

How DNS Works?

- Namespace is organized as a tree
- A *name server* controls one *zone* – a subtree of this tree
 - He either contains all *resource records* for nodes in this subtree
 - Or one of his offspring does
 - This is an administrative division, one domain (udel.edu) can have many zones
- *Clients* ask for the data they need
- *Resolvers* find out the data that clients need
 - They have it in their zone file (this machine is the name server)
 - Or they have it in their cache
 - Or they know who to ask and they will find out the answer

How DNS Works?

- Each record has time-to-live (TTL) in seconds – indicates how long can a record stay in the cache
 - This is done to provide for adaptation in a fast changing environment
- Each zone is required to have at least two name servers
 - Primary
 - Secondary (backup)
- Information is kept on the primary name server and periodically backed up on a secondary name server
- Inverse queries are satisfied from a dedicated domain in-addr.arpa ○

Misusing Trust Relationship

- Applications perform authentications based on DNS names, not addresses
 - This is not so frequent today, as they use public keys instead, but it used to be common 5-10 years ago
 - When such an application receives the request for connection it checks inverse address-to-name mapping
 - Only one line in in-addr.arpa holds this and, if changed, attacker can gain trusted access to the given application

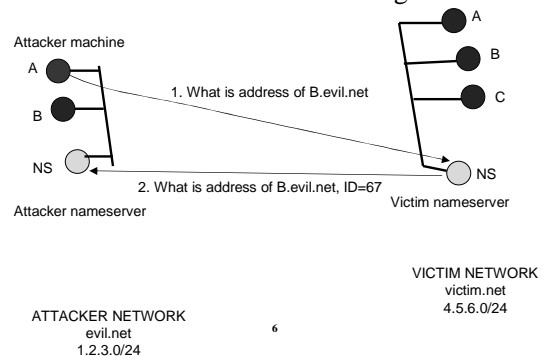
DNS Cache Poisoning

- There is no authentication method to assure that the reply we got for DNS query is:
 - Correct
 - Generated by a name server authoritative for the zone
 - Therefore anyone can generate a reply?
 - DNS request/reply packets have an ID number that is used for matching replies to requests
 - Anyone who wants to spoof replies must first guess the appropriate ID number
 - Sometimes this can be very easy to do

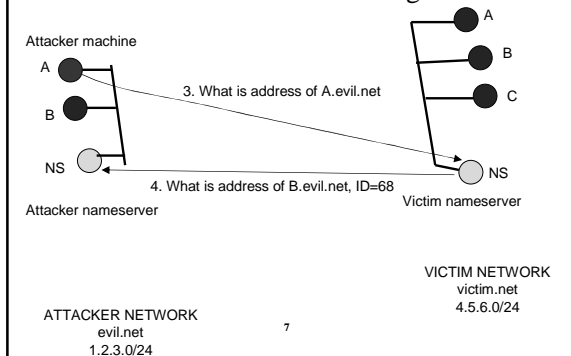
DNS Cache Poisoning

- The scope of the damage:
 - Denial of service for the client
 - Redirection of traffic for the client
 - DoS and/or redirection for the whole network if a cache of a server doing recursive lookup were infected

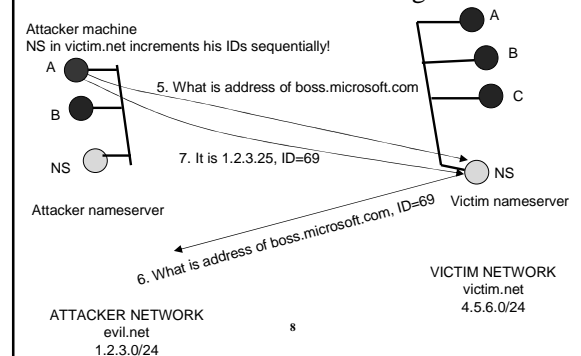
DNS Cache Poisoning



DNS Cache Poisoning



DNS Cache Poisoning



DNS Cache Poisoning

- OK, so one network cannot reach boss.microsoft.com
 - This is bad but still, there are many addresses that this network can reach, and this is just one network
- How about poisoning everyone's cache so they cannot reach any of the microsoft.com addresses?
- This can be done if microsoft.com has a secondary server as a public one
 - Secondary server updates his database from the primary server (likely inside microsoft.com domain) via *zone transfer* messages
 - Format of these messages is similar to DNS request/response format and can as easily be faked
 - Now everyone looking for microsoft.com will get fake data

9

Bandwidth Attacks Using DNS

- DNS replies are sometimes much larger than DNS requests (1:3 size difference)
- This offers potential for *amplification* denial-of-service attacks
 - Attacker needs to send very few small requests, spoofing victim's address
 - Victim will be overwhelmed with replies
- How is this attack different than Smurf?

10

Attacks on DNS Root Servers

- Concerted ping flood attack on all 13 of the DNS root servers in October 2002
- Successfully halted operations on 9 of them
- Lasted for 1 hour, turned itself off
- Appears to have been the work of experts
- Did not cause major impact on Internet
 - DNS uses caching aggressively
 - Several root servers were provisioned enough
- Longer, stronger attacks might have succeeded
- The perpetrator of this attack is still unknown

11

Attacks on DNS Name Servers

- Attack on Microsoft DNS servers in January 2001
 - Attacked router in front of Microsoft's DNS servers
 - During attack, as few as 2% of web page requests were being fulfilled
 - As opposed to 97%, under normal load
- Solved by a better configuration of Microsoft's DNS servers

12

Securing Name Servers

- Root nameservers hold data that changes very slowly
 - Every week or so a portion of it will change
 - And root servers don't have that much data anyways
- It is possible to replicate this data and only check occasionally to see whether it has changed
- It is also possible to have more than 13 root DNS servers
- Or to filter ICMP traffic for root DNS servers
 - Attacker can still flood with DNS requests
- Securing TLDs is harder since data changes frequently and there is much more of it

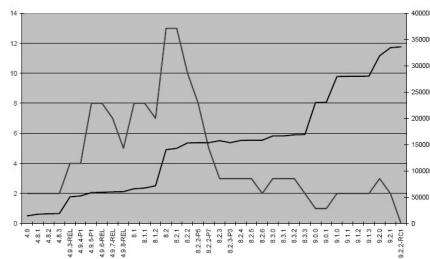
BIND Vulnerabilities

- BIND is the prevalent implementation of nameserver functionality
- Homogeneous nameservers are vulnerable
 - Successful attack can take them all down
- BIND is an open source software and has had many versions
 - Each version has fixed some vulnerabilities and introduced new ones

"Bound by Tradition: A Sampling of the Security Posture of the Internet's DNS Servers"
 Mike Schiffman mike@infonexus.com, February 2003

BIND Vulnerabilities

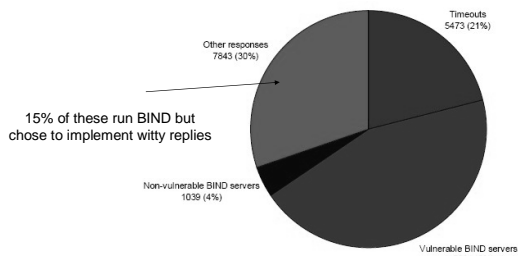
Chart 3: BIND lines of source per version with vulnerabilities 2003.01.05



BIND Vulnerabilities

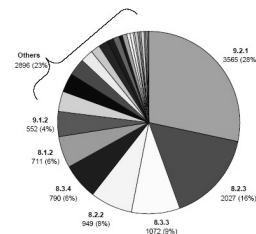
- 18 publicly disclosed bugs:
 - 9 of them enable remote code execution
 - 8 enable denial-of-service attacks
 - 1 enables information disclosure

BIND Prevalence



BIND Version

Chart 5: BIND server distribution across the Internet 2003.01.05



BIND Version for Root DNS

Table 1: Reported root name server DNS versions
2003.01.05

Root Name Server	Location	IP address	DNS Version
A.ROOT-SERVERS.NET	VA, US	198.41.0.4	VGSI
B.ROOT-SERVERS.NET	CA, US	128.9.0.307	8.3.3.REL
C.ROOT-SERVERS.NET	VA, US	192.213.4.12	8.3.3.REL
D.ROOT-SERVERS.NET	ME, USA	188.8.0.96	8.3.3.REL
E.ROOT-SERVERS.NET	CA, USA	192.203.230.10	8.3.3.REL
F.ROOT-SERVERS.NET	CA, US	192.5.5.241	9.2.2rel
G.ROOT-SERVERS.NET	VA, US	192.112.36.4	server <i>indef</i>
H.ROOT-SERVERS.NET	MD, US	128.63.2.53	9.2.2rel
I.ROOT-SERVERS.NET	Stockholm, SE	192.36.148.17	8.3.3.REL
J.ROOT-SERVERS.NET	VA, US	188.41.0.98	VGSI
K.ROOT-SERVERS.NET	London, UK	193.0.14.129	8.3.3.REL-HCESW
L.ROOT-SERVERS.NET	CA, USA	198.32.64.12	BIND-3.1.1-MA-PATCH-01
M.ROOT-SERVERS.NET	Tokyo, JP	202.112.27.33	8.3.4.REL

Other Name Server Implementations

- Dan Bernstein's tinydns
 - It was released in 1999
 - No publicly reported vulnerabilities
 - There is even an award of \$500 for the first vulnerability found

DNSSEC

- A modification of DNS protocol aimed at solving two problems:
 - Data integrity – noone modified this data
 - Data origin authentication – this data was generated by an authority responsible for the given zone
- No protection is provided against denial-of-service
- Worse, some amplification messages are added to make this problem more aggravating

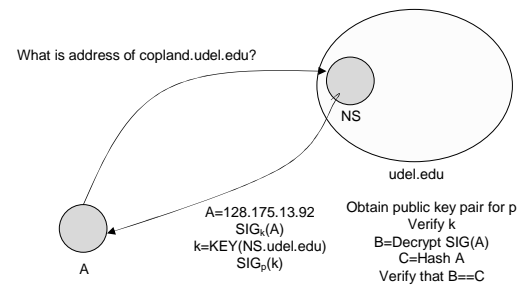
DNSSEC

- Use digital signatures to sign the data and guarantee integrity and authenticity
 - Authoritative name server will sign the data with his private key
 - Anyone can decrypt and verify using server's public key
 - If the private key is not kept online (on the name server) then there is no danger even if the server is compromised
- Let us remind ourselves how are signing and verification done!

DNSSEC

- Each zone will have a public/private key pair
 - We will sign each record in a zone file with the private key
 - This signature will form a new record – SIG record
 - Zone's public key is stored in another record – KEY record
- What if someone wanted to fake zone's public key?
 - This is prevented by adding a parent signature for KEY record

DNSSEC Example



Proving Non-Existence

- Usually, if something does not exist a generic NOEXIST record is returned
 - This is easily spoofed
 - Even signatures don't help (why?)
 - What if someone claims that a real record does not exist?
- We must sign something
 - We will sign the NXT record – this specifies which existing name is lexicographically after the one that does not exist
 - We will sort names, not numbers

Sorting

```

udel.edu. IN SOA ns.udel.edu. root.udel.edu. (99021800 1h 10m 30d 1d )
          IN NS ns.udel.edu.
          IN A 128.175.0.1
          IN MX 0 mail.udel.edu.
ns       IN A 128.175.0.1
mail     IN A 128.175.0.2
www      IN A 128.175.0.3
ftp      IN CNAME www.udel.edu.

Sorts to
udel.edu. IN SOA ns.udel.edu. root.udel.edu. (99021800 1h 10m 30d 1d )
          IN NS ns.udel.edu.
          IN A 128.175.0.1
          IN MX 0 mail.udel.edu.
ftp.udel.edu. IN CNAME www.udel.edu.
mail.udel.edu. IN A 128.175.0.2
ns.udel.edu. IN A 128.175.0.1
www.udel.edu. IN A 128.175.0.3
    
```

NXT record

```

udel.edu. IN SOA ns.udel.edu. root.udel.edu. (99021800 1h 10m 30d 1d )
udel.edu. IN NS ns.udel.edu.
udel.edu. IN A 128.175.0.1
udel.edu. IN MX 0 mail.udel.edu.
udel.edu. IN NXT ftp.udel.edu A NS SOA MX NXT
ftp.udel.edu. IN CNAME www.udel.edu.
ftp.udel.edu. IN NXT mail.udel.edu. CNAME NXT
mail.udel.edu. IN A 128.175.0.2
mail.udel.edu. IN NXT ns.udel.edu. A NXT
ns.udel.edu. IN A 128.175.0.1
ns.udel.edu. IN NXT www.udel.edu A NXT
www.udel.edu. IN A 128.175.0.3
www.udel.edu. IN NXT udel.edu. A NXT
    
```

Now if someone asks for morning.udel.edu
we would return signed
mail.udel.edu. IN NXT ns.udel.edu. A NXT

Inverse Address Mapping

