

# Introduction To DNS



*everything you never wanted to  
know about IP directory services*

Linux Users Victoria, April 3<sup>rd</sup> 2007  
Jonathan Oxaer <jon@ivt.com.au>



***what is the  
domain name  
system anyway?***



**it's like a  
phone book  
...kinda**



# DNS is (1) a directory service



# DNS is (2) an identity mechanism



# DNS is (3) a namespace structure



# DNS is (4) an abstraction layer



**think of the  
phone book...**



**maps**

# **hostnames**

**to**

# **IP addresses**



**maps**

**jon.oxer.com.au**

**to**

**221.133.213.151**



# forward vs reverse



maps

**221.133.213.151**

to

**jon.oxer.com.au**



# simple beginnings:

# hosts.txt

...but  
phone books



don't  
scale



so modern

**DNS is managed  
like a distributed  
phone book**



# DNS is (5) delegation of authority

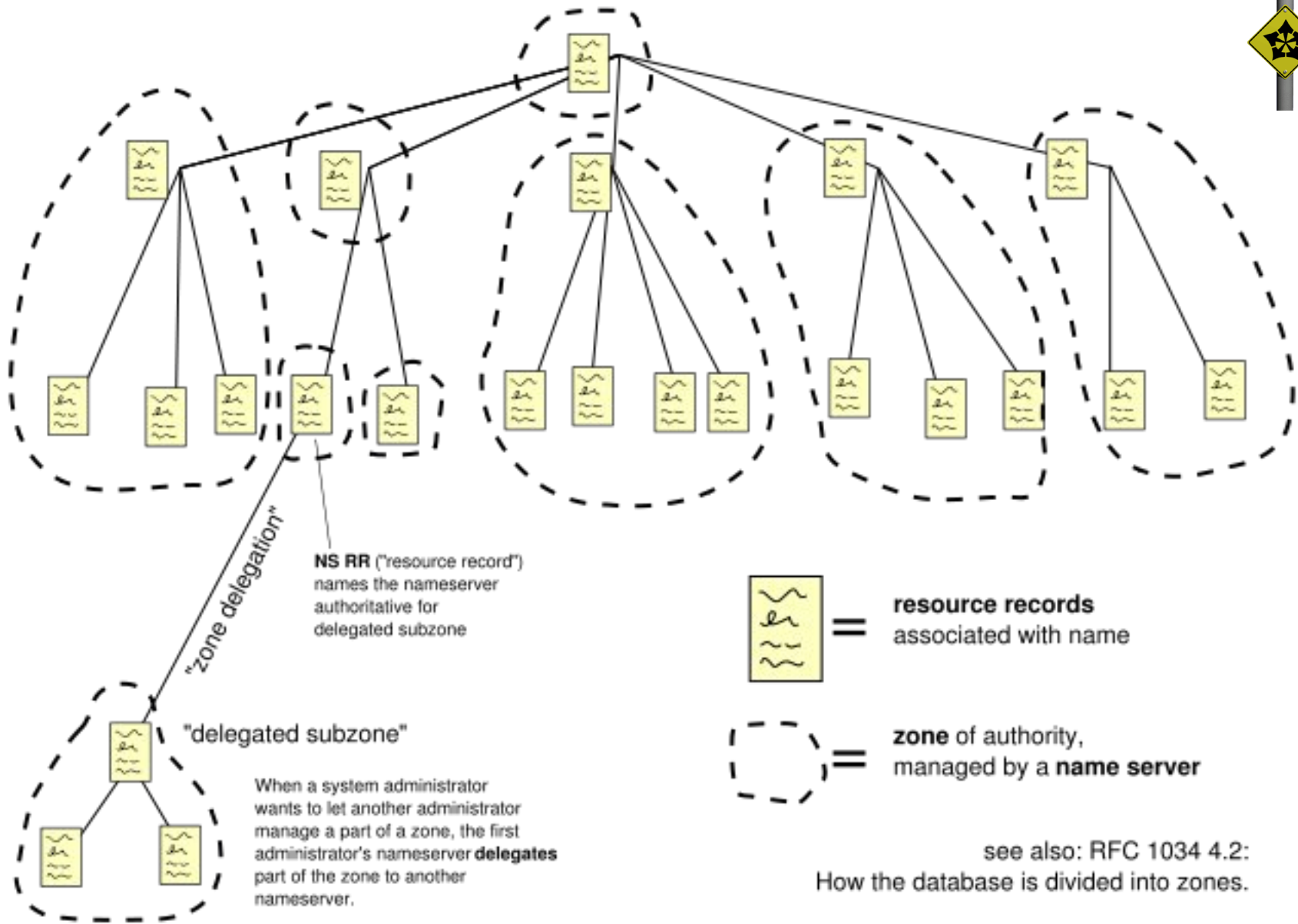


a “zone”  
defines an area  
of authority



**think of it  
as an  
inverted tree**

# Domain Name Space





# anatomy of a host name



**(a host name is  
a record inside  
a domain name)**



**read right to left:  
jon.oxer.com.au.**



**yes, it really  
ends in a dot!**



**root zone:**  
**jon.oxer.com.au.**



**top level domain:**  
**jon.oxer.com.au.**



# 2nd level zone: jon.oxer.com.au.



**3rd level zone:**  
**jon.oxer.com.au.**



**host name:**  
**jon.oxer.com.au.**



**back to that dot:**  
**jon.oxer.com.au.**



# **“ICANN’s 13”: the A to M root servers**



# root.hints



***“There can be  
only 13”***



**(UDP packets  
limited to 512B)**



**A response with  
more than 13  
entries > 512B**



**root servers**  
**replicated**  
**globally**  
**using anycast**





**root servers**

**delegate**

**ccTLDs, gTLDs,**

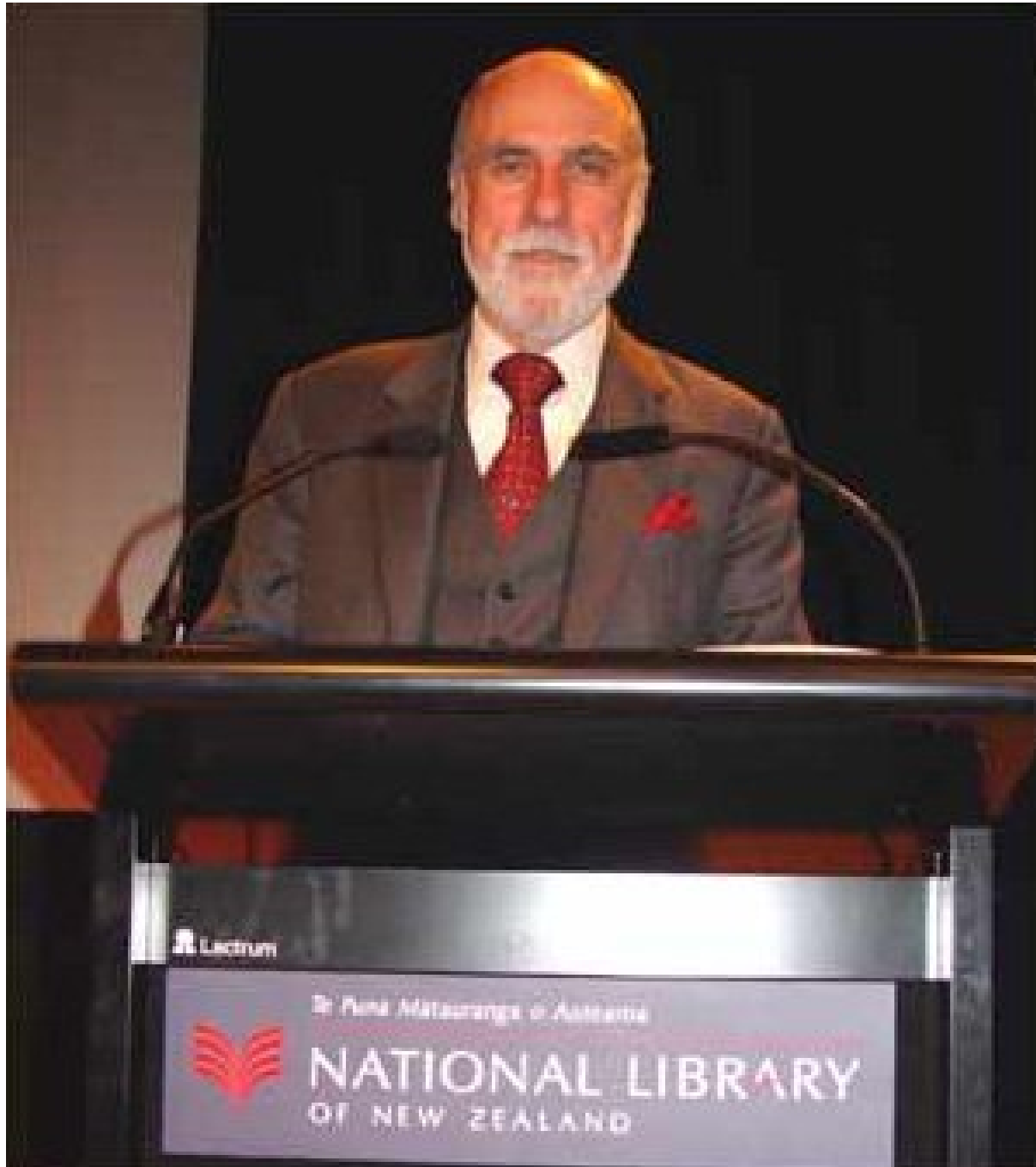
**and iTLDs**



so what is this  
*“delegation”*  
of which  
you speak?



**registries, registrars,  
resellers, registrants,  
InterNIC, ICANN,  
OpenSRS, oh my!**





# ICANN controls the registries



# registries control the registrars



# registrars control delegations



# domain allocation policies



# own or lease?



# trademarks and disputes





# alt roots (alternative DNS roots)

**DNS works  
because we  
agree to let  
it work**





**alt roots are  
just alternative  
agreements**



# critical concept alert!





# authoritative vs recursive servers



*authoritative*  
servers answer  
questions about  
zones they own



*recursive*  
resolvers query  
other servers  
on your behalf



**recursive  
lookups require  
multiple queries**

You!

resolver  
library

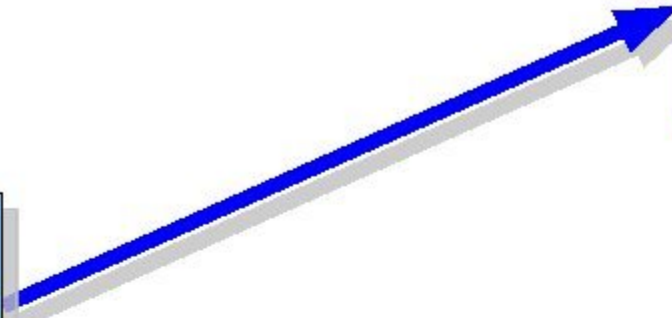
recursive  
resolver



jon.oxer.com.au?

resolver  
library

recursive  
resolver



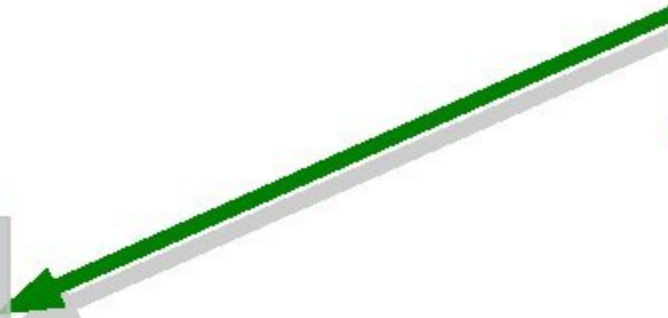
“ ”  
.  
nameserver

resolver  
library

“ ”  
nameserver

recursive  
resolver

resolver  
library



“ ”  
.  
nameserver

recursive  
resolver



“.au.”  
nameserver

resolver  
library

“ ”  
.  
nameserver

recursive  
resolver

“.au.”  
nameserver

resolver  
library



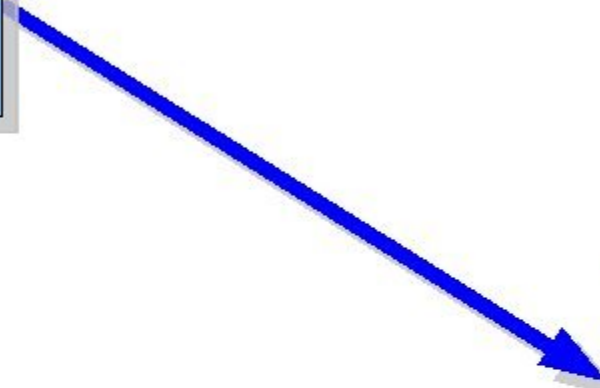
“.”  
nameserver

“.au.”  
nameserver

“.com.au.”  
nameserver

recursive  
resolver

resolver  
library



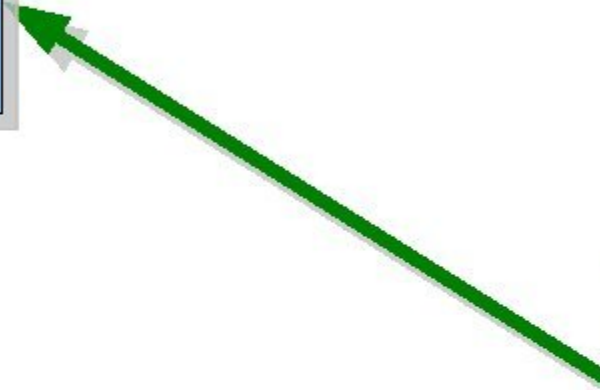
“.”  
nameserver

“.au.”  
nameserver

“.com.au.”  
nameserver

recursive  
resolver

resolver  
library



“.”  
nameserver

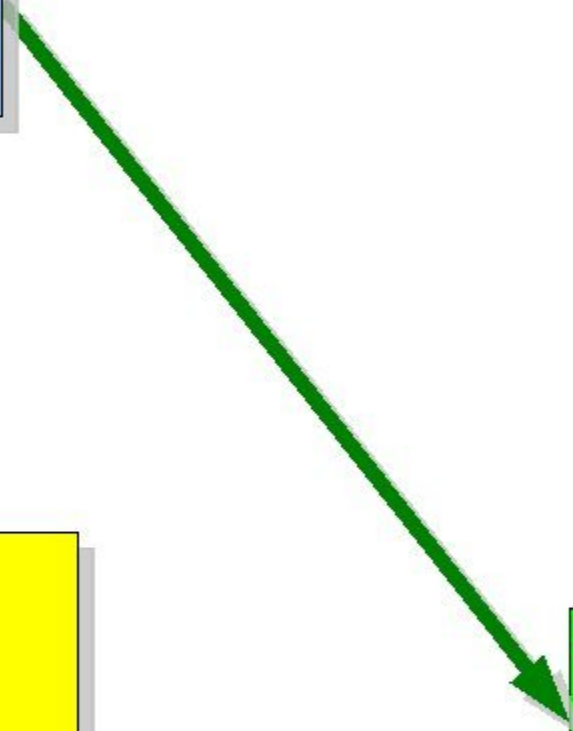
“.au.”  
nameserver

“.com.au.”  
nameserver

“.oxer.com.au.”  
nameserver

recursive  
resolver

resolver  
library



“ ”  
nameserver

“.au.”  
nameserver

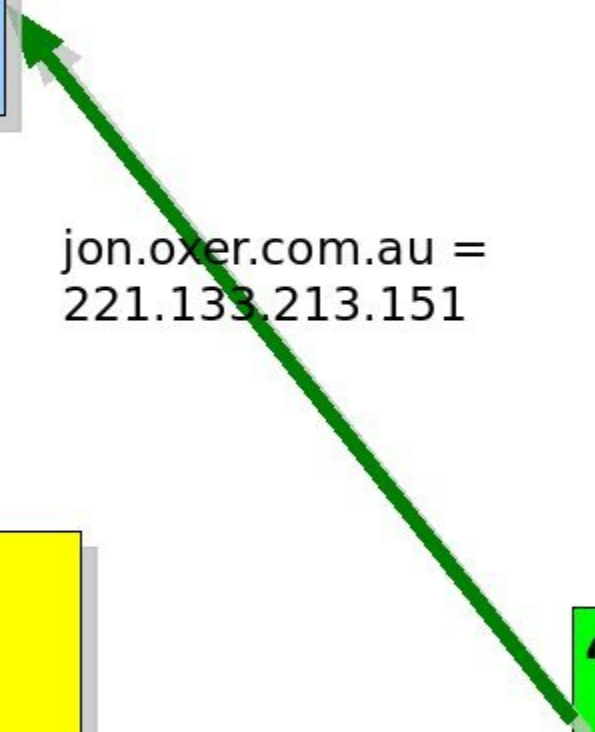
“.com.au.”  
nameserver

“.oxer.com.au.”  
nameserver

jon.oxer.com.au =  
221.133.213.151

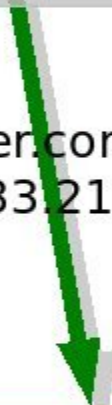
recursive  
resolver

resolver  
library



recursive  
resolver

jon.oxer.com.au =  
221.133.213.151



resolver  
library

“ ”  
nameserver

“.au.”  
nameserver

“.com.au.”  
nameserver

“.oxer.com.au.”  
nameserver

# caching good!



# caching bad!





# beware the cache

# caching: in the recursive DNS resolver





**(Big Pond bad!  
Bad, I say!)**



# caching: in your OSs resolver library

# caching: directly inside applications





**(IE very  
bad too!)**



# internationalisation



# anatomy of a zone[file]



```
; zone file for example.com.
$TTL 2d      ; 172800 TTL
@           IN      SOA      ns1.example.com. hostmaster.example.com. (
                                2007040304 ; serial
                                12h         ; refresh
                                15m         ; retry
                                3w          ; expiry
                                3h         ; minimum
                                )
           IN      NS       ns1.myprovider.com.
           IN      NS       ns1.example.com.
           IN      MX      10 mail.example.net.
homer     IN      A        192.168.254.3
marge     IN      A        192.168.12.15
www       IN      CNAME    homer
vpn       IN      CNAME    marge
```



# types of DNS records

“A”



(address)

links names and  
IPv4 addresses

**“AAAA”**

**(address)**



**links names and  
IPv6 addresses**

# “CNAME”



# (canonical name)

# aliases names to other names

**“MX”**



**(mail exchange)**

**name of machine  
for mail delivery**

# “NS”



# (name server)

# name of DNS server for a zone

**“TXT”  
(text)**



**arbitrary text  
string**

# “NAPTR”



# (naming auth pointer)

# fun with regex

# “SOA”



# (start of authority)

# controls inter-server data synchronisation



# SOA (Start Of Authority)



# SOA sets TTL (Time To Live)



**TTL says how  
long data may  
be cached**



# SOA parameters

***Serial*: identifies  
version of SOA**



# SOA parameters

***Refresh*: seconds  
between updates**



# SOA parameters

***Retry*: seconds to  
wait after failure**



# SOA parameters

***Expire*: seconds  
before data flushed**



# SOA parameters

***Minimum:*** used now  
for negative caching



# circular dependencies: self-delegation



# the solution: glue records



# breaking your brain: reverse DNS



# Let's look up 1.2.3.4!



# 4.3.2.1.in-addr.arpa.



# security



# DNS cache poisoning





# Practical example:

# Dr Evil wants to take over “www.bigbank.com”



# Dr Evil attack vector #1

**redirecting the target  
domain's nameserver**



(1)

**Dr Evil creates a sub-zone of a zone he controls, such as “bigbank.dr-evil.com”**

(2)



**Dr Evil delegates his  
evil zone to  
“www.bigbank.com”**



**(3)**

**Dr Evil configures his  
DNS server to return  
the wrong IP address  
for “www.bigbank.com”**



(4)

**Dr Evil issues a DNS  
lookup for  
“bigbank.dr-evil.com”  
to your DNS resolver**



(5)

**Your DNS server caches  
the evil IP and uses it for  
future requests for  
“www.bigbank.com”**

# what happened?

request:

`bigbank.dr-evil.com. IN A`



# what happened?



**request:**

`bigbank.dr-evil.com. IN A`

**response:**

**Answer:**

`(no response)`

# what happened?



**request:**

`bigbank.dr-evil.com. IN A`

**response:**

**Answer:**

`(no response)`

**Authority section:**

`bigbank.dr-evil.com. 3600 IN NS www.bigbank.com.`

# what happened?



**request:**

```
bigbank.dr-evil.com. IN A
```

**response:**

**Answer:**

(no response)

**Authority section:**

```
bigbank.dr-evil.com. 3600 IN NS www.bigbank.com.
```

**Additional section:**

```
www.bigbank.com. IN A 1.2.3.4
```





# Dr Evil attack vector #2

**redirect the NS record  
of the target domain**



# compare this with...

## request:

```
bigbank.dr-evil.com. IN A
```

## response:

Answer:

(no response)

Authority section:

```
bigbank.dr-evil.com. 3600 IN NS www.bigbank.com.
```

Additional section:

```
www.bigbank.com. IN A 1.2.3.4
```

# ...alternative attack



request:

```
bigbank.dr-evil.com. IN A
```

response:

Answer:

(no response)

Authority section:

```
bigbank.com. 3600 IN NS ns.dr-evil.com.
```

Additional section:

```
ns.dr-evil.com. IN A 1.2.3.4
```



# Dr Evil attack vector #3

**DNS forgery:  
respond before the  
real nameserver**



**not as easy  
as it sounds!**



do a

“birthday attack”

against the

*nonce* value

**Start with the Taylor series approximation to the probability of a “nonce” value collision where “n” is the number of attempts and “H” is the number of unique outputs:**



$$p(n) = 1 - \bar{p}(n) \approx 1 - e^{-(n(n-1))/2 \cdot H} \approx 1 - e^{-n^2/2 \cdot H},$$

**Invert the expression:**

$$n(p) \approx \sqrt{2 \cdot H \cdot \ln \left( \frac{1}{1-p} \right)},$$

**Now assigning a 0.5 probability of collision:**

$$n(0.5) = 1.1774\sqrt{H}$$



# 301 attempts against $2^{16}$ hash



# secure zone transfers



# (mis?)using DNS



# TCP-over-DNS



# dynamic DNS



# SPF



# useful tools

# nslookup

# useful tools



# whois

# useful tools



# dig



# DNS server software



# authoritative and recursive: BIND, MaraDNS



# authoritative: MyDNS, tinydns



# recursive: dnscache



# master vs slave

# firewall issues



## port 53 UDP *and* TCP



# Introduction to DNS



**Thankyou :-)**  
**questions?**

Slides: [jon.oxer.com.au/talks](http://jon.oxer.com.au/talks)

Contact: Jonathan Oxer <[jon@ivt.com.au](mailto:jon@ivt.com.au)>

We're hiring! [www.ivt.com.au/jobs](http://www.ivt.com.au/jobs)