**DNS fundamentals**

1. In Linux, run the commands hostname and nm-tool and investigate the output.

In Windows, run the command ipconfig /all and investigate the output.

• nm-tool output: What does the Address field tell us?

The Address field tells us our current internal IP-address if this device. The internal IP is an IP assign for this device, which is connected to a router. Conclusion internal IP is for the router to identify the device.

• nm-tool output: What does the Prefix field tell us?

The prefix tells us, that the 24 first bits of IP address indicates the network and subnet. The subnet mask is 255.255.255.0

• nm-tool output: What does the Gateway field tell us?

The gateway tells us the IP address of our current host router.

• nm-tool output: What does the DNS fields tell us?

The DNS field tells us which IP addresses we connect and make a lookup, when we will access the Internet.

1. In Linux or Windows, ping a webserver

• Try browsing the net by IP number and name using your browsers address field

After making a ping on google.com, we get an IP address 173.194.65.101.

We can now access google.com by typing 173.194.65.101 instead of google.com

1. Host Lookup Table (HLT)

• Edit your HLT and redirect the name of a web server to IP address of another

By editing in Host.txt, we can map loogle to 173.194.65.101, if we ping loogle, we will actually ping 173.194.65.101.

• Is the HLT looked through before your primary DNS server is queried?

The host file is looked through before the DNS server.

• Who first proposed the HLT?

It was Peggy Karp.

• When was the HLT proposed?

In 1921-1972.

1. What does TLD stand for?

TLD stands for “Top Level Domain”. TLD is like .com (dot com).

The TLD names are installed in the root-zone of the name space.

1. Why does a FQDN end with a dot, “.”?

FQDN is referred as absolute domain name, is a domain name that specifies the exact location in the tree hierarchy of the DNS.

The dot is there to distinguish between relative and absolute domain names.

Ex. <http://itonk.iha.dk> - itonk is the hostname in the domain iha.dk. the .dk is the TLD.

1. What is a DNS "A" record and what is it used for?

An A record is used to control the location of the resources on the internet. In short I maps the logical domain such as google.com to google’s hosting server IP address.

1. What is a DNS zone and what it is used for?

A **DNS zone** is a portion of a domain name space using the [Domain Name System](http://en.wikipedia.org/wiki/Domain_Name_System) (DNS) for which administrative responsibility has been delegated.

**Name resolution**

1. What is recursive and iterative name resolution?

Recursive name resolution is the process by which a DNS server uses the hierarchy of zones and delegations to respond to queries for which it is not authoritative.

An *iterative name query* is one in which a DNS client allows the DNS server to return the best answer it can give based on its cache or zone data. If the queried DNS server does not have an exact match for the queried name, the best possible information it can return is a*referral* (that is, a pointer to a DNS server authoritative for a lower level of the domain namespace). The DNS client can then query the DNS server for which it obtained a referral. It continues this process until it locates a DNS server that is authoritative for the queried name, or until an error or time-out condition is met.

1. How may DNS caching be more effective in recursive vs. iterative resolution?
2. Why are client-side communication costs reduced in recursive name resolution?
3. Why does recursive resolution put higher performance demands on each name server?

The iterative resolution

The client first ask the root server, gets a answer and then ask the next node in the DNS hierachy, gets the answer and so on down to the bottom of the hierachy.

In this case the client is making most of the work and therefore there is much communication, between the client and all the different name servers.

The recursive resolution

When using the recursive mode the client ask the root name server and then make this one do the work. The root server then asks the nodes underneath it, receives the answers from them and return the answer to the client server.

Using this method there is much less communication between the client and the name servers and therefore is the recursive name resolution faster.

But the downside with recursive resolution is you make the name servers do all the work. That is a problem in the top off the root, where there name serves will have to handle a lot of requests at once.

Often recursive and iterative requests can be combined in a single resolution, providing significant flexibility to the process as a whole.

**DNS security extensions**

1. What is DNSSEC and why is it needed?

The **Domain Name System Security Extensions** (**DNSSEC**) is a suite of [Internet Engineering Task Force](http://en.wikipedia.org/wiki/Internet_Engineering_Task_Force) (IETF) specifications for securing certain kinds of information provided by the [Domain Name System](http://en.wikipedia.org/wiki/Domain_Name_System) (DNS) as used on [Internet Protocol](http://en.wikipedia.org/wiki/Internet_Protocol) (IP) networks. It is a set of extensions to DNS which provide to DNS clients (resolvers) origin authentication of DNS data, authenticated denial of existence, and data integrity, but not availability or confidentiality.

1. What is a signed zone?

To limit replay attacks, there are not only the normal DNS TTL values for caching purposes, but additional timestamps in RRSIG records to limit the validity of a signature. Unlike TTL values which are relative to when the records were sent, the timestamps are absolute. This means that all security-aware DNS resolvers must have clocks that are fairly closely in sync, say to within a few minutes.

These timestamps imply that a zone must regularly be re-signed and re-distributed to secondary servers, or the signatures will be rejected by validating resolvers.

3. What common types of security vulnerabilities are hindered by DNSSEC?



**BIND DNS server**

1. Install BIND on a Linux machine and check the installation

2. Configure a *caching name server* and *forwarder*

3. Use Google’s Name Bench to find a suitable public DNS server to forward to

4. Test and document that forwarding works as intended

5. Test and document whether the DNS lookup time is reduced by caching

**Prototype**

* Consider a concrete case, e.g. home, school, office, or hospital, where employing a caching name server and forwarder serves a realistic purpose
* How would you set up BIND in your case?
* What functionalities in BIND would you use?  **Project report**  In your team, write a project report that 1. Includes answers to all exercises 2. Incorporate the answers in a natural way into your report text 3. Adheres to the report template on CampusNet