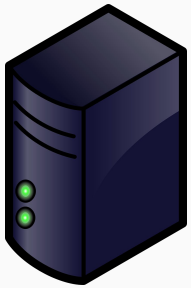
Seven fun ways to DoS yourself with DNS

SSAC | ICANN 80 Kigali

The DNS seems simple enough

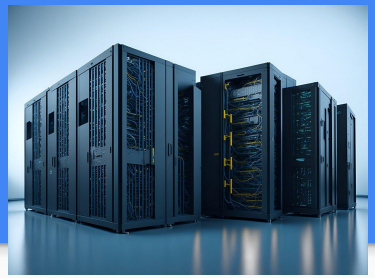
- You send a question, where is www.gmail.com?
- It sends an answer, 142.250.65.197.
- But there's a lot under the hood
- We keep adding features to the DNS, many of which don't fit well together ...
- The DNS Camel

DNS works by repeated delegation



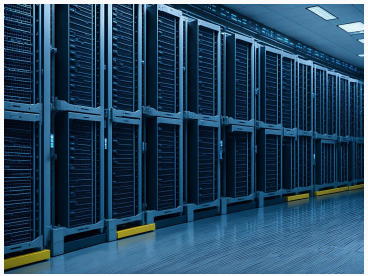
Where is www.gmail.com?

”ask .com”



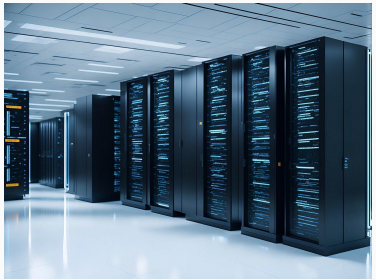
Root server

”ask google.com”



.com server

”it’s 142.250.65.197

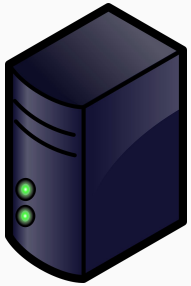


google.com server

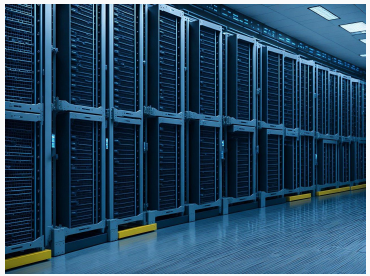
Three ways to fail

- Overload
  - Queries or responses that take a long time to process
  - Or cause a flood of other queries
- Crash
  - Queries or responses that find fatal bugs
- Cache Poison
  - Responses with fake data
  - Cache then returns them to queries

A lot of caches and indirection



.com server



”ask google.com”

Where is www.gmail.com?

```
;; AUTHORITY SECTION:
gmail.com. 172800      IN      NS      ns2.google.com.
gmail.com. 172800      IN      NS      ns1.google.com.
gmail.com. 172800      IN      NS      ns3.google.com.
gmail.com. 172800      IN      NS      ns4.google.com.
CK0...EFN8430Q.com. 86400 IN NSEC3 ...
CK0...EFN8430Q.com. 86400 IN IN RRSIG NSEC3 13 2 8...
```

```
;; QUESTION SECTION:
;ns1.google.com. IN A

;; ANSWER SECTION:
ns1.google.com. 41705 IN    A       216.239.32.10
```

How much indirection is too much?

- It depends

  a CNAME b
  b CNAME c
  c CNAME d
  d CNAME e
  e CNAME f
  f CNAME g
  g CNAME h
  ...

```
;; QUESTION SECTION:
;www.verizon.net.            IN     A

;; ANSWER SECTION:
www.verizon.net. 300 IN CNAME fp31C6.wpc.1A525D.xicdn.net.
fp31C6.wpc.1A525D.xicdn.net. 3600 IN CNAME fp31c6.wpc.xicdn.net.
fp31c6.wpc.xicdn.net.       3600 IN A 152.195.19.228
```

So limit the number of lookups

- For CNAMEs it’s not hard
- But some chains can be bushy

a.com NS n1.b.com
a.com NS n2.c.com

b.com NS n3.d.com
b.com NS n4.e.com

c.com NS n5.f.com
c.com NS n6.g.com

d.com NS n7.h.com
d.com NS n8.i.com
...

QNAME minimization attacks (QMIN)

- Lookups used to leak a lot of data
  - To root: hot-chix.bloggy.com
  - To .com: hot-chix.bloggy.com
  - To bloggy.com: hot-chix.bloggy.com
- Minimized queries
  - To root: com
  - To .com: bloggy.com
  - To bloggy.com: hot-chix.bloggy.com
- Each server sees only what it needs to

QNAME minimization attacks (QMIN)

- Clients can’t tell where the boundaries are
- Naive clients amplify long names to many queries

a.b.c.d.e.f.g.h.i.j.k.com
k.com
j.k.com
i.j.k.com
h.i.j.k.com
g.h.i.j.k.com
f.g.h.i.j.k.com
e.f.g.h.i.j.k.com
d.e.f.g.h.i.j.k.com
c.d.e.f.g.h.i.j.k.com
b.c.d.e.f.g.h.i.j.k.com
a.b.c.d.e.f.g.h.i.j.k.com

CAMP Compositional Amplification attacks

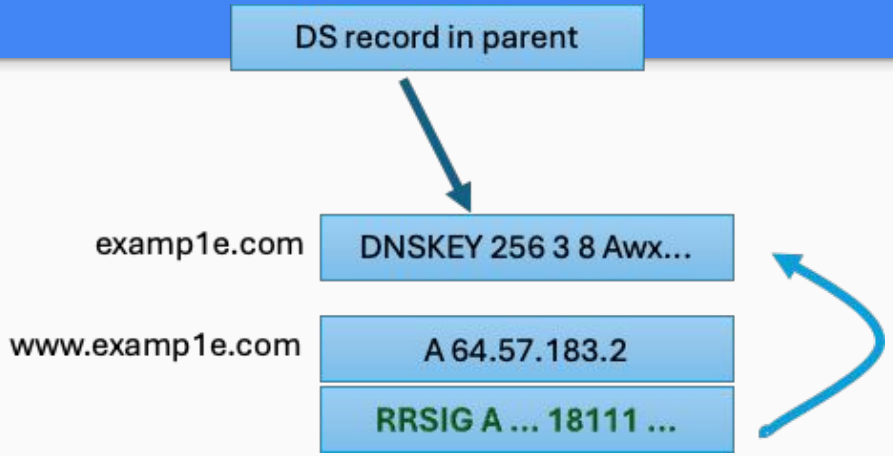
- Combine NS, QMIN, and CNAME chains
- Recent paper has created 700 : 1 amplification or more

Use botnets for random name attacks

- Random names aren't in the cache
- Lots of queries provoke lots of traffic
- If asking for DNSSEC, answers can be large

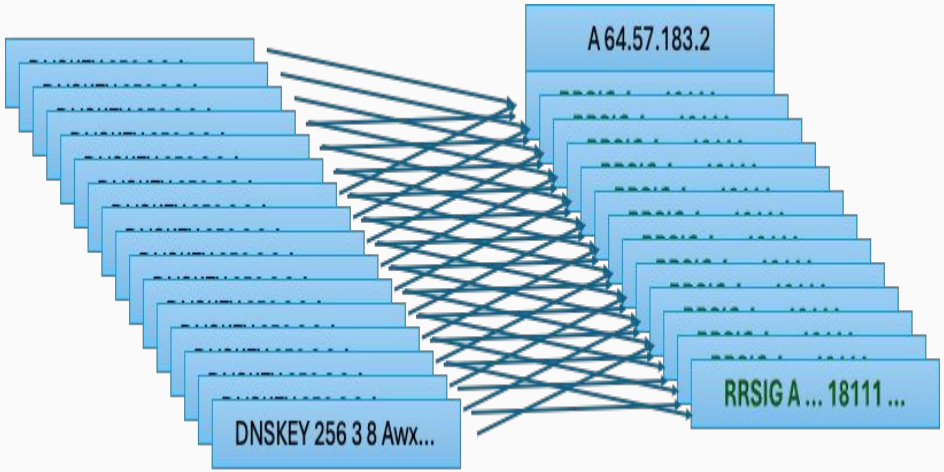
fake1.domain.com A ?
fake2.domain.com A ?
fake3.domain.com A ?
fake4.domain.com A ?
fake5.domain.com A ?
...

Keytrap: too many DNS signatures



- DNSKEY validates RRSIGs
- Can be multiple DNSKEY and RRSIG
- Separated by ID, but easy to duplicate IDs

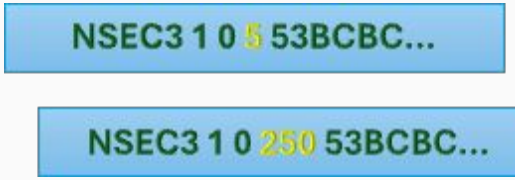
Keytrap: too many DNS signatures



- 100 DNSKEY with ID 18111
- 100 RRSIG with ID 18111
- 100x100 = 10,000 validations
- Validation done in main thread of DNS cache, can hang it for minutes
- Oops

NSEC3 closest encloser

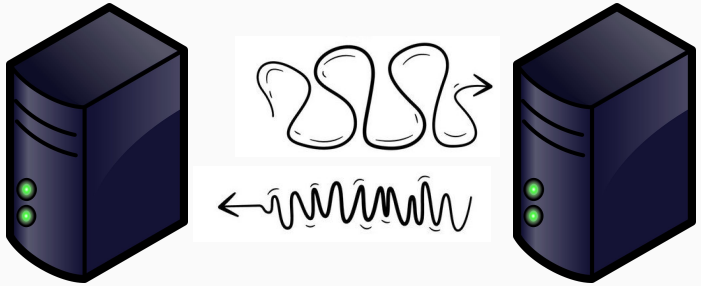
- NSEC3 uses name hashes
- To find closest name, client checks each possible name individually
- Long names: many queries
- Slow if NSEC3 uses multiple hash (deprecated but still allowed)



aa.bb.cc.dd.ee.ff.gg.hh.ii.jj.kk.ll.mm.examp1e.com
bb.cc.dd.ee.ff.gg.hh.ii.jj.kk.ll.mm.examp1e.com
cc.dd.ee.ff.gg.hh.ii.jj.kk.ll.mm.examp1e.com
dd.ee.ff.gg.hh.ii.jj.kk.ll.mm.examp1e.com
ee.ff.gg.hh.ii.jj.kk.ll.mm.examp1e.com
ff.gg.hh.ii.jj.kk.ll.mm.examp1e.com
...
examp1e.com

Fuzzing

- Send slightly randomized queries or answers, look for unexpected results
- ResolverFuzz did 700,000 queries to six DNS packages over three days
- Found a lot of bugs
  - 13 cache poison
  - 9 overload
  - 1 crash



What to do about all this

- Is there a better or more general way to collect and limit all the indirect work for a request?
  - Bushy chains of requests
  - Random compute bombs
  - What else?
- Is it time to push back harder on “this is stupid”
  - Who needs 100 or even 10 keys or signatures with the same ID?
  - Who needs 100 or even 5 NSEC3 hashes?
  - Who needs 10 DNS servers in 10 TLDs, each of which have 10 servers in 10 … ?
  - etc.

Seven fun ways to DoS yourself with DNS

SSAC | ICANN 80 Kigali