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

Charlotte, NC, USA

Feb 9, 2024

UCI Samueli School of Engineering
University of California, Irvine
清华大学 TSINGHUA UNIVERSITY 1911

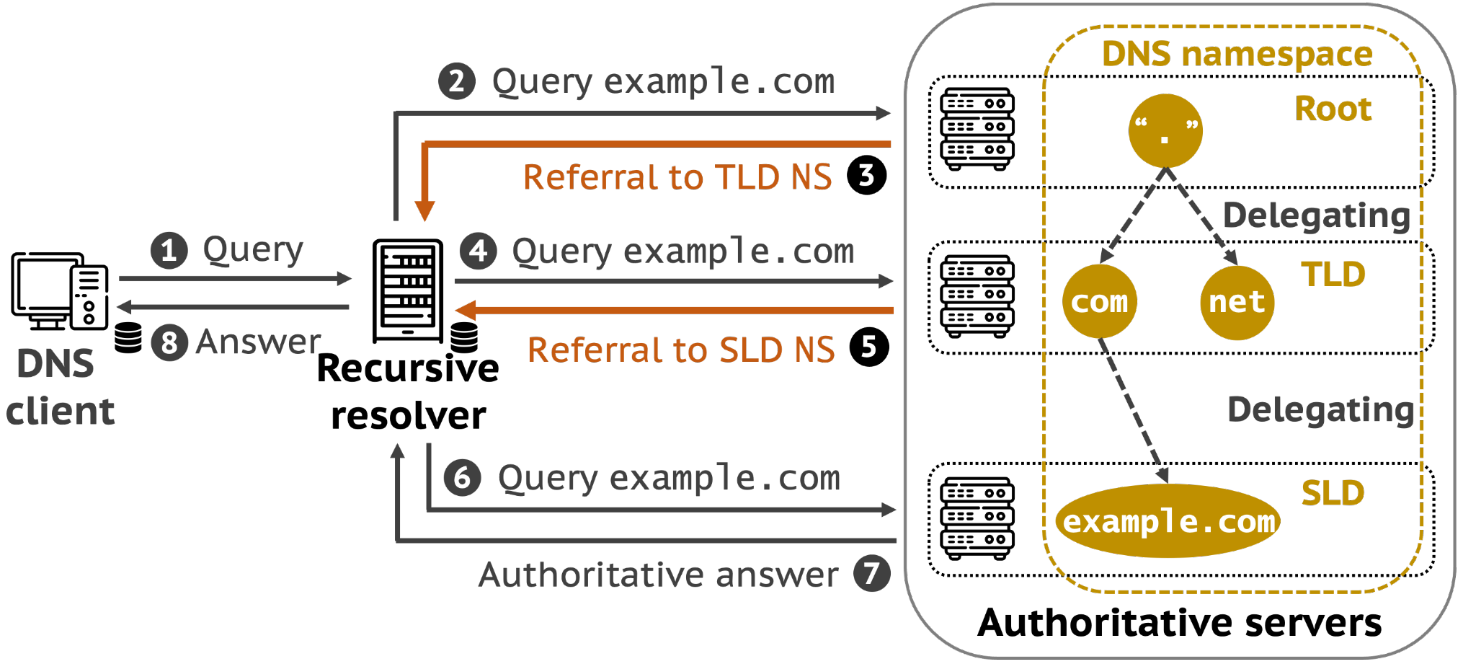
Domain Name System

- **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

DNS Resolution

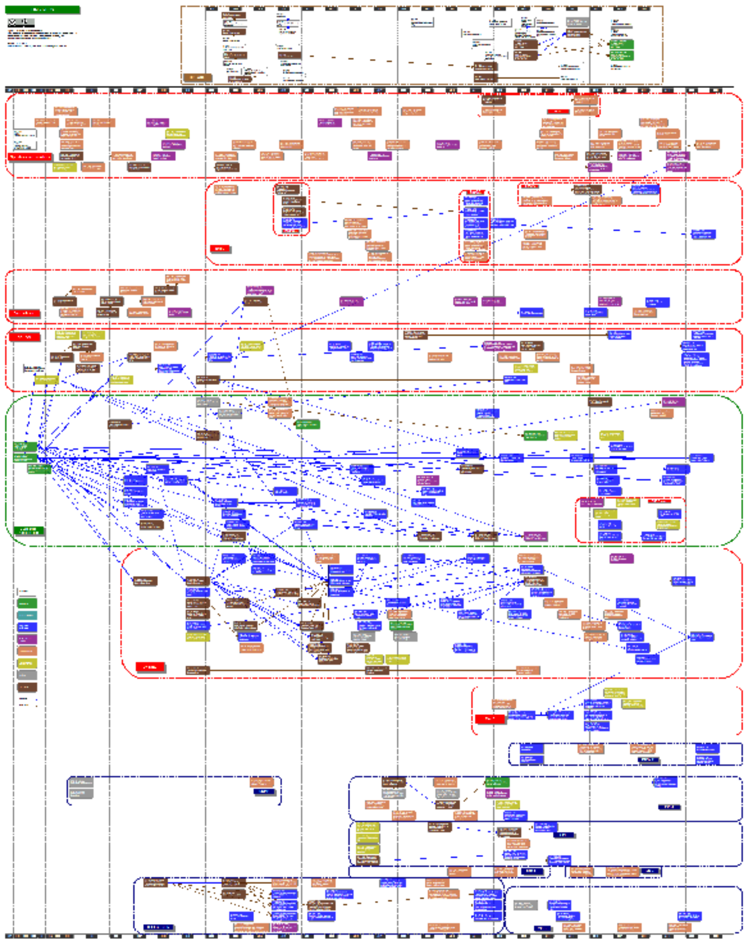
➢Recursive/Iterative process

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



DNS is complicated

- **Over 100 RFCs**
- **Many use cases**
  - Web browsing, email, zero-trust network, autonomous vehicle (!), etc.
- **Many implementations**
  - 20+ widely used software
- **Fragmented service ecosystem**
  - Millions of nameservers, open resolvers, local resolvers, and forwarders [1]



DNS RFCs (as of 2020)

[1] Mark Allman. Comments On DNS Robustness. IMC'18

DNS Failures & Attacks Happened a Lot

Help Net Security
October 26, 2021

Share

72% of organizations hit by DNS attacks in the past year

Unpatched DNS Bug Puts Millions of Routers, IoT Devices at Risk

MASQUERADE PARTY —

DNS cache poisoning, the Internet attack from 2008, is back from the dead

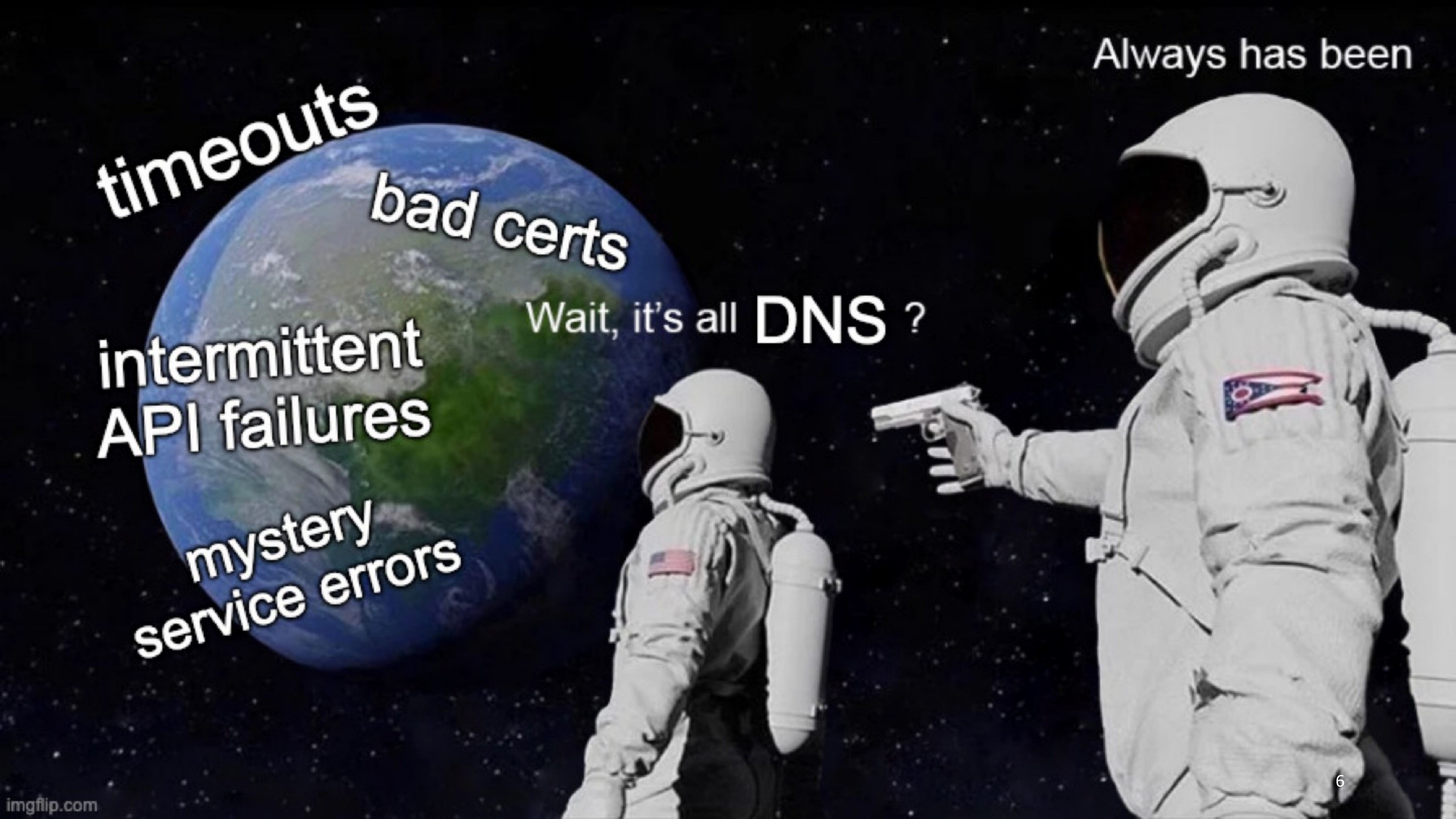
A newly found side channel in a widely used protocol lets attackers spoof domains.

DAN GOODIN - 11/12/2020, 6:30 AM

Facebook outage was a series of unfortunate events

A badly written command, a buggy audit tool, a DNS system that hobbled efforts to restore the network, and tight data-center security all contributed to Facebook's seven-hour Dumpster fire.

By Tim Greene
Executive Editor, Network World | OCT 5, 2021 6:25 PM PDT


timeouts
bad certs
intermittent API failures
mystery service errors
Wait, it's all DNS ?
Always has been
imgflip.com

Fuzzing in a Nutshell

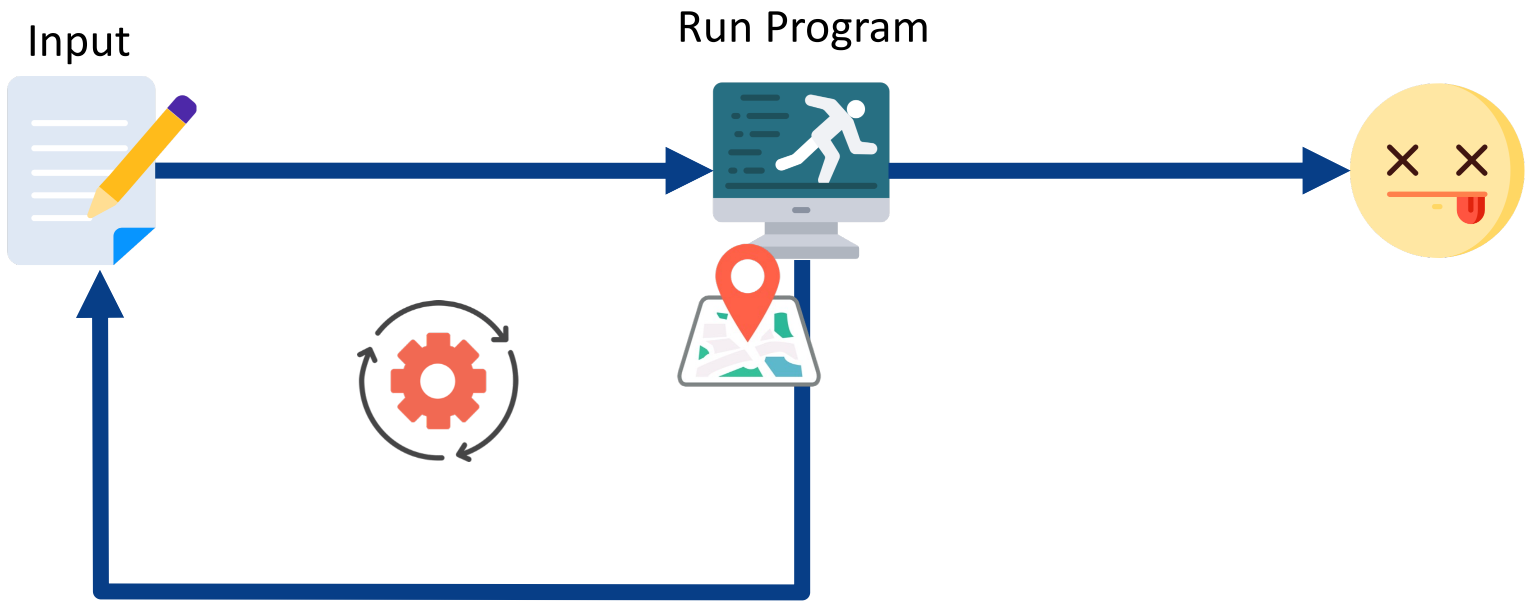
```
$ ./testme --help
Usage: testme <int32_arg>
```

```
$ ./testme --help
Usage: testme <int32_arg>
```

```
$ cat fuzzer.sh
while :
do
  input="$(dd if=/dev/urandom bs=4 count=1)"
  ./testme $input || echo $input >> crash_seeds
done
```

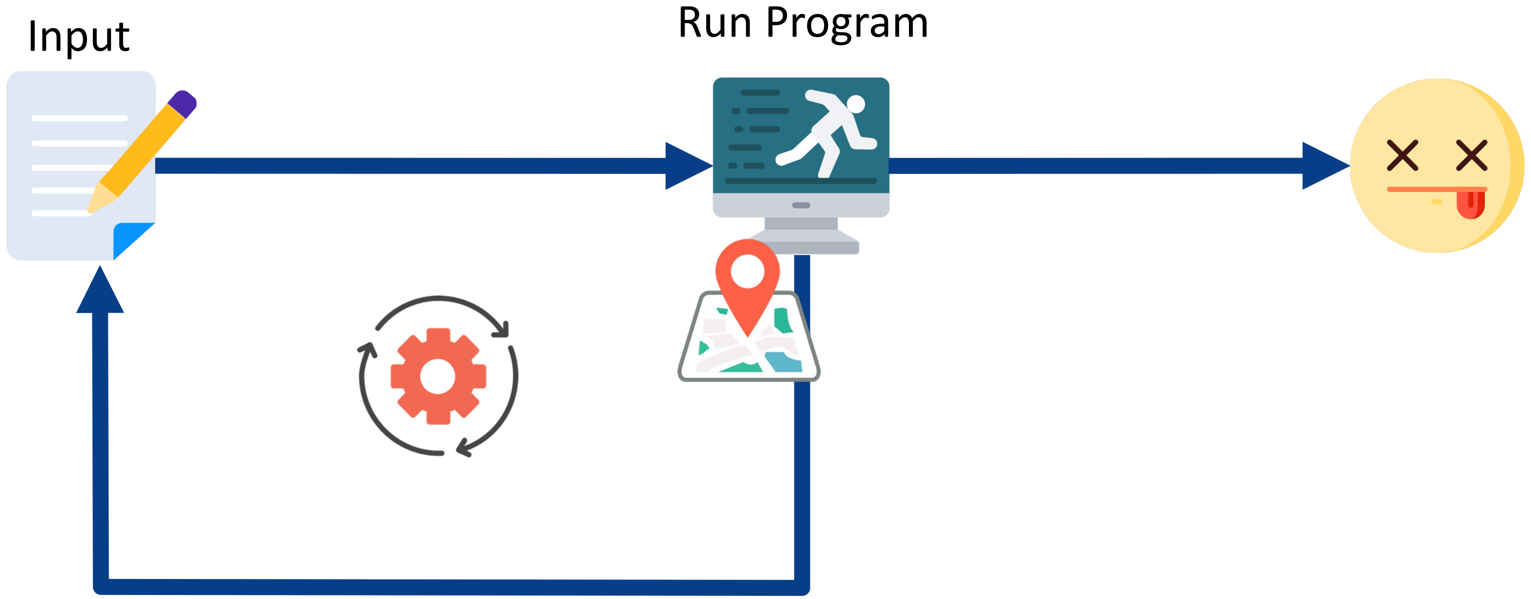
Fuzzing: Automated (Fuzz) Testing

- Coverage-based greybox fuzzing, e.g., AFL



What are the challenges for ResolverFuzz?

DNS Fuzzing: Challenge 1



DNS Bugs:

+ Cache poisoning
+ Denial-of-service
+ Access violation

Not always crash!

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

- **Manual analysis of 423 DNS CVEs from 1999-2023**
  - *291* CVEs about 6 DNS software
    - *245* CVEs about DNS resolvers
    - 109 CVEs don't trigger any crash!
    - *93* crash CVEs are non-memory (e.g., assertion failures)

Software*	# CVE							
	Non-crash				Crash			Total
	Cache Poisoning	Resource Consum.[1]	Others[2]	Total	Non-memory	Memory	Total	
BIND	18	18	11	47	75	22	97	144
Unbound	4	5	4	13	5	8	13	26
Knot Resolver	6	4	0	10	2	0	2	12
PowerDNS Recursor	13	8	9	30	7	6	13	43
MaraDNS	2	3	0	5	4	7	11	16
Technitium	3	1	0	4	0	0	0	4
Total	46	39	24	109	93	43	136	245

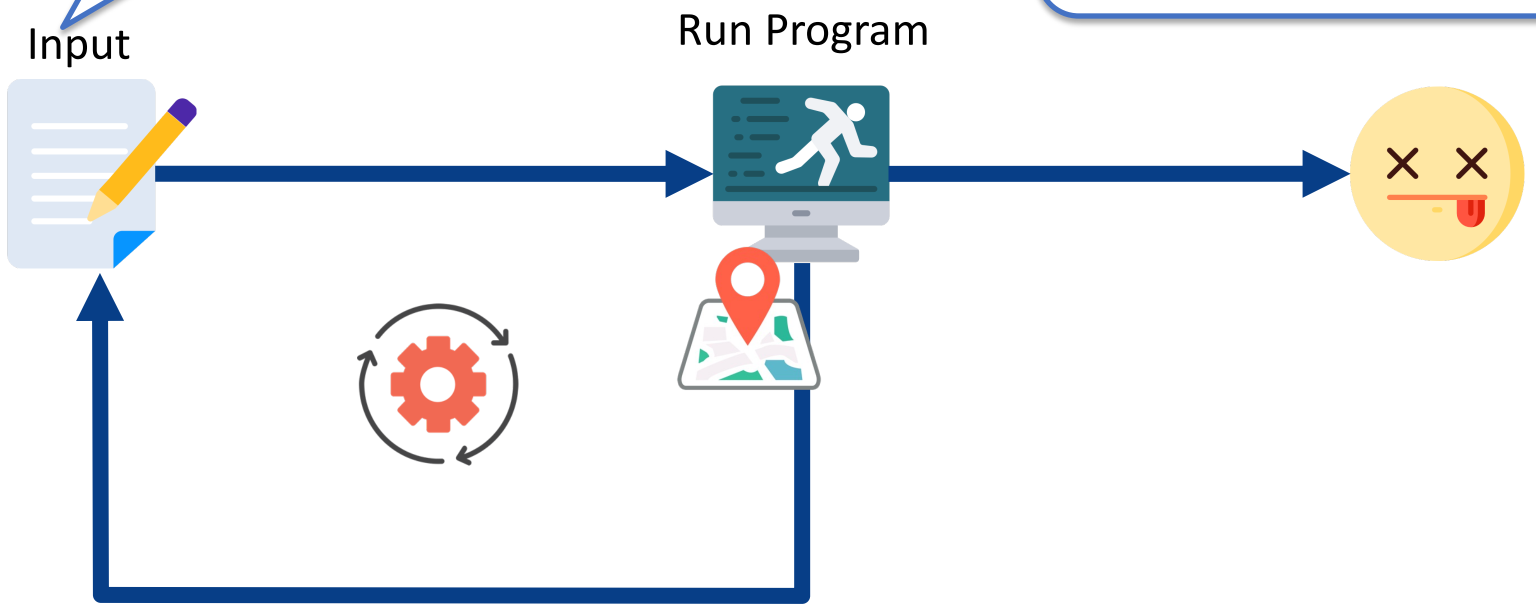
DNS Fuzzing: Challenge 2

Standard fuzzing:

+ Stateless (1 input per round)

DNS:

+ Stateful at resolver
+ Multi-party (client, resolver, name server)

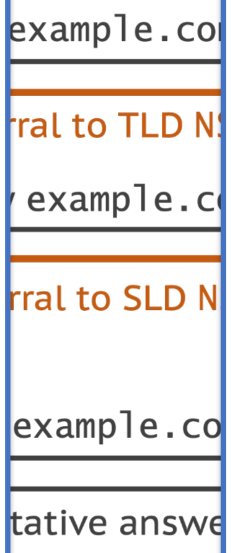


Stateless Fuzzing vs Stateful Resolver

Response without query

CVE-2021-25220:

+ Bogus NS response
+ Cache poisoning

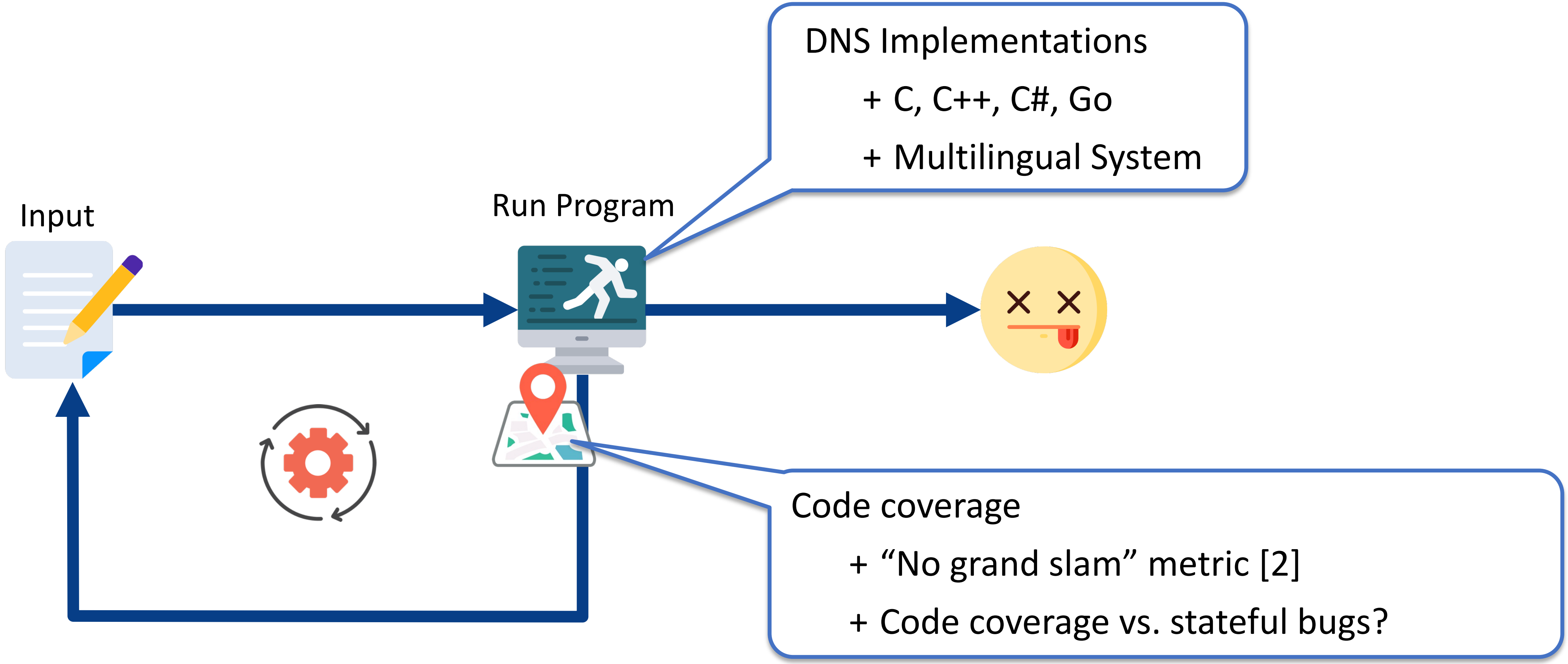


Query without response

CVE-2022-3924:

+ Many recursive queries
+ Stale option enabled
+ Race condition & crash

DNS Fuzzing: Challenge 3



[2] Wang et al. Be Sensitive and Collaborative:Analyzing Impact of Coverage Metrics in Greybox Fuzzing. RAID’19

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

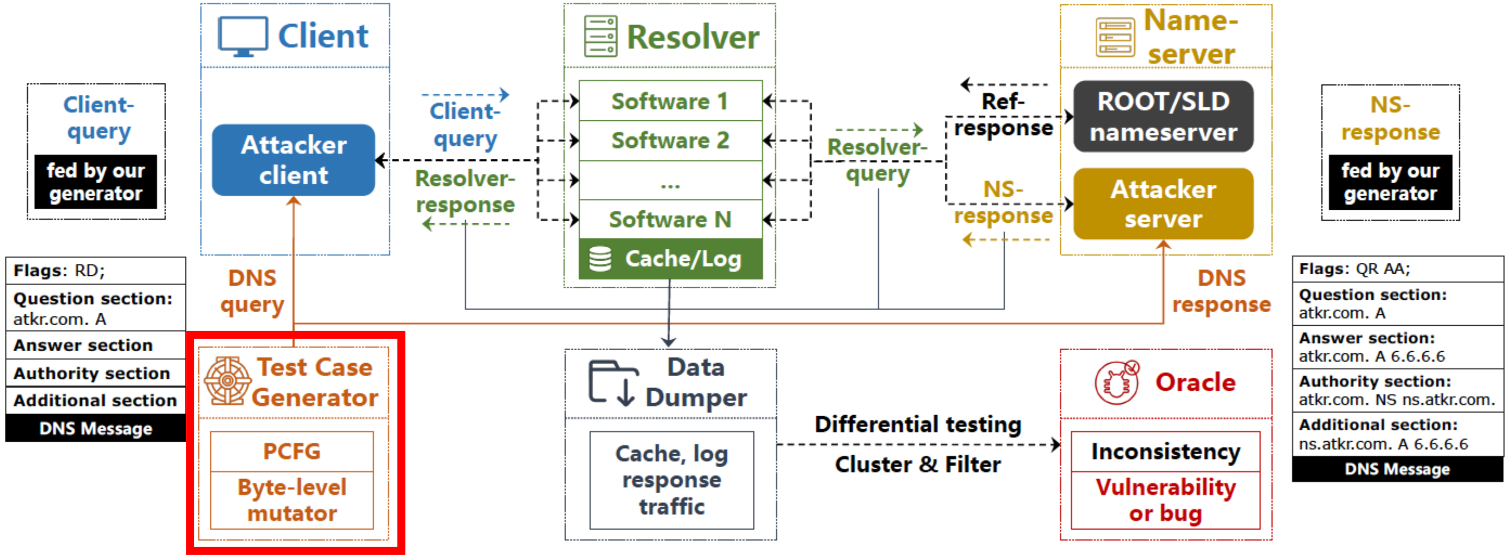


Figure 3: Workflow of RESOLVERFUZZ.

ResolverFuzz Infrastructure

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

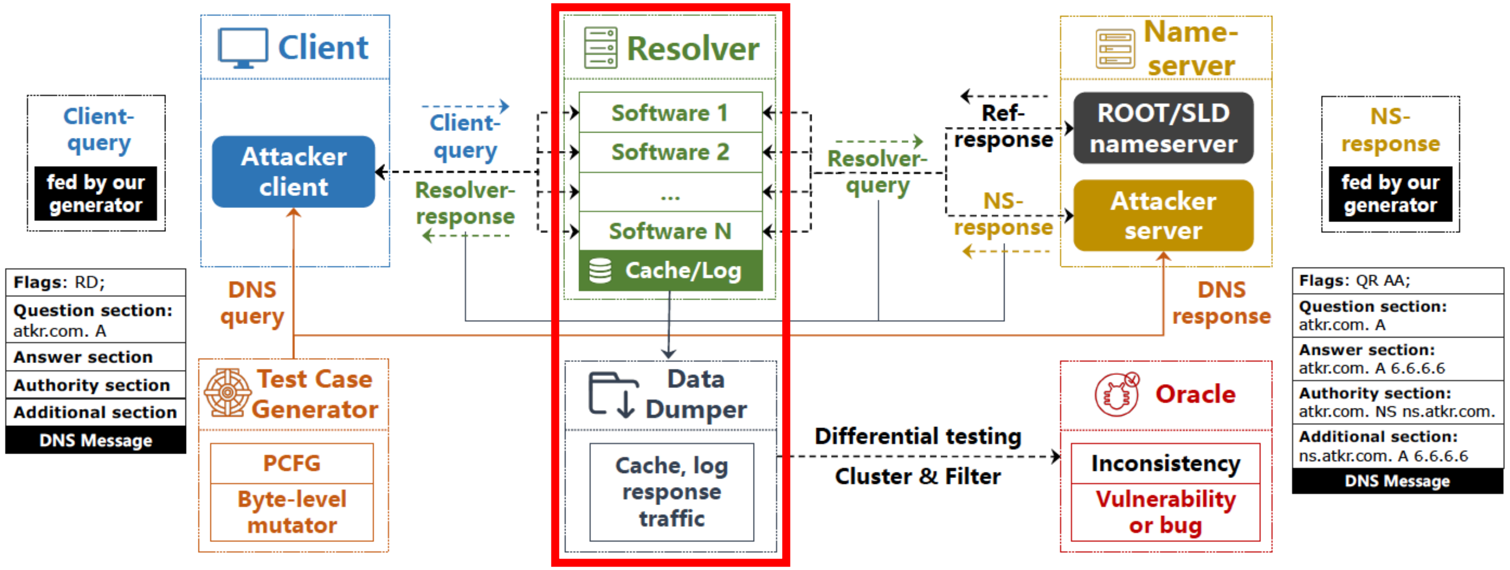


Figure 3: Workflow of RESOLVERFUZZ.

ResolverFuzz Infrastructure

➢Oracle:

- ➢Measure divergence
- ➢Bug/vuln. analysis

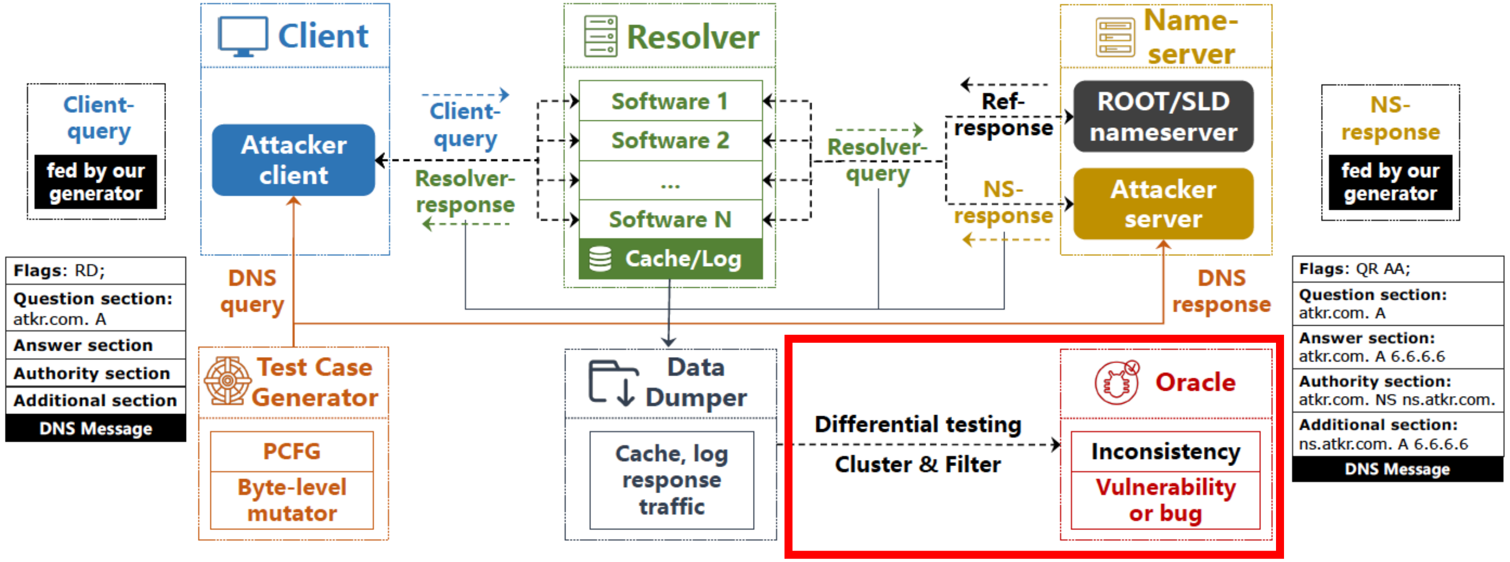
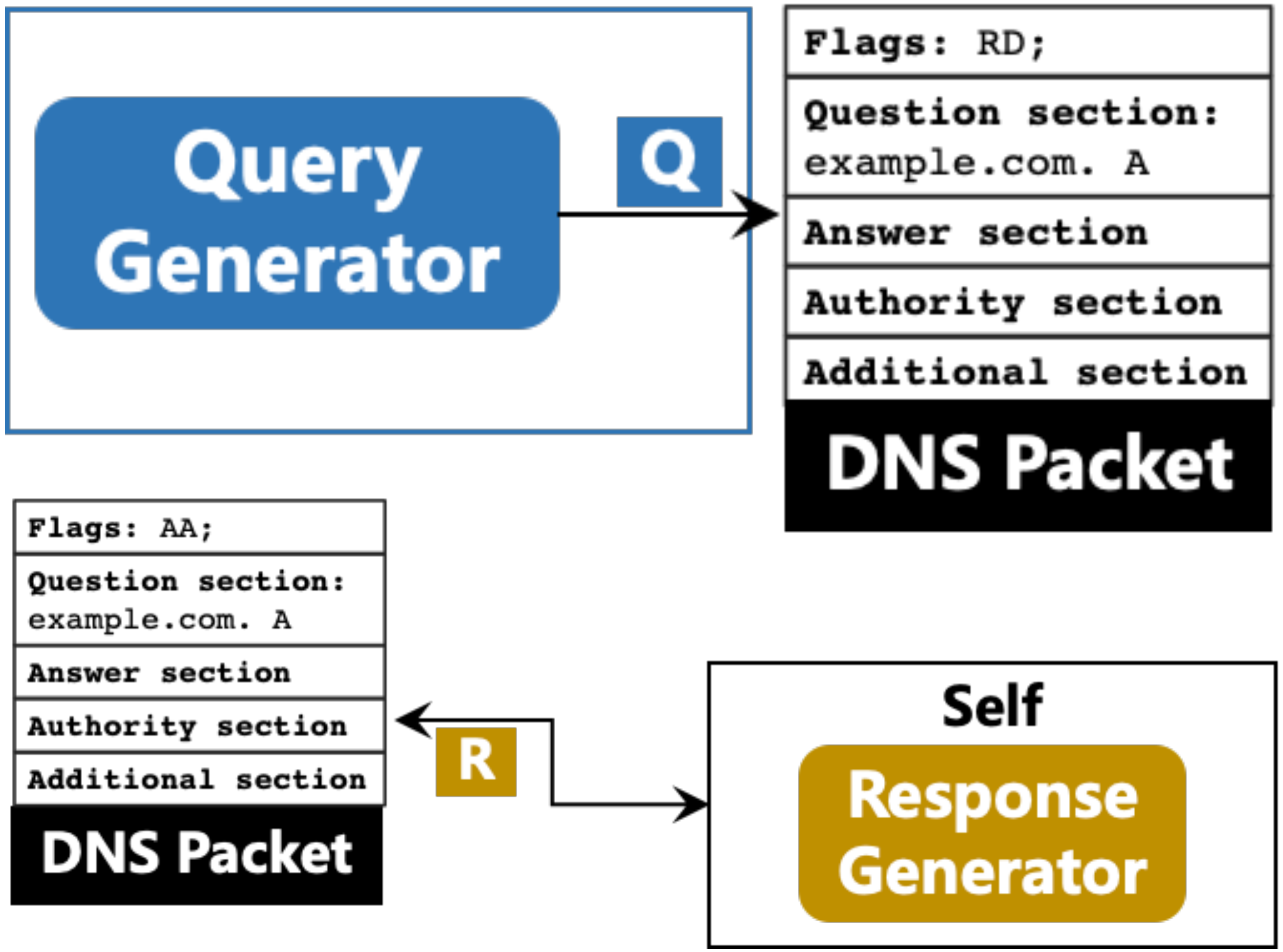


Figure 3: Workflow of RESOLVERFUZZ.

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

ResolverFuzz: Workflow

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

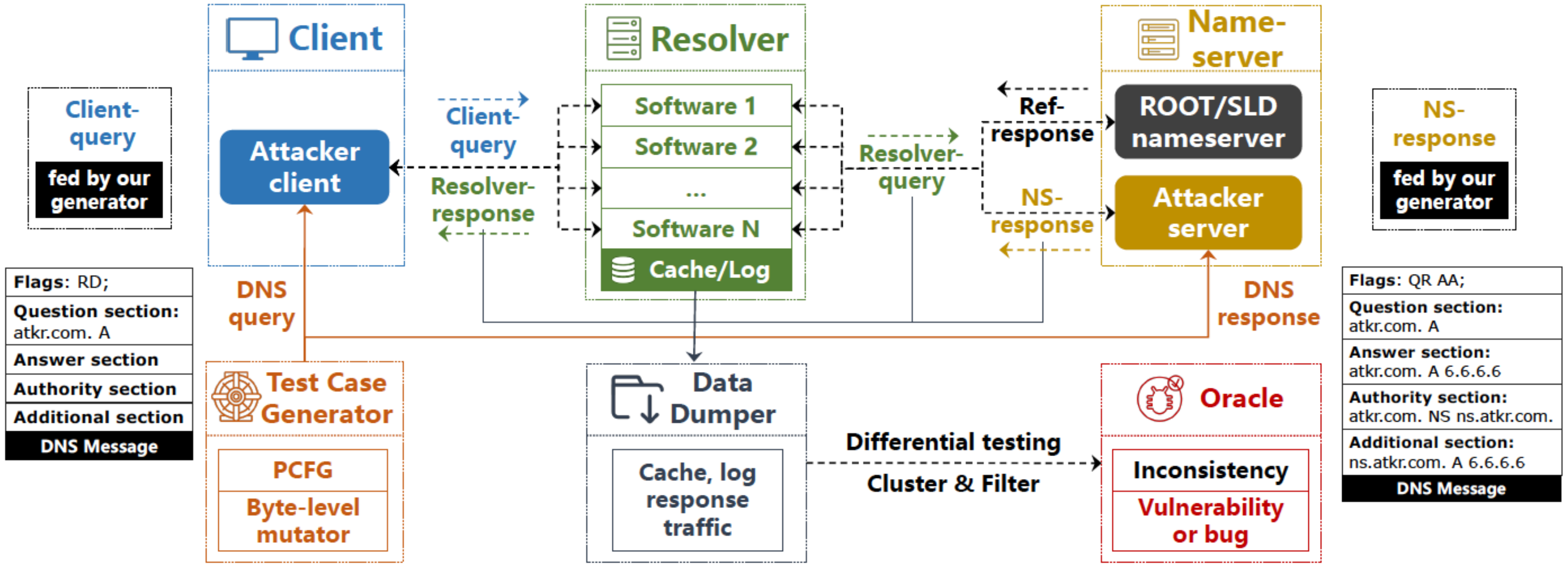


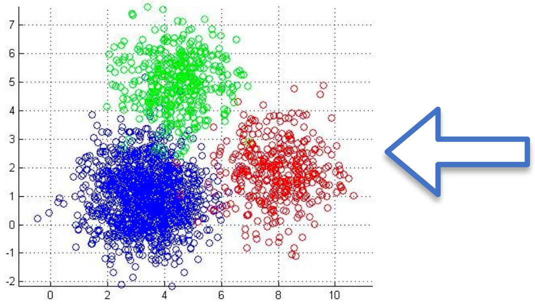
Figure 3: Workflow of ResolverFuzz.

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**"

Oracle

- **Different DNS software**
  - Objects of differential analysis
- **Three Oracles**
  - Cache poisoning oracle
  - Resource consumption oracle
  - Crash & Corruption oracle



DNS Software cache records

Bisecting K-means

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)

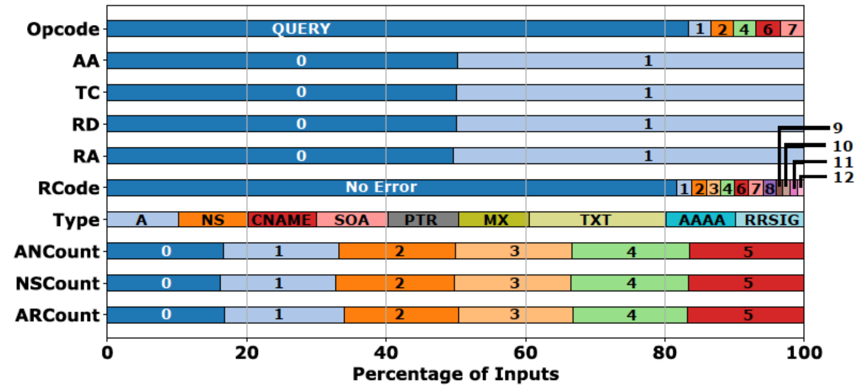
```
options {
	recursion yes;
}
// create a forward zone for test-cdns.example.com
zone "test-cdns.example.com" {
	type forward;
	forwarders { x.x.x.x port 53; };
	forward first; // fallback mode enabled
}
```

(d)

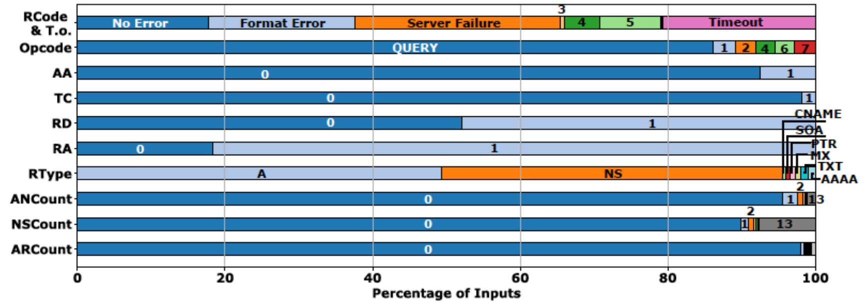
Figure 11: Example BIND configs of a) recursive-only, b) forward-only, c) CDNS without fallback, and d) CDNS with fallback.

Evaluation

- Analysis of test 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

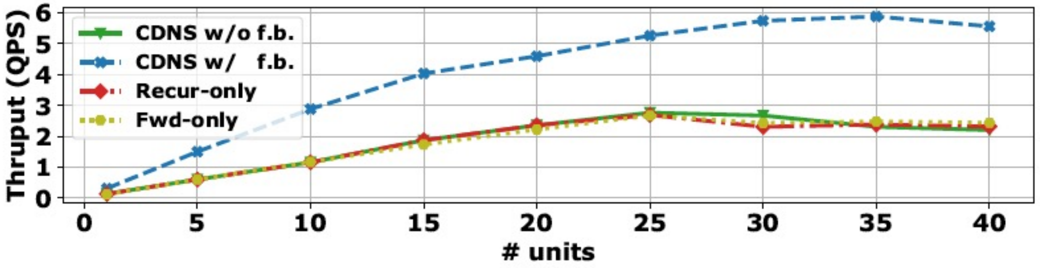


Figure 7: Throughput ("*Thruput*") of 4 modes with regard to the number of units. *CDNS w/o f.b.*, *CDNS w/ f.b.*, *Recur-only* and *Fwd-only* refers to *CDNS without fallback*, *CDNS with fallback*, *Recursive-only*, and *Forwarder-only*.

How many new vuln. are discovered?

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

Discovered Vulnerabilities

MaginotDNS [Security’23]

Phoenix Domain [NDSS’23, OARC 39]

TuDoor [S&P’24, OARC’42]

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	✓†	✗	✓ (red)	✓	3	✗	✗	✗	✗	✗	✗	✗	0	✓	4
Unbound	✗	✗	✓ (red)	✓†	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 ✓ (red): Vulnerable. ✓ (red): Discussed but no immediate action. ✓: Confirmed and/or fixed by vendors. ✗: Not vulnerable. †: CVEs assigned. ‘-’: Not applicable.

\# Amount of test cases: *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).

Thanks for listening!

Any questions?

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