8 QPS (other modes)
    - ➢ MaraDNS, PowerDNS: low on efficiency
- ➢Similar speed with real-world DNS resolution
  - ➢Google DNS: 300-400 ms per query
    - ➢ i.e., 2.5-3.3 QPS

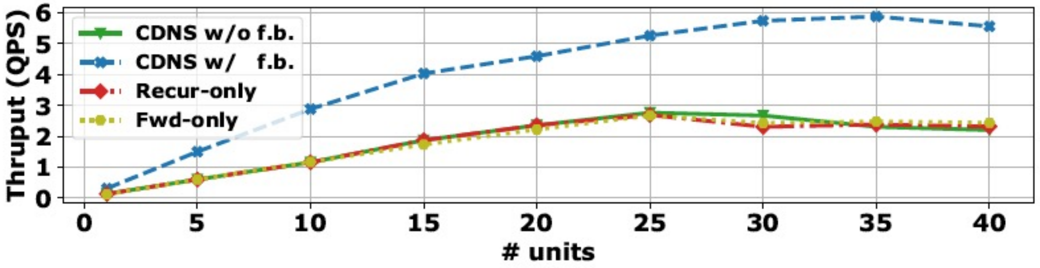


Figure 7: Throughput ("*Thruput*") of 4 modes with regard to the number of units. *CDNS w/o f.b.*, *CDNS w/ f.b.*, *Recur-only* and *Fwd-only* refers to *CDNS without fallback*, *CDNS with fallback*, *Recursive-only*, and *Forwarder-only*.

How many new vuln. are discovered?

23 vulnerabilities identified
19 confirmed, **15** CVEs assigned
Categorized into 3 classes

Discovered Vulnerabilities

- **23 vulnerabilities identified**
  - 19 vulnerabilities confirmed
  - 15 CVEs assigned
  - Details available in the paper

Table 2: Identified bugs and test cases of six mainstream DNS software.

Software[*]	Cache poisoning					Resource consumption								Crash& Corruption	Total
	CP1	CP2	CP3	CP4[1]	Tot.[2]	RC1	RC2	RC3	RC4	RC5	RC6	RC7	Tot.	CC1	
BIND	✓[†]	✗	✓	✓	3	✗	✗	✗	✗	✗	✗	✗	0	✓	4
Unbound	✗	✗	✓	✓[†]	2	✗	✓	✓	✗	✓	✓	✗	4	-	6
Knot	✓[†]	✗	✓	✓[†]	3	✗	✗	✗	✗	✗	✗	✓	1	-	4
PowerDNS	✗	✓	✗	✓[†]	2	✓	✗	✓	✗	✗	✗	✗	2	-	4
MaraDNS	✗	✗	-	✓[†]	1	✗	✗	✗	✓[†]	✗	✗	✗	1	-	2
Technitium	✓[†]	✗	-	✓[†]	2	✗	✗	✗	✓[†]	✗	✗	✗	1	-	3
Total	3	1	3	6	13	1	2	1	2	1	1	1	9	1	23

[*]: Recursive or forwarding modes. [1]: They are triggered by different responses and their cache are inconsistent. [2]: Total.
✓or ✓: Vulnerable. ✓: In discussion. ✓: Confirmed and/or fixed by vendors. ✗: Not vulnerable. [†]: CVEs are assigned. '-': Not applicable.
Amount of test case: *CP*1 (19), *CP*2 (1,422), *CP*3 (111,328), *CP*4 (7,856), *RC*1 (539,745), *RC*2 (112,126), *RC*3 (88,935), *RC*4 (132), *RC*5 (272) *RC*6 (6,264), *RC*7 (4,448), and *CC*1 (5).

CP1: Out-of-Bailiwick Cache Poisoning

- **Bailiwick rule**
  - NS **should not** return RRs out of **their controlled zone**
  - E.g., RRs from .com server should not contain .org RRs

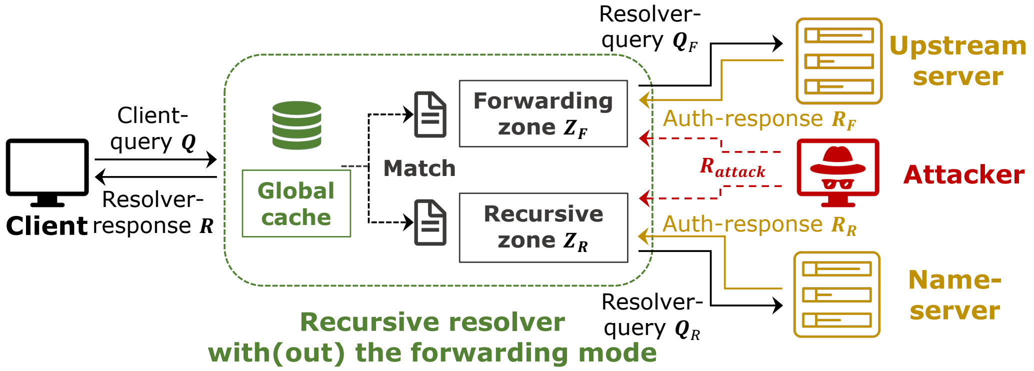
Header: TXID; QR AA;
Question section: atkr-fwd.com. A
Answer section: atkr-fwd.com. A x.x.x.x
Authority section: com. NS ns.atkr-fwd.com.
Additional section: ns.atkr-fwd.com. A **a.t.k.r**

CP1: Out-of-Bailiwick Cache Poisoning

➢Out-of-Bailiwick attack

- ➢ First found in BIND under CDNS without fallback mode
  - ➢ Also identified in Knot and Technitium
- ➢ Forged NS records with AA Flag have **higher** trust level
- ➢ Resolvers may overwrite cached records with the forged one
  - ➢ Some DNS resolver do not check the response
- ➢ Hijack the whole .com zone into ns.atkr-fwd.com
- ➢ Details analyzed in MaginotDNS [Security'23]

Header: TXID; QR AA;
Question section: atkr-fwd.com. A
Answer section: atkr-fwd.com. A x.x.x.x
Authority section: com. NS ns.atkr-fwd.com.
Additional section: ns.atkr-fwd.com. A **a.t.k.r**



RC1: Excessive cache search operations

- **Forward-only mode, PowerDNS**
- **Looks up its local cache for trust anchors and NS records before sending it to a server**
  - E.g., s.atkr-fwd.com
    - Should be only one search only
  - PowerDNS: search records in the order of s.atkr-fwd.com, atkr-fwd.com, .com and root servers
    - Until an NS record is found
  - May cause resource consumption due to excessive cache search

Conclusion

- **Comprehensive study of published DNS CVEs**
- **Develop a blackbox fuzzing system for DNS resolvers**
- **Novel techniques**
  - Stateful fuzzing
  - Differential testing
  - Grammar-based fuzzing
- **12 types of vulnerabilities and 15 CVEs assigned**

Thanks for listening!

Any questions?

Qifan Zhang, Department of EECS, UC Irvine
qifan.zhang@uci.edu

Discovered Vulnerabilities

➢Cache poisoning (CP)

- ➢CP1: Out-of-bailiwick cache poisoning
- ➢CP2: In-bailiwick cache poisoning
- ➢CP3: Fragmentation-based cache poisoning
- ➢CP4: Iterative subdomain caching

Table 2: Identified bugs and test cases of six mainstream DNS software.

Software*	Cache poisoning					Resource consumption								Crash& Corruption	Total
	CP1	CP2	CP3	CP4[1]	Tot.[2]	RC1	RC2	RC3	RC4	RC5	RC6	RC7	Tot.	CC1	
BIND	✓†	✗	✓	✓	3	✗	✗	✗	✗	✗	✗	✗	0	✓	4
Unbound	✗	✗	✓	✓†	2	✗	✓	✓	✗	✓	✓	✗	4	-	6
Knot	✓†	✗	✓	✓†	3	✗	✗	✗	✗	✗	✗	✓	1	-	4
PowerDNS	✗	✓	✗	✓†	2	✓	✗	✓	✗	✗	✗	✗	2	-	4
MaraDNS	✗	✗	-	✓†	1	✗	✗	✗	✓†	✗	✗	✗	1	-	2
Technitium	✓†	✗	-	✓†	2	✗	✗	✗	✓†	✗	✗	✗	1	-	3
Total	3	1	3	6	13	1	2	1	2	1	1	1	9	1	23

*: Recursive or forwarding modes. [1]: They are triggered by different responses and their cache are inconsistent. [2]: Total.
✓or ✓: Vulnerable. ✓: In discussion. ✓: Confirmed and/or fixed by vendors. ✗: Not vulnerable. †: CVEs are assigned. '-': Not applicable.
Amount of test case: *CP1* (19), *CP2* (1,422), *CP3* (111,328), *CP4* (7,856), *RC1* (539,745), *RC2* (112,126), *RC3* (88,935), *RC4* (132), *RC5* (272) *RC6* (6,264), *RC7* (4,448), and *CC1* (5).

Header: TXID; QR AA;
Question section: atkr-fwd.com. A
Answer section: atkr-fwd.com. A x.x.x.x
Authority section: com. NS ns.atkr-fwd.com.
Additional section: ns.atkr-fwd.com. A **a.t.k.r**

(a) Auth-response for *CP1*.

Header: TXID; QR AA;
Question section: vctm-fwd.com. A
Answer section: vctm-fwd.com. A x.x.x.x
Authority section: s.vctm-fwd.com. NS ns.vctm-fwd.com.
Additional section: ns.vctm-fwd.com. A **a.t.k.r**

(b) Auth-response for *CP2*.

Header: TXID; QR AA;
Answer section: victim.com. A x.x.x.x ... victim.com. RRSIG xxx...x

(c) 1st fragment for *CP3*.

Authority section: victim.com. NS ns.victim.com.
Additional section: ns.victim.com. A **a.t.k.r**

(d) spoofed 2rd fragment for *CP3*.

Header: TXID; QR AA;
Question section: s.atkr-rev.com. A
Answer section: s.atkr-rev.com. A **a.t.k.r**
Authority section: s.atkr-rev.com. NS ns.atkr-rev.com.
Additional section: ns.atkr-rev.com. A **a.t.k.r**

(e) Auth-response for *CP4*.

Header: TXID; QR AA;
Question section: s.atkr-rev.com. A
Answer section: (Empty)
Authority section: s.atkr-rev.com. NS ns.atkr-rev.com.
Additional section: ns.atkr-rev.com. A **a.t.k.r**

(f) Ref-response for *CP4*.

Figure 9: DNS responses utilized for cache poisoning attacks. Red parts carry the attacking payloads.

Discovered Vulnerabilities

➢Resource Consumption Bugs (RC)

- ➢RC1: Excessive cache search operations
- ➢RC2: Unlimited cache store operations
- ➢RC3: Ignoring the RD flag
- ➢RC4: Following a self-CNAME reference
- ➢RC5: Large responses to clients
- ➢RC6: Overlong waiting time over UDP
- ➢RC7: Excessive queries for resolution over TCP

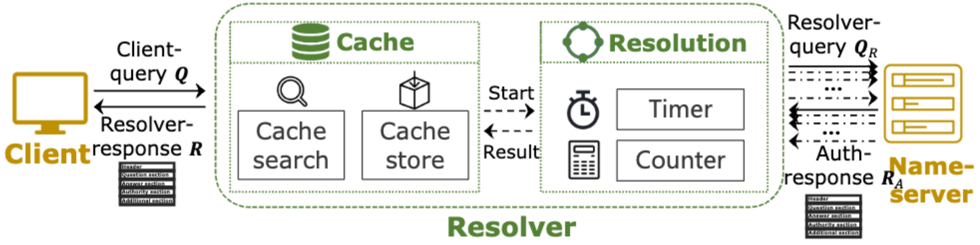


Figure 10: Threat model of resource consumption bugs.

Discovered Vulnerabilities

➢Crash & Corruption Bugs

➢Assertion failure when receiving queries

Table 2: Identified bugs and test cases of six mainstream DNS software.

Software[*]	Cache poisoning					Resource consumption								Crash& Corruption	Total
	CP1	CP2	CP3	CP4[1]	Tot.[2]	RC1	RC2	RC3	RC4	RC5	RC6	RC7	Tot.	CC1	
BIND	✓[†]	✗	✓	✓	3	✗	✗	✗	✗	✗	✗	✗	0	✓	4
Unbound	✗	✗	✓	✓[†]	2	✗	✓	✓	✗	✓	✓	✗	4	-	6
Knot	✓[†]	✗	✓	✓[†]	3	✗	✗	✗	✗	✗	✗	✓	1	-	4
PowerDNS	✗	✓	✗	✓[†]	2	✓	✗	✓	✗	✗	✗	✗	2	-	4
MaraDNS	✗	✗	-	✓[†]	1	✗	✗	✗	✓[†]	✗	✗	✗	1	-	2
Technitium	✓[†]	✗	-	✓[†]	2	✗	✗	✗	✓[†]	✗	✗	✗	1	-	3
Total	3	1	3	6	13	1	2	1	2	1	1	1	9	1	23

[*]: Recursive or forwarding modes. [1]: They are triggered by different responses and their cache are inconsistent. [2]: Total.
✓or ✓: Vulnerable. ✓: In discussion. ✓: Confirmed and/or fixed by vendors. ✗: Not vulnerable. [†]: CVEs are assigned. '-': Not applicable.
Amount of test case: *CP*1 (19), *CP*2 (1,422), *CP*3 (111,328), *CP*4 (7,856), *RC*1 (539,745), *RC*2 (112,126), *RC*3 (88,935), *RC*4 (132), *RC*5 (272) *RC*6 (6,264), *RC*7 (4,448), and *CC*1 (5).