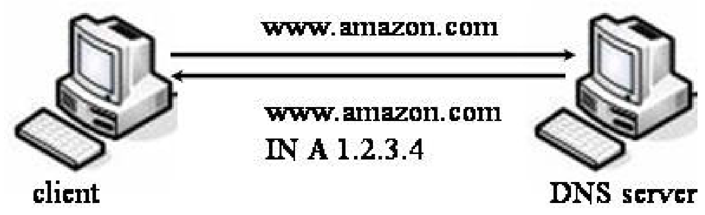
CS 261 Computer Security
Prof. David Wagner

Fall 2007
Scribe: Karen Hsu

DNS Security / October 23, 2007

1 How DNS Works

- DNS attacks are critical to internet security
- Forward mapping is the process of determining the corresponding IP address of a given hostname
- Reverse mapping is the process of determining the corresponding hostname of a given IP address
- These 2 mappings might not have any relation to each other, up to admin



Say you want to talk to **amazon.com** and you perform a DNS lookup. If the DNS server knows the answer, it will respond with an address record. Otherwise, it will tell you where to look (hardcoded list ~13-15 DNS root servers that know how to resolve **.com**, **.org**, **.net**, etc. Query the root server to know who to talk to about **.com** and the root will tell you ".com is served by [name server] + [address of that name server]")

```
.com IN NS ns.com
ns.com IN A 2.4.5.6
.amazon.com IN NS dns.amazon.com
dns.amazon.com IN A 6.7.8.9
```

- DNS server likely has a cache for improved performance
- What about UDP? There's a16-bit ID number in query; when server responds, includes that ID number.

2 Attacks

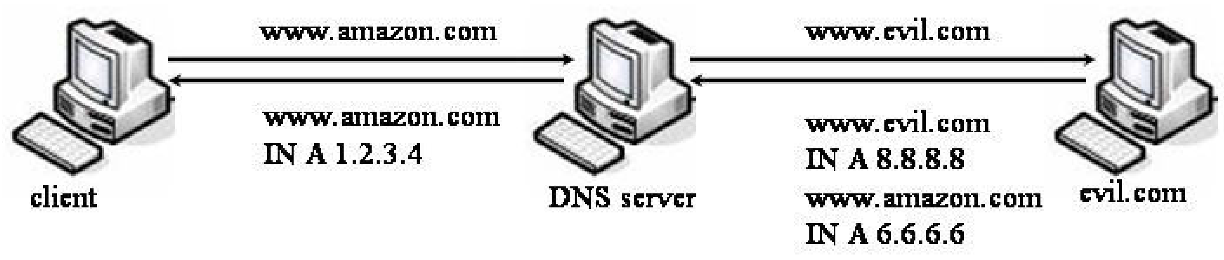
2.1 Poor Man's Attack

An attacker can run his or her own DNS server and arrange with ISP so that it would host queries for your IP address. The attacker can ask his or her DNS server to lie; when he or she attempts to rlogin, it will try to validate the name (not the IP address) of the calling machine. The attacker can simply claim to be someone else on the trusted list.

DEFENSE
Do a forward lookup - server can check consistency by doing both a forward & backward look up. This works because the attacker controls only one of the two mappings.

2.2 Cache pollution

Cache pollution arises because DNS servers cache responses they've seen from other queries for performance. Cache pollution tricks a DNS server to cache an erroneous record.



An attacker can make a bogus query to figure out who is authoritative for **evil.com**. The **evil.com** nameserver will be contacted, and because the attacker controls **evil.com** he or she can return an address for **evil.com** + a blatantly irrelevant record that provides an address for **amazon.com**.

DEFENSE
Why can't I just disallow the additional acceptance of records? DNS will be messed up – can't get bootstrapped, functionality issue. Moreover, this would hurt caching and performance. Clients would have to make extra queries.

Two solutions easily retrofit-able:
SOLUTION 1: Cache additional records, but only return them in response to similar queries. In our example, in response to a query for **amazon.com**, don't return the cached query because the cache is associated with the query and not **amazon.com**. This solution hurts performance, but adds security.

SOLUTION 2: Keep track of where you got each record from and who is authoritative for each record. It's hard to reconcile with having a caching nameserver; if you accept a response for **eecs** from **amazon**… is **amazon** authoritative for **eecs**? no. tricky!

2.3 Hack the DNS Server

Using whatever vulnerability, you can rewrite DNS tables. Trivial for discussion.

2.4 Transitive Trust

DNS record snippets:

```
berkeley.edu IN NS adns1.berkeley.edu
berkeley.edu IN NS phloem.uoregon.edu
uoregon.edu IN NS arizona.edu
arizona.edu IN NS pendragon.cs.purdue.edu
purdue.edu IN NS nsl.rice.edu
```

What are the security implications of these records?
Transitive dependencies: If an attacker hacks any one of these nameservers (on the right-hand side), they win. The attacker can DoS everyone in the chain so that the servers are forced to rely on their fallback (eventually to one the attacker has compromised). This vulnerability is prevalent.

2.5 Wireless DNS Attack

DHCP tells you which DNS server to use. If the access point is malicious, it can point you to a malicious DNS server that lies.

For example, you can type in gmail.com and when the lookup is performed you might be sent to a bogus website:

SCENARIO 1:

https://gmail.com	Certificate warning? Decline!
http://gmail.com	No SSL → no certificate. Can be spoofed!

SCENARIO 2:

Go to http://www.amazon.com
When you click "buy," you get a page requesting your user and password. That link should be to https/SSL.

An attacker can
(1) set up a fake website that does not use SSL
(2) set up his or her own domain name that looks like **amazon.com**.

DNS spoofing attack is hard to detect.
The attack relies on setting up a malicious AP and configuring the DHCP server to direct you to there.

DEFENSE
Type in https yourself.

3 Attack Prevalence

X	Poor Man's Attack	Not Real – pretty much cured
√	Cache Pollution	Real – You'll find occasional bind servers w/o defenses against this type of attack, but most have upgraded. i.e. Attacker caused www.InterNIC.net to respond to his own IP address and put up a vandalized front page. Fled to Canada and went to jail.
√	Hack the DNS Server	Real – more common than you'd expect. 17% DNS servers vulnerable
√	Transitive Trust	Real and current – 45% Domains affected (from 17% DNS servers vulnerable) 46 Servers
√	Wireless DNS Attack	Very Real – you see this frequently. i.e. virus that propagated by enabling machine to act as an access point

4 Other Attacks and Observations

- An attacker can use brute force to guess the 16-bit ID in the responses to send forged replies. This may take many, many tries.. but is doable! Otherwise, if an attacker obtains 2 successive rand values, he or she can guess the internal state. Rand sucks!

  DEFENSE
  Use unpredictable IDs and unpredictable source ports, since the attacker would need to guess both.

- SSL link not enough; the server you're talking to might be legitimate, but it might also be lying.

  DEFENSE
  Public key / private key signatures

  Augment DNS: Each record is also signed.

  $$[\text{www.amazon.com IN A 1.2.3.4}]_{k_{amazon}}$$

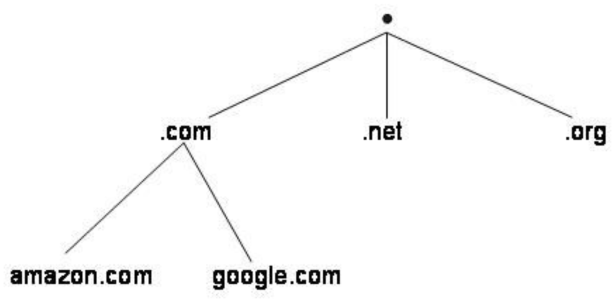
  Now you don't have to worry about the cache being polluted, because the record is signed directly by Amazon. This adds lots of extra complexity.

  ISSUES
  Key management: When I get a signed record, I'm going to need to check the validity of the signature using Amazon's public key. How do I get this? If I don't know its IP address, it's unlikely that I'd know its public key. So, what to do? Get public key from DNS:

  ```
  Amazon.com IN KEY 0xDEADBEEF…
  ```

  Here, you're relying on DNS again and you're therefore back to step one. DNSSec says this should be signed with the private key of .com.

  $$[\text{www.amazon.com IN KEY 0xDEA...}]_{k_{com}}$$



That way, the .com registry is responsible for maintaining those who are registered as well as their public keys.

$$[\text{com IN KEY 0x1234FC...}]_k$$

The root key must be carefully protected – if it is compromised, then you are incredibly hosed. To reduce this risk, one option would be to perform signing offline – the signing machine is never connected to the network and signatures are then carried to the DNS server. Records can be signed in advance.

- If you do a DNS query for foo.cs.berkeley.edu and there is no host by that name, you will get a negative response. Turns out, negative responses can be forged.

5 Takeaways

- Distributed systems are hard to secure, especially if some nodes are malicious
- DNS security issues are so problematic. Naming can be very damaging to security because it affects direct connection - it's the way we translate human intent into something computers can act on.
- Threat models:
    - Cryptographer's threat model – the entire network is the adversary; everything is insecure. You can assume malicious routers spoof DNS.
    - Pragmatic threat model – the attacker controls some nodes of his or her own (spoof packets). The attacker can break in if routers are compromised, but otherwise pretty hard.
- Note: For broken systems in the cryptographer's threat model, years later you may find you have security weaknesses in the pragmatic threat model.