APNIC 44

Re-Engineering the Root of the DNS

Geoff Huston,
Chief Scientist APNIC

TAICHUNG, TAIWAN
7–14 September 2017

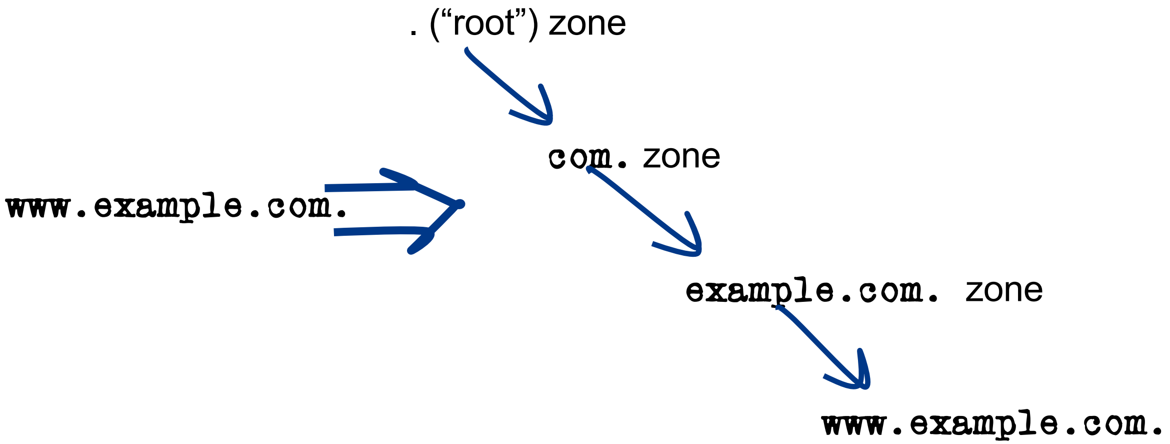
#apnic44

In this presentation

- I’ll talk about the DNS, and the root server infrastructure in particular
- And some recent initiative by APNIC to try and improve the situation

The Structure of the Domain *Name System*

The Domain Name System (DNS) is a distributed data collection using a delegation hierarchy that reflects the internal hierarchical structure of domain names. At each level in the name hierarchy each label represents a potential point of administrative delegation



. ("root") zone

`com.` zone

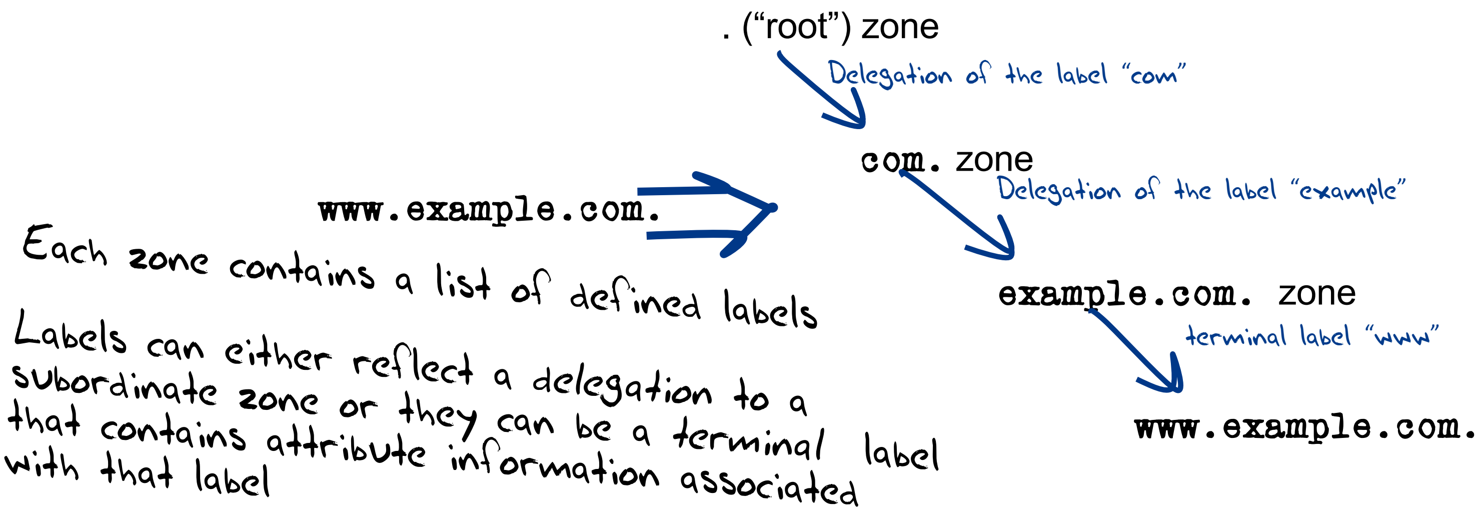
`www.example.com.`

`example.com.` zone

`www.example.com.`

The Structure of the Domain *Name System*

The Domain Name System (DNS) is a distributed data collection using a delegation hierarchy that reflects the internal hierarchical structure of domain names. At each level in the name hierarchy each label represents a potential point of administrative delegation



Each zone contains a list of defined labels

Labels can either reflect a delegation to a subordinate zone or they can be a terminal label that contains attribute information associated with that label

DNS Name Servers

- Every DNS zone has a set of authoritative servers that can answer queries for names defined by that zone
- The Root Zone is just another zone in that respect, and the authoritative servers for that zone are called "Root Servers"
  - There are 13 Root Server names
  - And these names are used to label Anycast Name Server constellations
  - Which means that there are probably some thousands of discrete Root Server instances if you could peek inside all of these these anycast clouds

Resolving a DNS Name

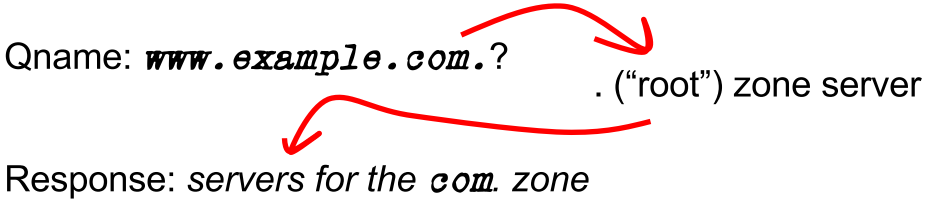
Your resolver needs need to ask a DNS server for the zone that contains the terminal label for the associated information (resource record) associated with the DNS name

But…

- Where exactly is the zone cut?
- Who are the servers?

So resolvers *discover* this information by performing a top-down iterative search…

Resolving a DNS Name

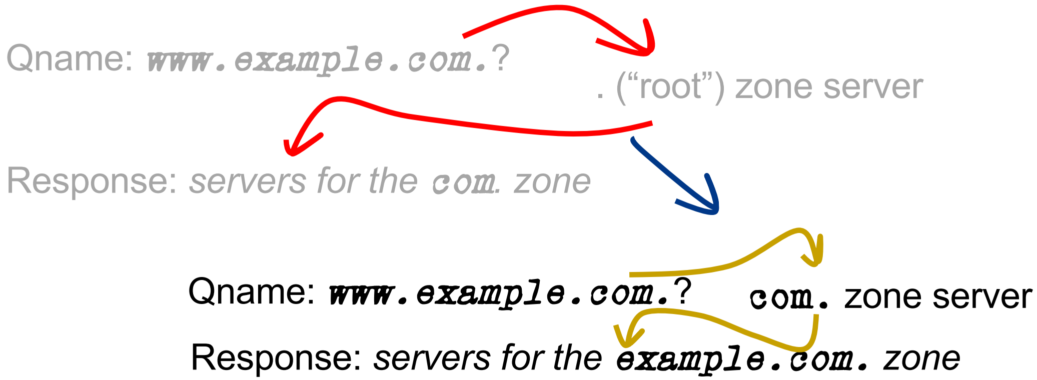


Qname: ***www.example.com.***?

. ("root") zone server

Response: *servers for the* ***com****. zone*

Resolving a DNS Name



Qname: `www.example.com.`?

. (“root”) zone server

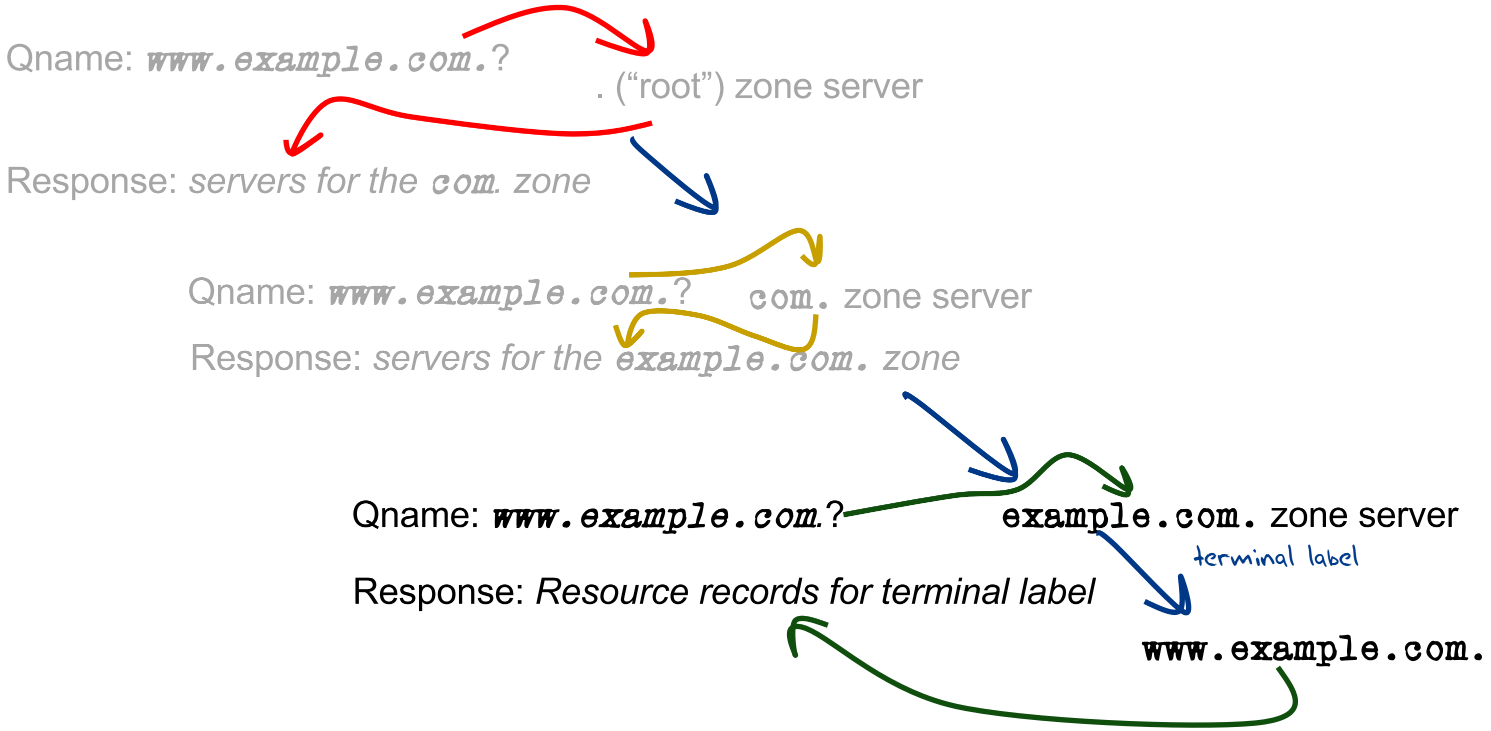
Response: *servers for the* `com`*. zone*

Qname: `www.example.com.`?

`com.` zone server

Response: *servers for the* `example.com.` *zone*

Resolving a DNS Name



Qname: `www.example.com.`?

. ("root") zone server

Response: *servers for the `com.` zone*

Qname: `www.example.com.`?

`com.` zone server

Response: *servers for the `example.com.` zone*

Qname: `www.example.com.`?

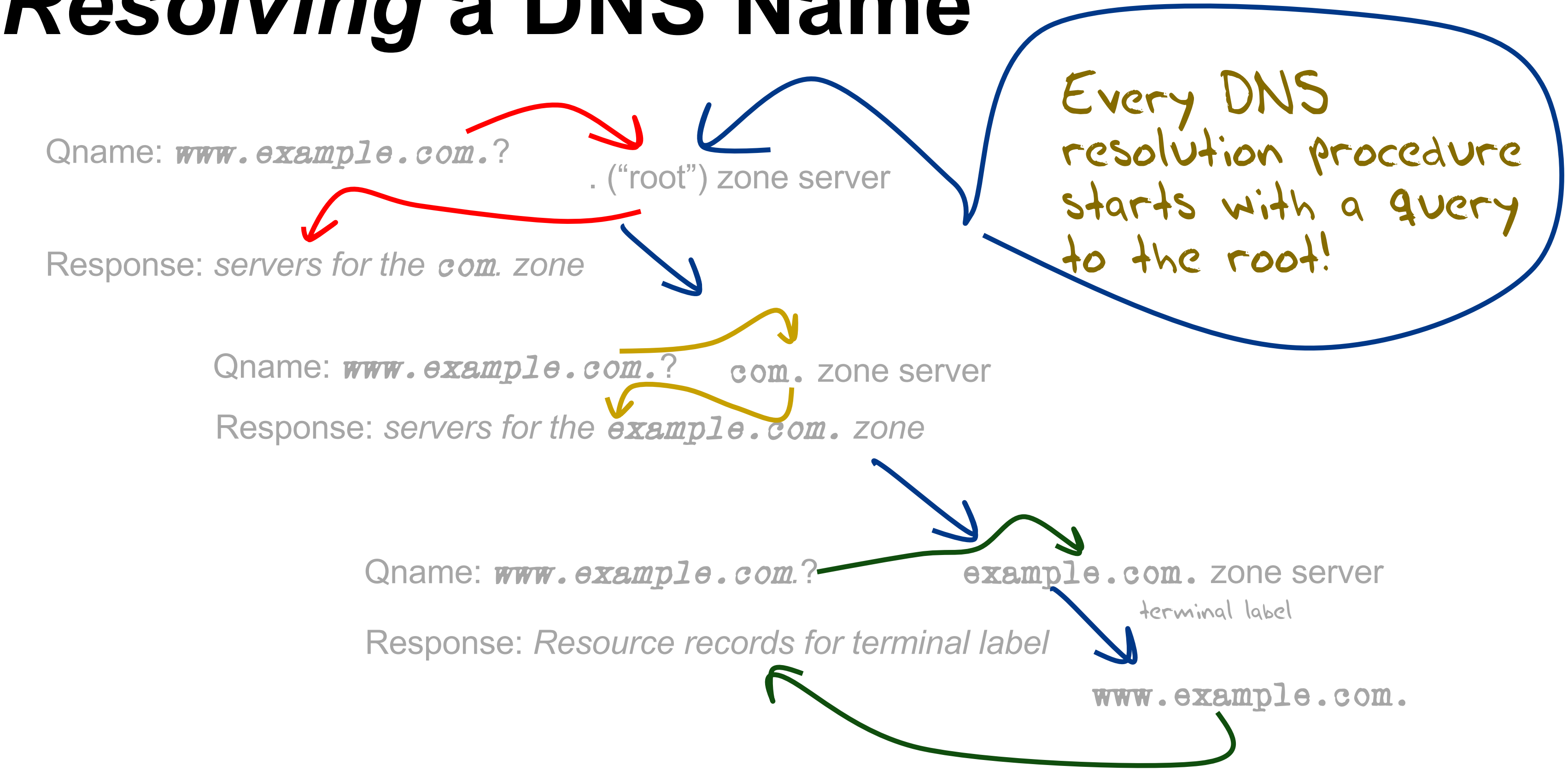
`example.com.` zone server

terminal label

Response: *Resource records for terminal label*

`www.example.com.`

Resolving a DNS Name



Qname: *`www.example.com.`*?

. ("root") zone server

Response: *servers for the `com`. zone*

Qname: *`www.example.com.`*?

`com.` zone server

Response: *servers for the `example.com.` zone*

Qname: *`www.example.com`*.?

`example.com.` zone server

terminal label

Response: *Resource records for terminal label*

`www.example.com.`

Every DNS resolution procedure starts with a query to the root!

How to be bad

Every DNS resolution procedure starts with a query to the root!

If an attacker could prevent the root servers from answering DNS queries then the entire Internet will suffer!

Caching in the DNS

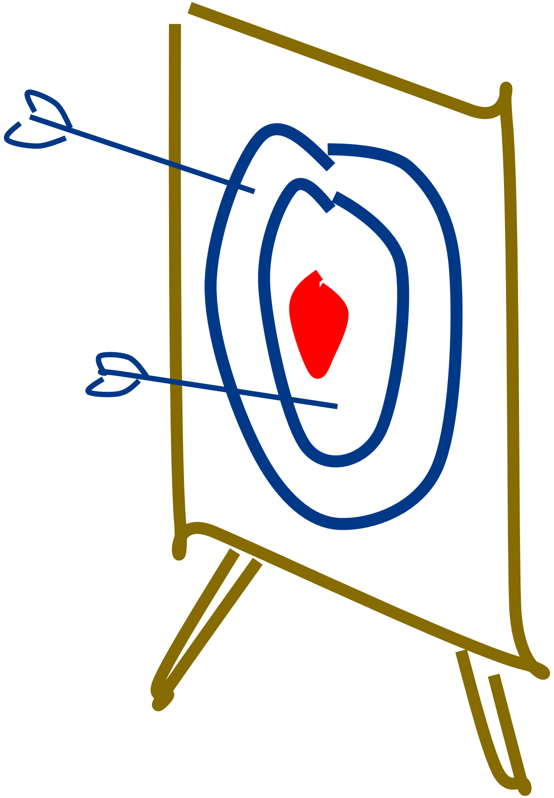
The main role of the root server system is to answer queries that are not cached in local name caches

The vast majority of the queries that are passed to the root zone servers (some 2/3 of root queries) generate a “no-such-name” (NXDOMAIN) response from the root system

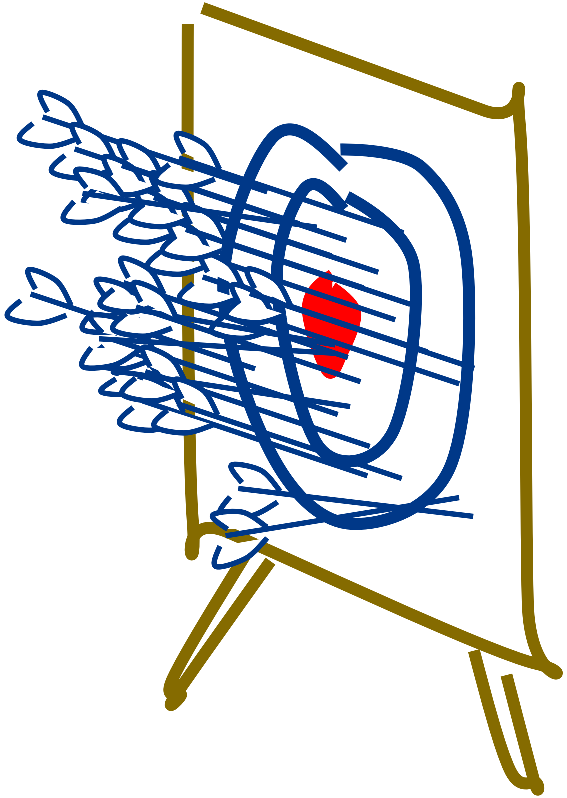
How to be bad

To attack the root servers you need to get past DNS resolver caches.

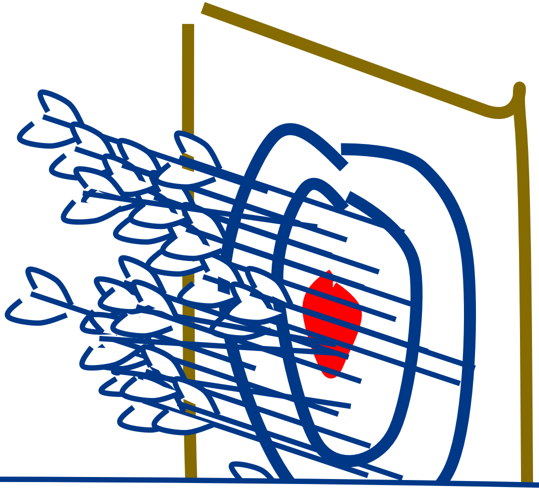
This means you need to have every query in the DNS attack flow ask for a different non-existent name



Root Servers are a highly visible attack target



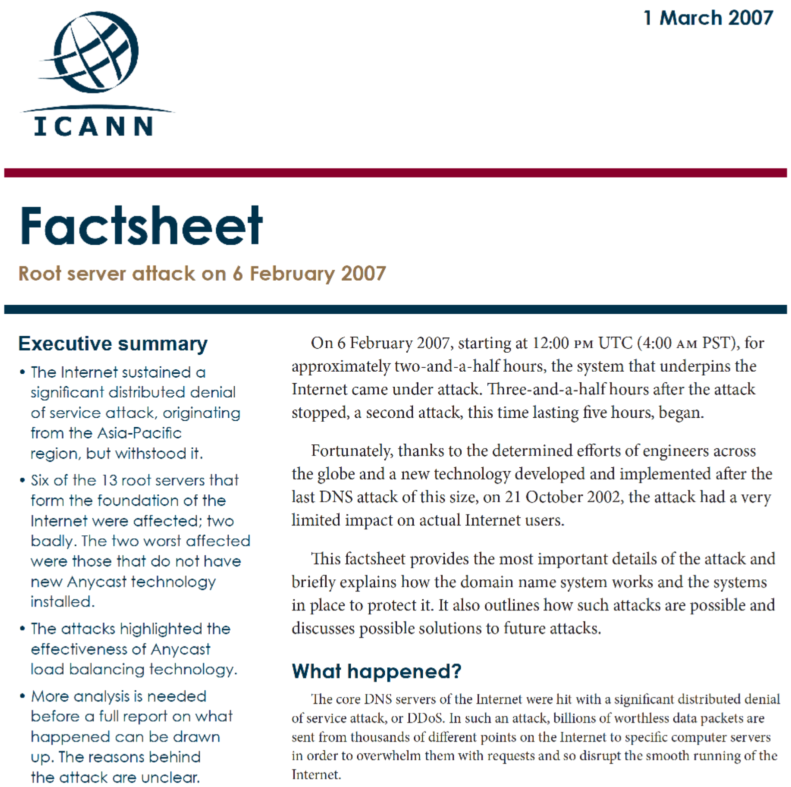
Root Servers are a highly visible attack target



Root Servers are a highly visible attack target

if you can prevent resolvers from getting answers from the root then the resolvers will stop answering queries as their local cache expires

If you can
root then
queries as



1 March 2007

ICANN

Factsheet

Root server attack on 6 February 2007

Executive summary

- The Internet sustained a significant distributed denial of service attack, originating from the Asia-Pacific region, but withstood it.
- Six of the 13 root servers that form the foundation of the Internet were affected; two badly. The two worst affected were those that do not have new Anycast technology installed.
- The attacks highlighted the effectiveness of Anycast load balancing technology.
- More analysis is needed before a full report on what happened can be drawn up. The reasons behind the attack are unclear.

On 6 February 2007, starting at 12:00 PM UTC (4:00 AM PST), for approximately two-and-a-half hours, the system that underpins the Internet came under attack. Three-and-a-half hours after the attack stopped, a second attack, this time lasting five hours, began.

Fortunately, thanks to the determined efforts of engineers across the globe and a new technology developed and implemented after the last DNS attack of this size, on 21 October 2002, the attack had a very limited impact on actual Internet users.

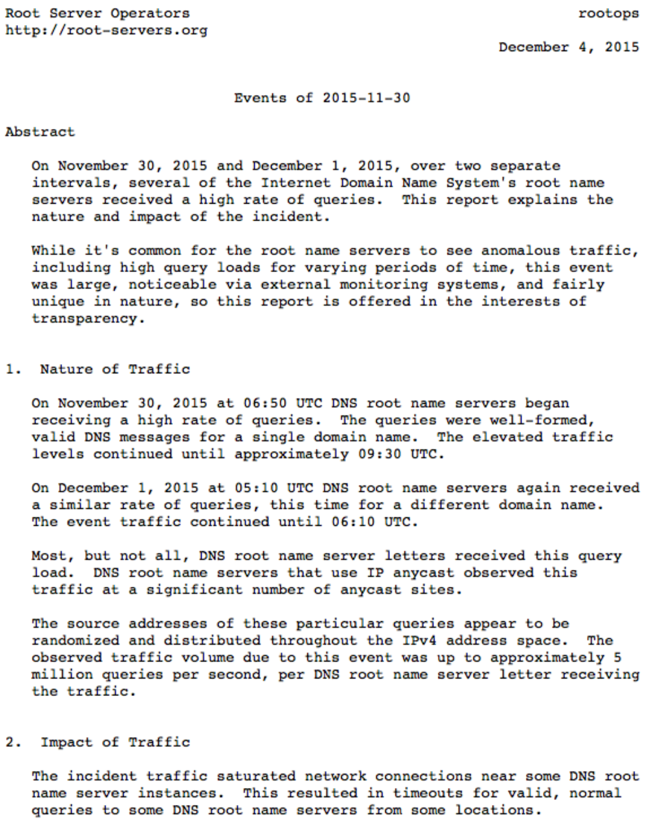
This factsheet provides the most important details of the attack and briefly explains how the domain name system works and the systems in place to protect it. It also outlines how such attacks are possible and discusses possible solutions to future attacks.

What happened?

The core DNS servers of the Internet were hit with a significant distributed denial of service attack, or DDoS. In such an attack, billions of worthless data packets are sent from thousands of different points on the Internet to specific computer servers in order to overwhelm them with requests and so disrupt the smooth running of the Internet.

Servers are a highly
attack target

if
roo
que



Root Server Operators   rootops
http://root-servers.org
                                                    December 4, 2015

Events of 2015-11-30

Abstract

On November 30, 2015 and December 1, 2015, over two separate intervals, several of the Internet Domain Name System's root name servers received a high rate of queries. This report explains the nature and impact of the incident.

While it's common for the root name servers to see anomalous traffic, including high query loads for varying periods of time, this event was large, noticeable via external monitoring systems, and fairly unique in nature, so this report is offered in the interests of transparency.

1. Nature of Traffic

On November 30, 2015 at 06:50 UTC DNS root name servers began receiving a high rate of queries. The queries were well-formed, valid DNS messages for a single domain name. The elevated traffic levels continued until approximately 09:30 UTC.

On December 1, 2015 at 05:10 UTC DNS root name servers again received a similar rate of queries, this time for a different domain name. The event traffic continued until 06:10 UTC.

Most, but not all, DNS root name server letters received this query load. DNS root name servers that use IP anycast observed this traffic at a significant number of anycast sites.

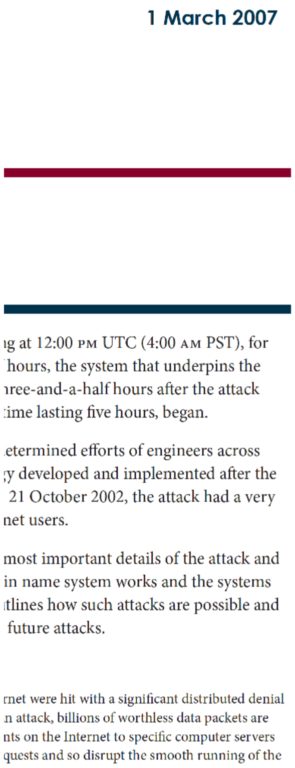
The source addresses of these particular queries appear to be randomized and distributed throughout the IPv4 address space. The observed traffic volume due to this event was up to approximately 5 million queries per second, per DNS root name server letter receiving the traffic.

2. Impact of Traffic

The incident traffic saturated network connections near some DNS root name server instances. This resulted in timeouts for valid, normal queries to some DNS root name servers from some locations.

the attack are unclear.

Internet.



1 March 2007

ıg at 12:00 PM UTC (4:00 AM PST), for
hours, the system that underpins the
ıree-and-a-half hours after the attack
ime lasting five hours, began.

etermined efforts of engineers across
y developed and implemented after the
21 October 2002, the attack had a very
net users.

most important details of the attack and
in name system works and the systems
tlines how such attacks are possible and
future attacks.

rnet were hit with a significant distributed denial
n attack, billions of worthless data packets are
nts on the Internet to specific computer servers
quests and so disrupt the smooth running of the

Servers are a highly attack target

How should we defend the Root?

- Larger Root Server platforms?
- More Root Server Letters?
- More Anycast Instances?
- Change Root Server response behaviours?
- Or…

How should we defend the Root?

- Larger R Can't scale * platforms?
- More Root Server Letters?
- More Anycast Instances?
- Change Root Server response behaviours?
- Or…

* Distributed parallel attacks can scale up in intensity more effectively than a single point of service can scale its defence mechanisms

How should we defend the Root?

- Larger R Can't scale platforms?
- Mor Err, umm – well no * s?
- More Anycast Instances?
- Change Root Server response behaviours?
- Or…

* The limit of 13 distinct root server names is an inherited limit that these days has a political dimension that has largely supersedes the original technical reasons for the limit. in any case more letters is not a very good DDOS defence!

How should we defend the Root?

- Larger R ver platforms? Can't scale
- Mor s? Err, umm – well no *
- More Anycast Instances? Today's practice for root servers
- Change Root Server response behaviours?
- Or…

Anycast Root Servers

12 of the 13 root server “letters” operate some form of “anycast” server constellation. All the servers in a constellation respond to the same public IP addresses. The routing system will direct resolvers to pass their query to a particular root letter to the “closest” member of the letter’s anycast constellation.

Anycast provides:

- faster responses to queries to the root for many DNS resolvers
- Greater resilience to hostile traffic by load sharing widely distributed attacks across the entire anycast constellation, and absorbing a single point attack on a single server instance

Anycast Root Servers

12 of the 13 root server “l   of “anycast”
serv   n respond to the
sam   irect resolvers
to p   losest”
mem

Anycast is more about more effective defence against DDOS attack and not about any marginal changes in resolution performance that anycast may offer

Anycast provides:

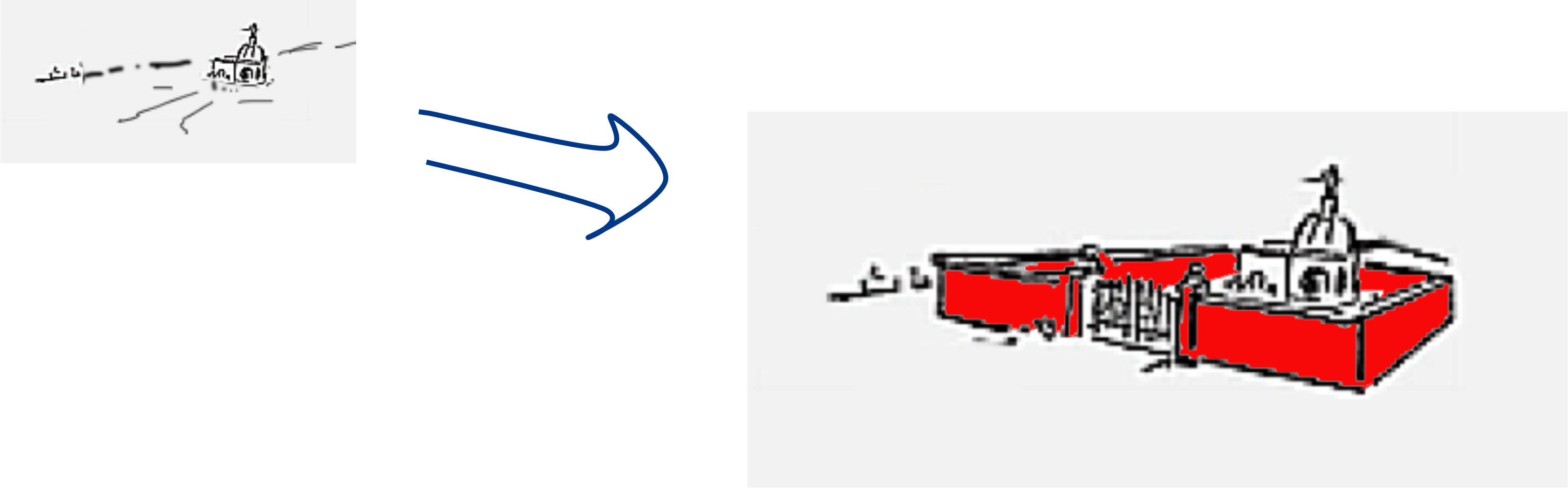
- ~~faster responses to queries to the root for many DNS resolvers~~
- Greater resilience to hostile traffic by load sharing widely distributed attacks across the entire anycast constellation, and absorbing a single point attack on a single server instance

How do we defend the Root today?

As the traffic levels to the root servers increases both as steady state query levels and instances of attacks, we keep on adding more instances to the existing anycast clouds

The attacks get bigger

Our defence is bigger walls



The attackers are our own recursive resolvers!

We are scaling the DNS root server infrastructure in order to be resilient against floods of queries about non-existent names coming from the existing DNS resolvers, who are scaling their own capabilities to survive the very same query attacks that are being directed against them!

How do we defend the Root today?

As the traffic levels to the root servers increases both as steady state query levels and instances of attacks, we keep on adding more instances to the existing anycast clouds

What we are in effect doing is building ever bigger and larger trash processors to handle ever larger amounts of garbage queries to cope with these ever larger attacks

Can we jump out of this vicious cycle?

Can we change the behaviour of the DNS system to improve both its service and its resilience?

DNSSEC changes Everything

Before DNSSEC we relied on the assumption that if we asked an IP address of a root server, then the response was genuine

With DNSSEC we can ask anyone, and then use DNSSEC validation to assure ourselves that the answer is genuine

How can we use this?

DNSSEC-Enabled Directions for the Root Service

If we could answer NXDOMAIN queries from recursive resolvers we could reduce the load on the root servers by close to 70%

This would be a very significant win:

- reducing root query traffic
- providing faster response to these queries
- reduces the local cache load on recursive resolvers

Local Root Secondaries – RFC 7706

Enlist DNS resolvers to offer a root zone secondary service

If resolvers use this approach then they only need to query a root server infrequently and perform a zone transfer of the current state of the root zone (IXFR from a root server), and use this validated copy of the root zone to directly answer all queries that refer to the root zone

NSEC caching – RFC 8198

Most of the queries seen at the root are for non-existent domains, and resolvers cache the non-existence of a given name

But a DNSSEC-signed NXDOMAIN response from the root zone actually describes a *range* of labels that do not exist, and it's the *range* that is signed, not the actual query name

If resolvers cached this range and the signed response, then they could use the same signed response to locally answer a query for any name that falls within the same label range

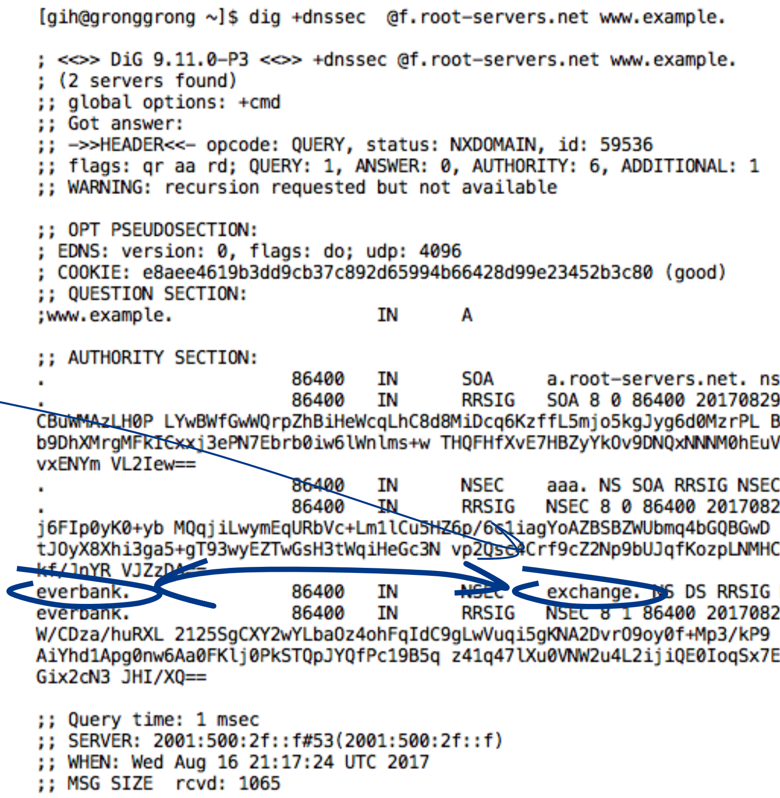
This has a similar effect to RFC7706, but without any configuration overhead, nor is there any requirement for supporting root zone transfers.

NSEC caching

For example, if you were to query the root server for the non-existant name **`www.example.`** the returned response from the root says that there are NO TLDS between **`everbank.`** and **`exchange.`**

The same response can be used to respond to queries for every TLD between these labels.

So we can cache this range response and use it to respond to subsequent queries that fall into the same range



```
[gih@gronggrong ~]$ dig +dnssec  @f.root-servers.net www.example.

; <<>> DiG 9.11.0-P3 <<>> +dnssec @f.root-servers.net www.example.
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 59536
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; COOKIE: e8aee4619b3dd9cb37c892d65994b66428d99e23452b3c80 (good)
;; QUESTION SECTION:
;www.example.                   IN      A

;; AUTHORITY SECTION:
.                       86400   IN      SOA     a.root-servers.net. ns
.                       86400   IN      RRSIG   SOA 8 0 86400 20170829
CBuWMAzLH0P LYwBWfGwWQrpZhBiHeWcqLhC8d8MiDcq6KzffL5mjo5kgJyg6d0MzrPL B
b9DhXMrgMFkICxxj3ePN7Ebrb0iw6lWnlms+w THQFHfXvE7HBZyYkOv9DNQxNNNM0hEuV
vxENYm VL2Iew==
.                       86400   IN      NSEC    aaa. NS SOA RRSIG NSEC
.                       86400   IN      RRSIG   NSEC 8 0 86400 2017082
j6FIp0yK0+yb MQqjiLwymEqURbVc+Lm1lCu5HZ6p/6c1iagYoAZBSBZWUbmq4bGQBGwD
tJOyX8Xhi3ga5+gT93wyEZTwGsH3tWqiHeGc3N vp2Qse4Crf9cZ2Np9bUJqfKozpLNMHC
kf/JnYR VJZzDA==
everbank.               86400   IN      NSEC    exchange. NS DS RRSIG
everbank.               86400   IN      RRSIG   NSEC 8 1 86400 2017082
W/CDza/huRXL 2125SgCXY2wYLbaOz4ohFqIdC9gLwVuqi5gKNA2DvrO9oy0f+Mp3/kP9
AiYhd1Apg0nw6Aa0FKlj0PkSTQpJYQfPc19B5q z41q47lXu0VNW2u4L2ijiQE0IoqSx7E
Gix2cN3 JHI/XQ==

;; Query time: 1 msec
;; SERVER: 2001:500:2f::f#53(2001:500:2f::f)
;; WHEN: Wed Aug 16 21:17:24 UTC 2017
;; MSG SIZE  rcvd: 1065
```

Architecturally speaking…

- Rather than have recursive resolvers act as “amplifiers” for DNS queries for non-existent names, NSEC caching enlists these recursive resolvers to act on behalf of the root servers, and provide the answers for them.
- This approach uses existing DNS functionality and existing queries – there is nothing new in this.
- The change here is to take advantage of the use of the NSEC response to define a range of names, allowing what is in effect semi-wildcard cache entries that can be used to respond to a range of query labels

Impacts…

- Rather than trying to expand the capabilities of the root zone servers, we can leverage the massive number of already deployed recursive resolvers to extend their cache to cover both defined and non-existant root labels
- We anticipate that this will have a major effect on the DNS by absorbing most of the current root query load at the edge, rather than passing these queries into the root system

Impacts…

NSEC caching can also help recursive resolvers

- Instead of caching non-existent individual names they can cache the NSEC-described range, and refresh the cached NSEC record instead of any individual name
- This will shrink the demands placed on the local cache, which can improve local cache performance in the recursive resolver

Coming to a Bind Resolver near you

APNIC has sponsored the inclusion of this NSEC caching code for the root zone in the forthcoming Bind 9.12 release

This function will be enabled by default in this release

We hope that other DNS resolver vendors also implement this feature, as widespread use of NSEC caching will have a dramatic positive impact on the root server ecosystem!

APNIC 44

#apnic44

TAICHUNG, TAIWAN

7–14 September 2017