APNIC 44

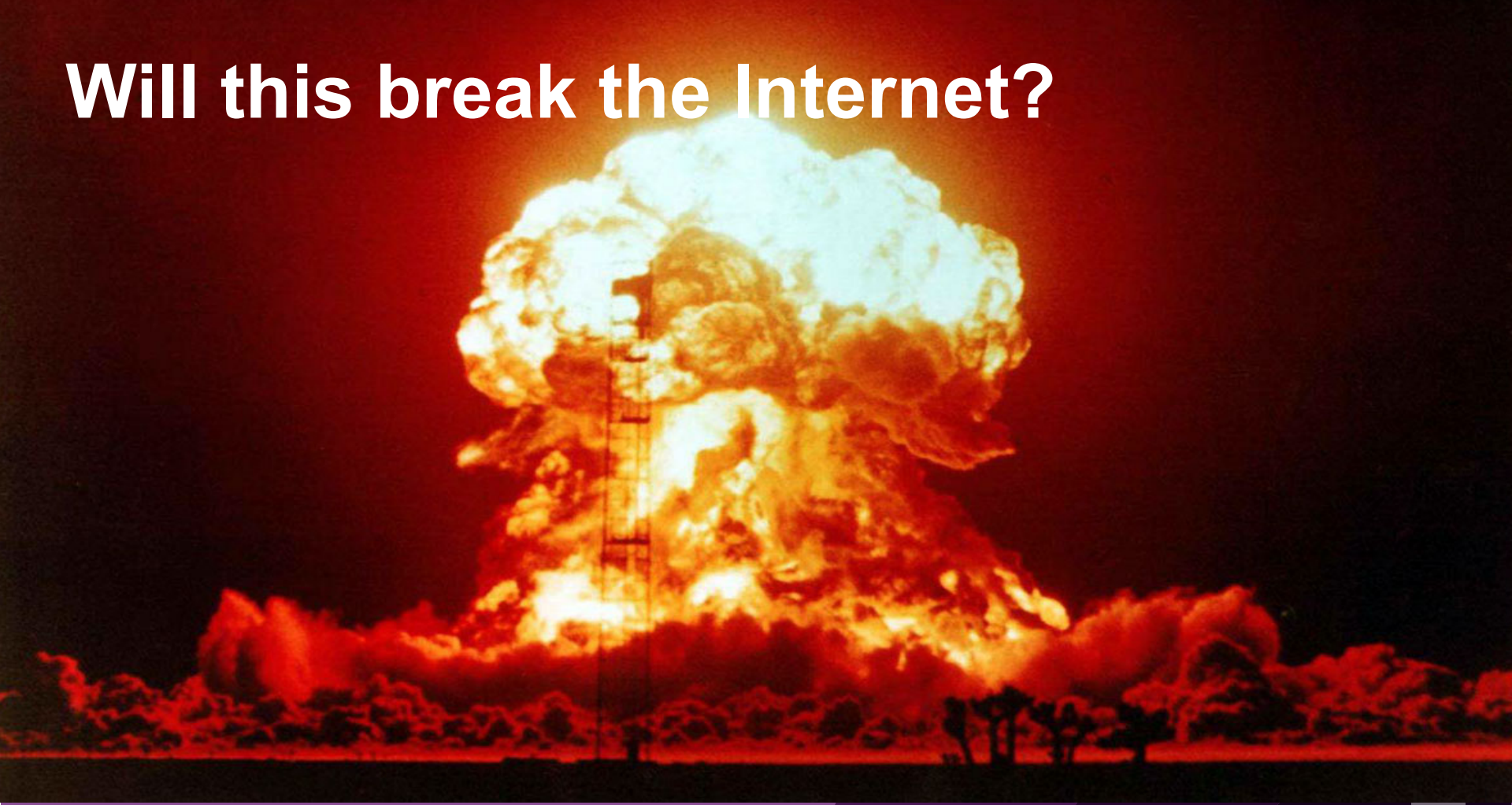
Rolling the Root KSK

Geoff Huston
APNIC Labs
September 2017

TAICHUNG, TAIWAN
7–14 September 2017

#apnic44

Will this break the Internet?



Why?

If we stuff up this trust anchor key roll then resolvers that perform DNSSEC validation will fail to provide responses

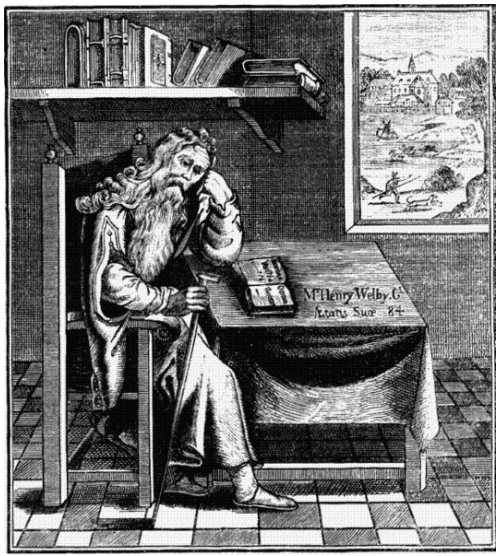
- For DNSSEC-signed names!
- and
- For unsigned names!

These resolvers will go completely dark if they lose their relationship with the signed root of the DNS

Ok – lets strip out the hyperbole!

Can we estimate the extent of the Internet's population (both human and other) that MAY be impacted by this change in the DNSSEC Trust Anchors?

Can we estimate the LIKELY impact of this change?



How many users …

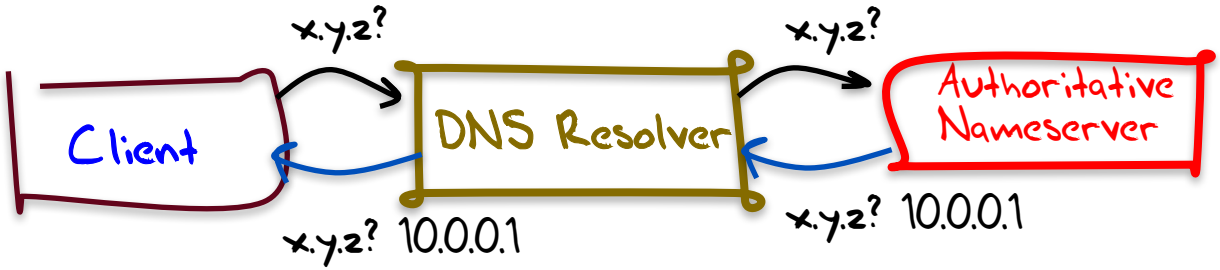
Send their DNS queries towards recursive resolvers that perform DNSSEC validation?

Because these resolvers WILL be sensitive to a change in the KSK and this number is an approximate estimate of the upper bounds of impact of the KSK change

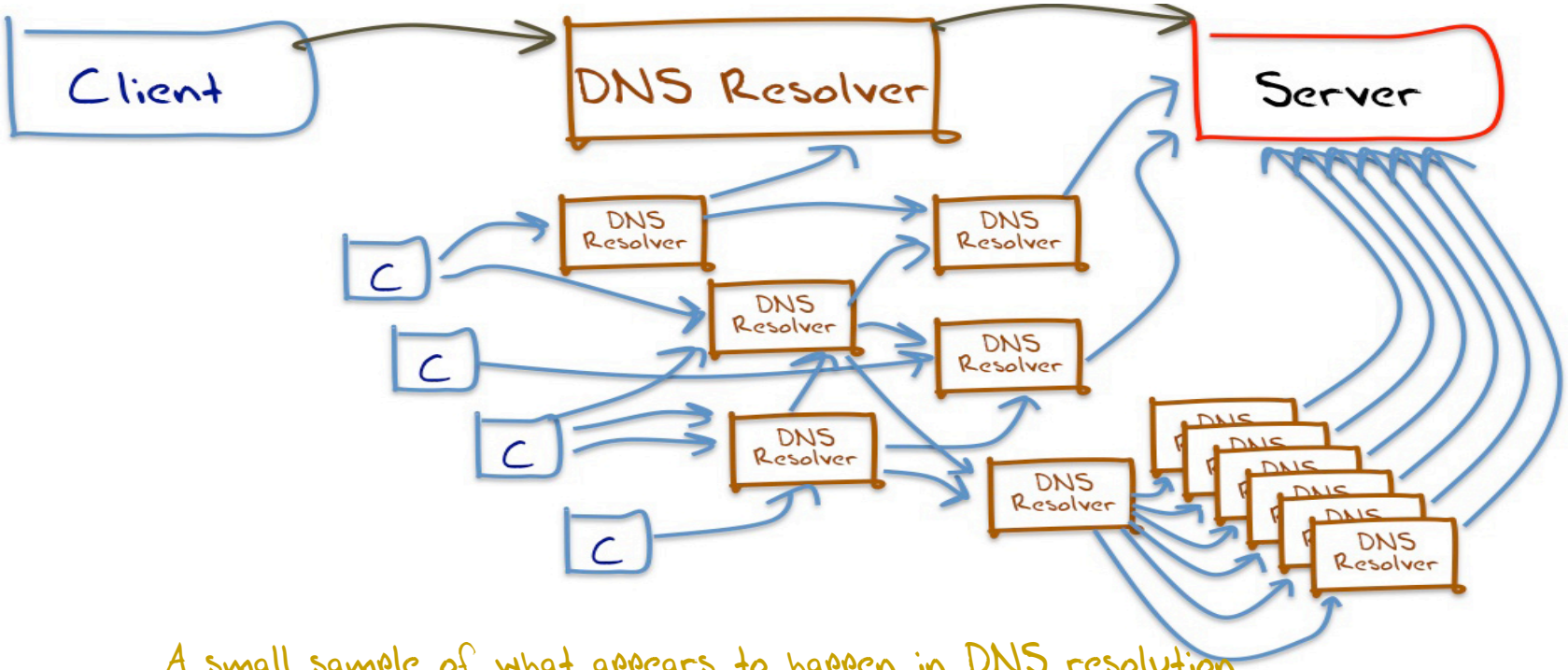
Digression: Let’s measure DNSSEC

Understanding DNS Resolvers is “tricky”

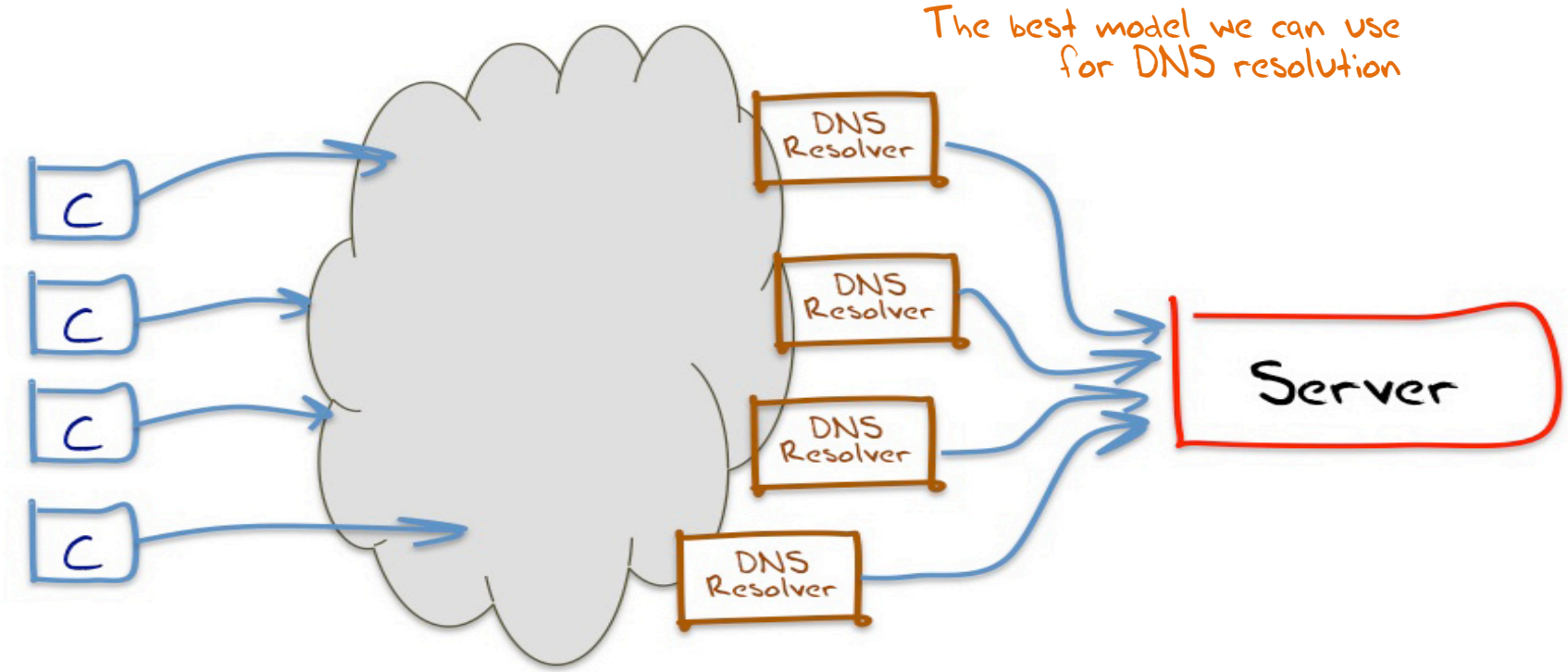
What we would like to think happens in DNS resolution!



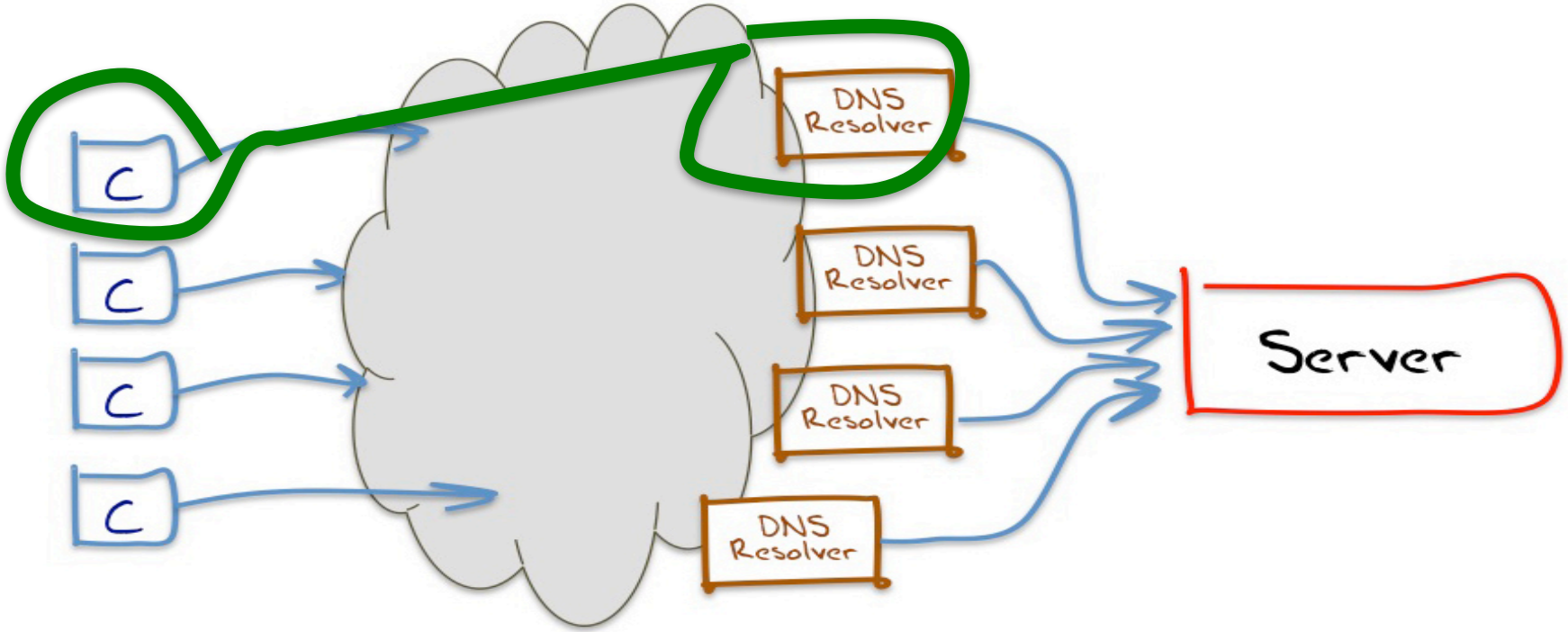
Understanding DNS Resolvers is “tricky”



Understanding DNS Resolvers is “tricky”



Understanding Resolvers is “tricky”



This means…

That it's hard to talk about "all resolvers"

- We don't know the ratio of the number of resolvers we cannot see compared to the resolvers we can see from the perspective of an authoritative name server

We can only talk about "visible resolvers"

This means…

And there is an added issue here:

- It can be hard to tell the difference between a visible resolver performing DNSSEC validation and an occluded validating resolver performing validation via a visible non-validating forwarder

This means…

And there is an added issue here:

- It can be hard to tell the difference bet… performing DNSSEC … rming validation via a visi…

(Yes, I know it's a subtle distinction, but it makes looking at RESOLVERS difficult!)

This means…

It’s easier to talk about end clients rather than resolvers, and whether these end clients use / don’t use a DNS resolution service that performs DNSSEC validation

Ok – so measuring the DNS is tricky

But we want to measure DNSSEC validation.

Server-Side Measurement

We can’t instrument the user-side

Instead we instrument the server side, and capture all packets to the authoritative DNS servers and the web servers

So we are trying to infer the capabilities of the end user environment based upon the queries we see at our servers in response to passing the user a “known” question that they have to resolve

The Experiment

We have an online Ad that contains a scripted collection of URLS to fetch – when the ad is “impressed” the ad script is executed by the user and the user attempts to retrieve all the listed URLs

Three URLs:

the good (DNSSEC signed)

the bad (invalid DNSSEC signature)

the control (no DNSSEC at all)

DNSSEC Validating

DNSSEC-Validating resolvers will:

- ask for the DNSKEY and DS RRs for both names
- fetch the valid-signed object
- NOT fetch the Invalid-signed object

Non DNSSEC-validating resolvers will:

- Not ask for DNSKEY and DS RRs
- fetch both objects

But what if I have two resolvers in my local config, one validates, one does not?

- ask for the DNSKEY and DS RRs for both names
- fetch both objects

DNSSEC Validating

DNSSEC-Validating resolvers will:
- ask for the DNSKEY and DS RRs for both names
- fetch both objects
- Validate the Invalid-signed object

Non DNSSEC-validating resolvers will:
- Not ask for DNSKEY and DS RRs
- fetch both objects

But what if I have two resolvers in my local config, one validates, one does not?
- ask for the DNSKEY and DS RRs for both names
- fetch both objects

So lets measure DNSSEC

How many users use ONLY DNS resolvers that perform DNSSEC validation?

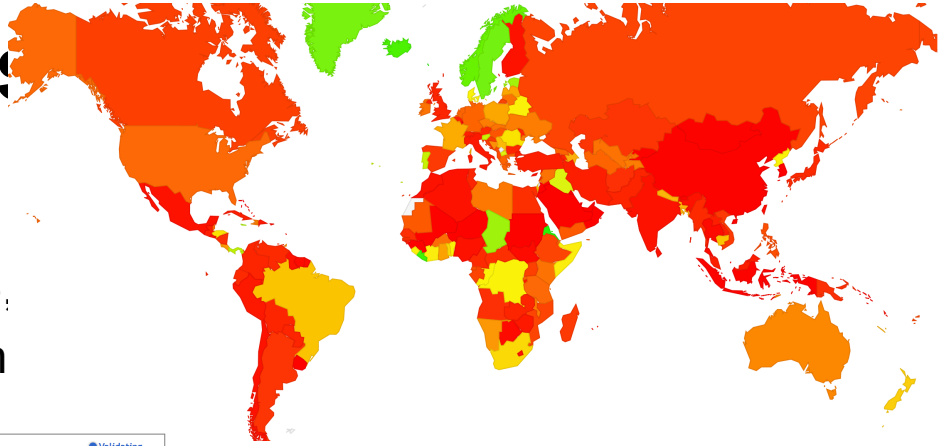
Measurement Results

August 2017

- Presented: 426,676,126 experiments to clients
- 53,121,177 experiments showed behaviour that was consistent with DNSSEC validation

- i.e. **12.45%** of users use DNSSEC-validating resolvers!

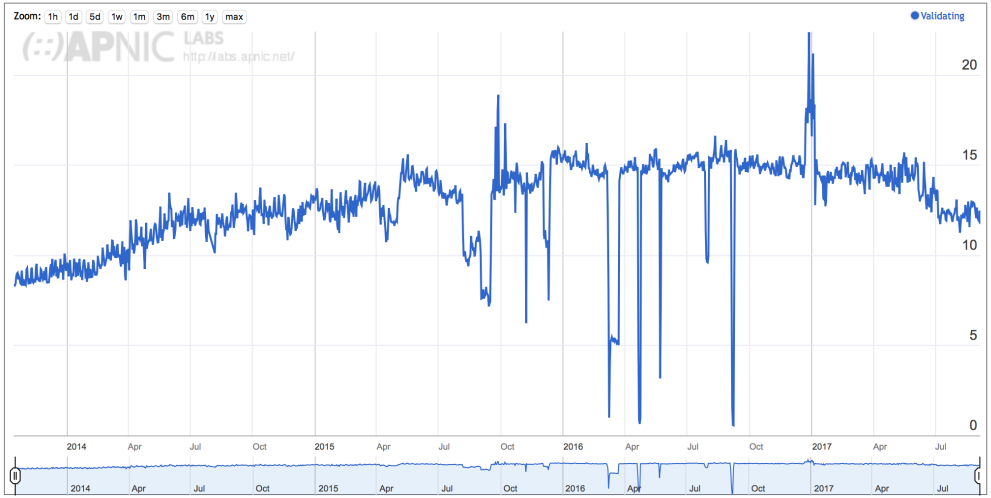
Measurement Res



August 2017

- Presented: 426,676
- 53,121,177 experim

Use of DNSSEC Validation for World (XA)



DNSSEC-validating

DNSSEC Validating

DNSSEC-Validating resolvers will:

- ask for the DNSKEY and DS RRs for both names
- fetch the Valid-signed object
- NOT fetch the Invalid-signed object

Non DNSSEC-validating resolvers will:

- Not ask for DNSKEY and DS RRs
- fetch both objects

But what if I have two resolvers in my local config, one validates, one does not?

- ask for the DNSKEY and DS RRs for both names
- fetch both objects

“Partial” DNSSEC Validating

DNSSEC-Validating resolvers will:

- ask for the DNSKEY and DS RRs for both names
- fetch the valid-signed object
- NOT fetch the Invalid-signed object

12.45%

Non DNSSEC-validating resolvers will:

- Not ask for DNSKEY and DS RRs
- fetch both objects

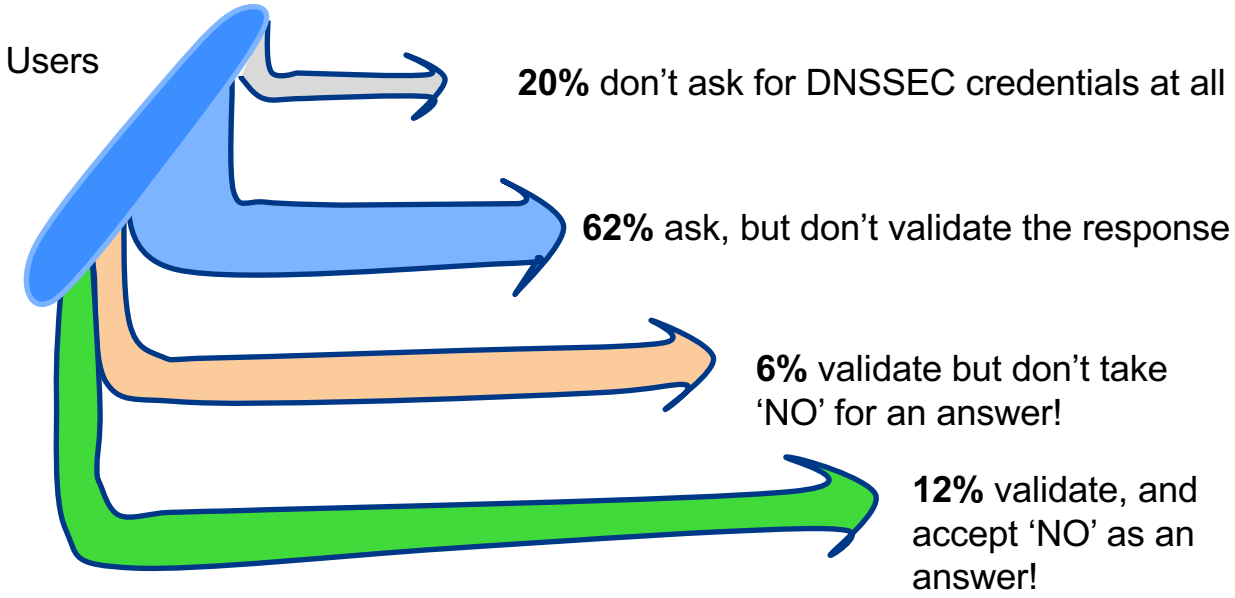
But what if I have two resolvers in my local config, one validates, one does not

- ask for the DNSKEY and DS RRs for both names
- fetch both objects

5.75%

DNSSEC Use in the Internet

There is a lot of DNSSEC validation out there!

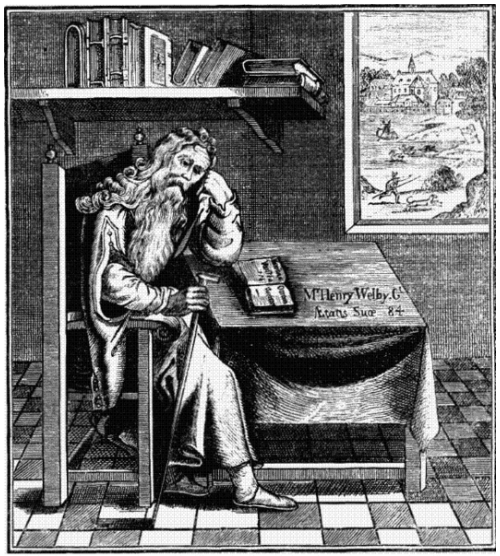


End Digression

Ok – lets strip out the hyperbole!

Can we estimate the extent of the Internet's population (both human and other) that MAY be impacted by this change in the DNSSEC Trust Anchors?

Can we estimate the LIKELY impact of this change?



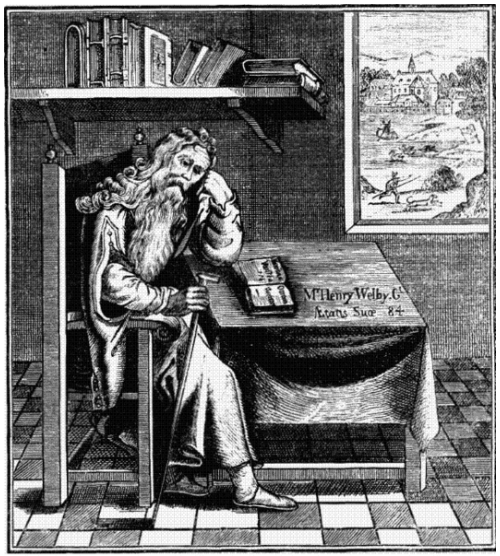
Ok – lets strip out the hyperbole!

Can we estimate the extent of the Internet's population of human users behind validating DNS resolvers?

Somewhere between 1 in 5 and 1 in 8 users use DNSSEC-aware DNS resolvers

...nge in the ...rust Anchors?

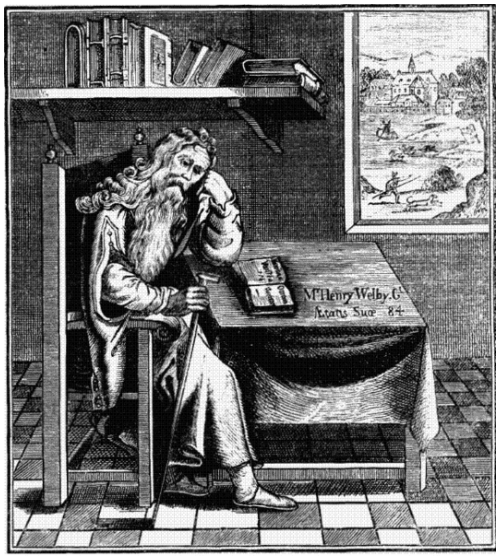
Can we estimate the LIKELY impact of this change?



Ok – lets strip out the hyperbole!

Can we estimate the extent
the Internet's
human
Somewhere between 1 in 5 and 1 in 8 users use DNSSEC-aware DNS resolvers
ange in the
rust Anchors?

Can we estimate the LIKELY impact of this change?



Our Major Concerns

1. That resolvers who validate DNS responses will fail to pick up the new DNS root key
   - they do not have code that follows RFC5011 procedures for the introduction of a new KSK
   - Or they are using a manually loaded key as the trust point
   - Or they came in late!

2. The resolvers will be unable to receive the larger DNS responses that will occur during the dual signature phase of the rollover

Let the keys roll automatically

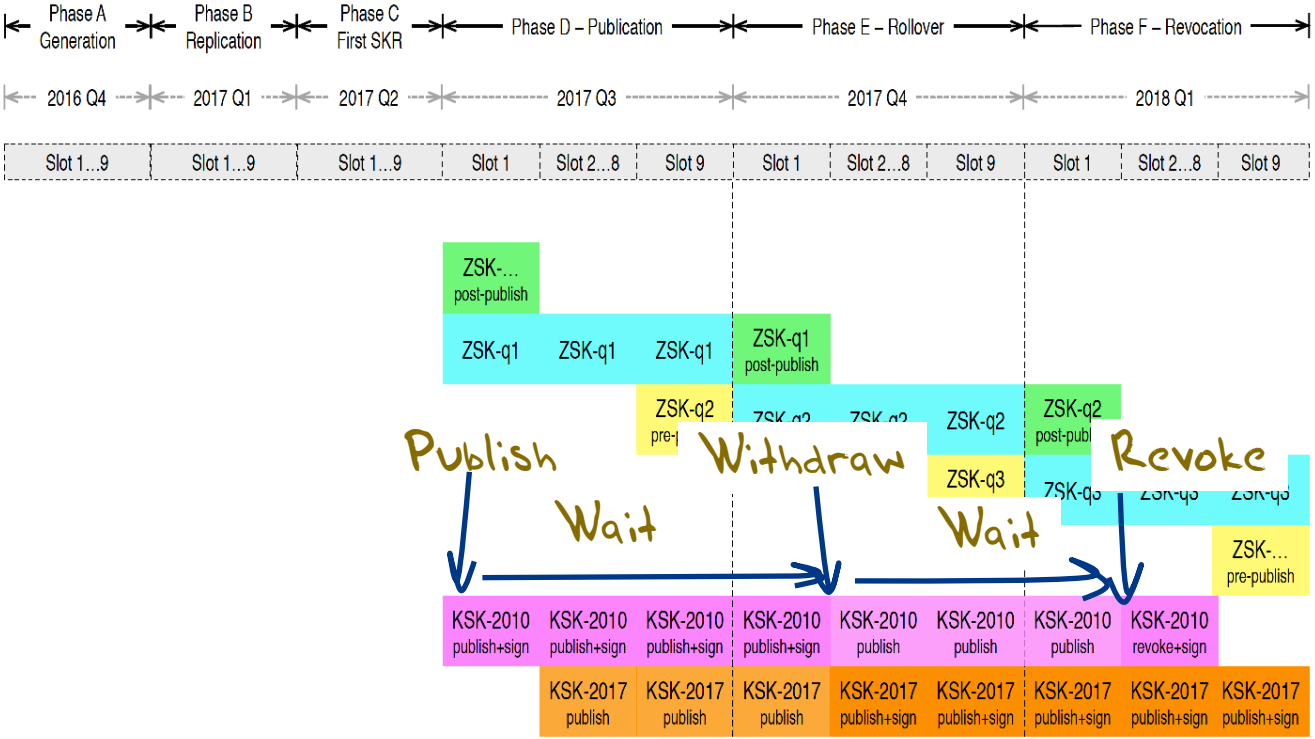
```
# // recursive resolver configuration - Bind

…

managed-keys {

        . initial-key 257 3 5 "AwEAAfdqNV

          JMRMzrppU1WnNW0PWrGn4x9dPg

…

          =„; };
```

A Bind resolver uses the “managed keys” clause in its configuration to allow the KSK to be managed automatically.

When a new KSK is added to the DNSKEY record of the root, signed by the trusted key, then the resolver recognises this as a candidate trusted key. After 30 days of continuous publication of this new key, the resolver is prepared to trust the new KSK

Easy, Right? Just follow RFC5011…

- Publish a new KSK and include it in DNSKEY responses, signed by the old KSK
  - Resolvers use old-signs-over-new to pick up the new KSK, validate it using the old KSK, and add the new KSK to the local cache of trust anchor material (i.e. this steps allows resolvers to “learn” the new KSK as a trust point)
- Wait
  - For at least 30 days
- Withdraw the old KSK
  - And sign the DNSKEY RR in the root zone with only the new KSK
- Wait
  - For a a while
- Revoke the old KSK
  - Because its never wise to keep old information in a trusted state


Phase A Generation
Phase B Replication
Phase C First SKR
Phase D – Publication
Phase E – Rollover
Phase F – Revocation
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
Slot 1...9
Slot 1...9
Slot 1...9
Slot 1
Slot 2...8
Slot 9
Slot 1
Slot 2...8
Slot 9
Slot 1
Slot 2...8
Slot 9
ZSK-... post-publish
ZSK-q1
ZSK-q1
ZSK-q1
ZSK-q1 post-publish
ZSK-q2 pre-
ZSK-q2
ZSK-q2 post-publ
ZSK-q3
Publish
Wait
Withdraw
Wait
Revoke
ZSK-... pre-publish
KSK-2010 publish+sign
KSK-2010 publish+sign
KSK-2010 publish+sign
KSK-2010 publish+sign
KSK-2010 publish
KSK-2010 publish
KSK-2010 publish
KSK-2010 revoke+sign
KSK-2017 publish
KSK-2017 publish
KSK-2017 publish
KSK-2017 publish+sign
KSK-2017 publish+sign
KSK-2017 publish+sign
KSK-2017 publish+sign
KSK-2017 publish+sign

Our Major Concerns

1. That resolvers who validate DNS responses will fail to pick up the new DNS root key
   - they do not have code that follows RFC5011 procedures for the introduction of a new KSK
   - Or they are using a manually loaded key as the trust point
   - Or they came in late!

Our Major Concerns

1. That resolvers who ... pick

We can't measure (yet) how many resolvers will pick up the new KSK

If RFC8145 was commonly implemented then we could tell what was going on with take up of the new key according to the RFC5011 procedures

But this tool is not around in resolvers yet

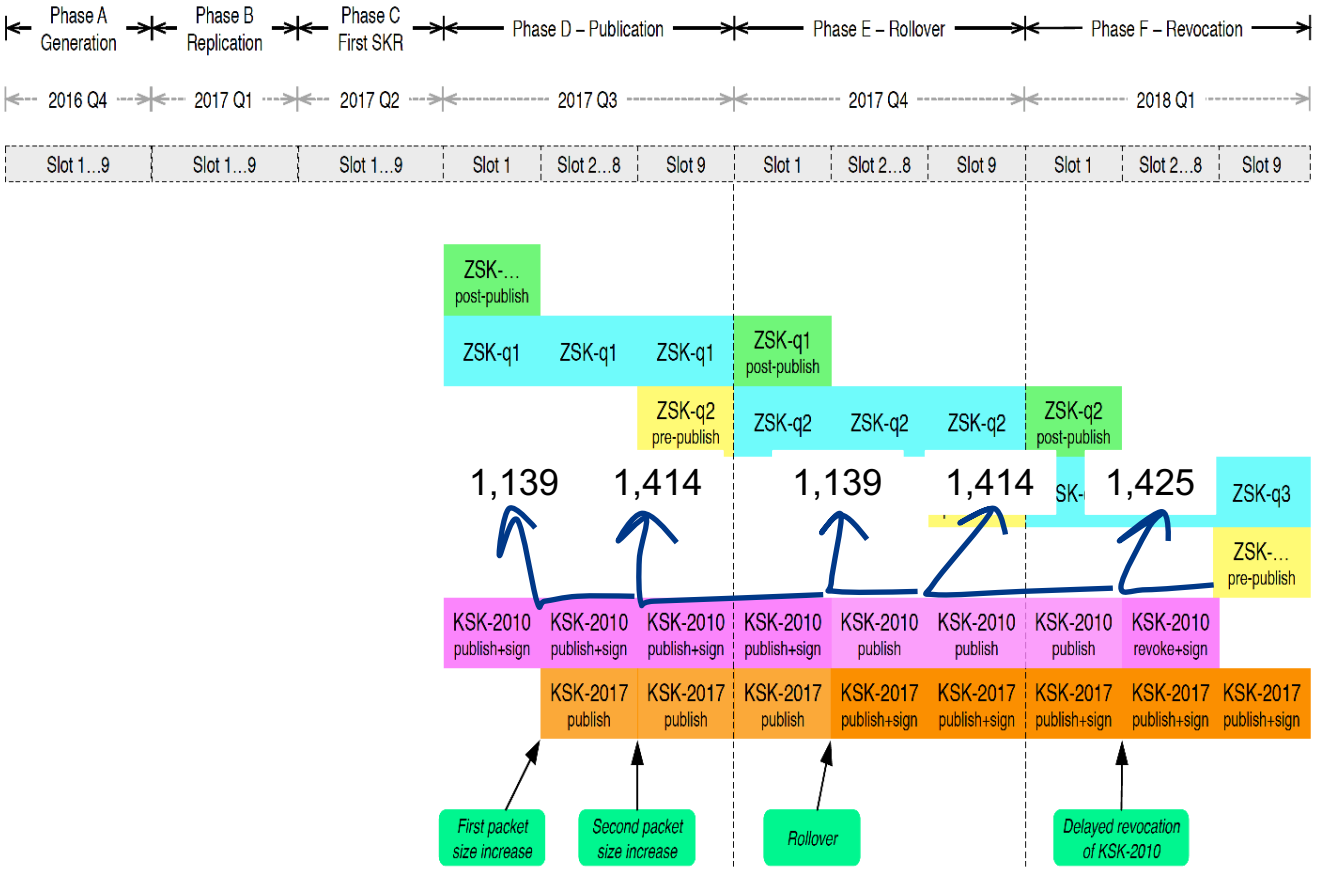
- So we just can't tell whether manually (mis)managed keys are prevalent or not

We will only know after October 11

Our Major Concerns

1. That resolvers who validate DNS responses will fail to pick up the new DNS root key
   - they do not have code that follows RFC5011 procedures for the introduction of a new KSK
   - Or they are using a manually loaded key as the trust point
   - Or they came in late!

2. The resolvers will be unable to receive the larger DNS responses that will occur during the dual signature phase of the rollover


Phase A Generation
Phase B Replication
Phase C First SKR
Phase D – Publication
Phase E – Rollover
Phase F – Revocation
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
Slot 1...9
Slot 1...9
Slot 1...9
Slot 1
Slot 2...8
Slot 9
Slot 1
Slot 2...8
Slot 9
Slot 1
Slot 2...8
Slot 9
ZSK-... post-publish
ZSK-q1
ZSK-q1
ZSK-q1
ZSK-q1 post-publish
ZSK-q2 pre-publish
ZSK-q2
ZSK-q2
ZSK-q2
ZSK-q2 post-publish
1,139
1,414
1,139
1,414
1,425
ZSK-q3
ZSK-... pre-publish
KSK-2010 publish+sign
KSK-2010 publish+sign
KSK-2010 publish+sign
KSK-2010 publish+sign
KSK-2010 publish
KSK-2010 publish
KSK-2010 publish
KSK-2010 revoke+sign
KSK-2017 publish
KSK-2017 publish
KSK-2017 publish
KSK-2017 publish+sign
KSK-2017 publish+sign
KSK-2017 publish+sign
KSK-2017 publish+sign
KSK-2017 publish+sign
First packet size increase
Second packet size increase
Rollover
Delayed revocation of KSK-2010

Large Responses

The larger DNS responses invoke different behaviours:

- Some root servers hand back a large unfragmented UDP packet
- Some root services hand back a fragmented UDP packet
- Some root servers hand back a truncated UDP packet, with fallback to TCP

Large Responses

- This presents a problem with testing – not all root server instances behave the same way when delivering large responses
- We can test each behaviour in isolation, but to test the diversity of the root server environment is beyond the capabilities of reasonable accuracy of our ad-based measurement framework
- Our tests with 1,430 octet responses in IPv4 show that the noise component drowns out any coherent signal – the loss rate is less than 1%
- IPv6 only has a higher loss rate for UDP fragmentation (40%), but as long as a resolver is dual-stack then this is not a major operational issue

- As a related data point, .org has been running a DNSKEY response of 1,650 octets for some years, and nobody is calling out .org as an operational failure!

Large Responses

- This presents a problem with testing – not all root server instances behave the same way when delivering large responses
- We can test each behaviour in isolation, but to test ... server environment is beyond the capability ... ad-based measurement framework
- Our tests with ... show that the noise component drops ... loss rate is less than 1%
- ... higher loss rate for UDP fragmentation (40%), but as long as a ... is dual-stack then this is not a major operational issue

We don't expect large response failure to be a major factor in KSK roll 'breakage'

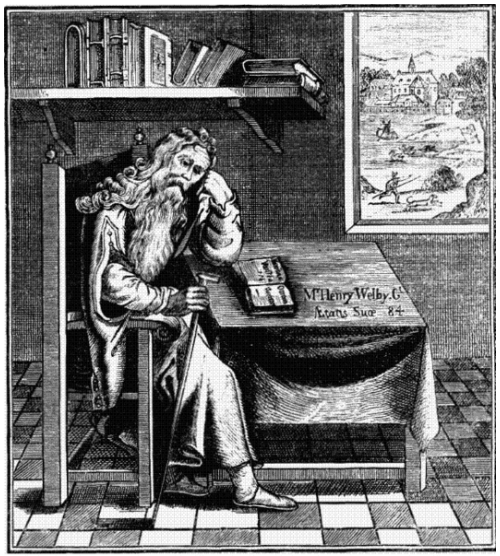
- As a related data point, .org has been running a DNSKEY response of 1,650 octets for some years, and nobody is calling out .org as an operational failure!

Ok – lets strip out the hyperbole!

Can we estimate the extent
the Internet's
human
Somewhere between 1 in 5 and 1 in 8 users use DNSSEC-aware DNS resolvers
ange in the
rust Anchors?

Can we
We have no idea about the extent of issues with manual key management

Less than 1 in 100 MIGHT be affected by the large response size



Where are we?

- A key roll of the Root Zone KSK will cause some DNS resolvers to fail:
  - Resolvers who do not pick up the new key in the manner described by RFC5011
  - Resolvers who cannot receive a DNS response of >1,400 octets
  - The failure will not occur at the exact time of the key roll – it will occur when the local cache of old signed root entries ages out of the cache, which will take up to 7 days
- Many users who use these failing resolvers will just switch over to use a non-validating resolver
- A small pool of users will be affected with no DNS

What can I do?

Check your recursive resolver config!

Check your trusted key set

There is no need to turn off DNSSEC validation!

Questions?