DNS Spoofing Attack

Dr. Neminath Hubballi

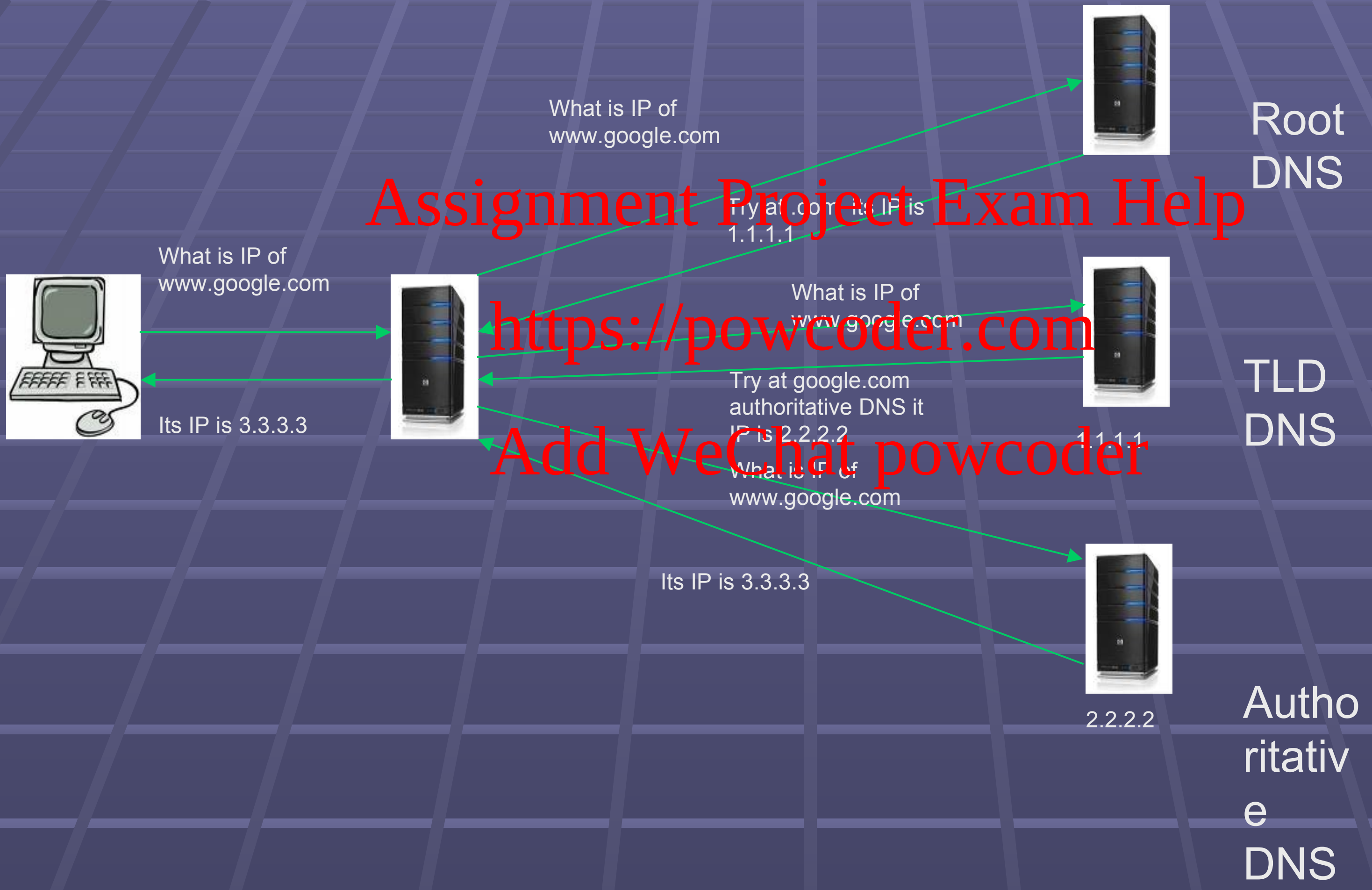
DNS Basics

- We are not good at remembering numbers
- Computers work with numbers
- Mapping between IP addresses and URLs is maintained as a service
- DNS servers does this job of transforming between these two
- Historically the work done by DNS servers was done with hosts.txt
- Every host maintains a list of mapping IP addresses and computer names
    - Was feasible in ARPANET time
    - Scalability became an issue

DNS

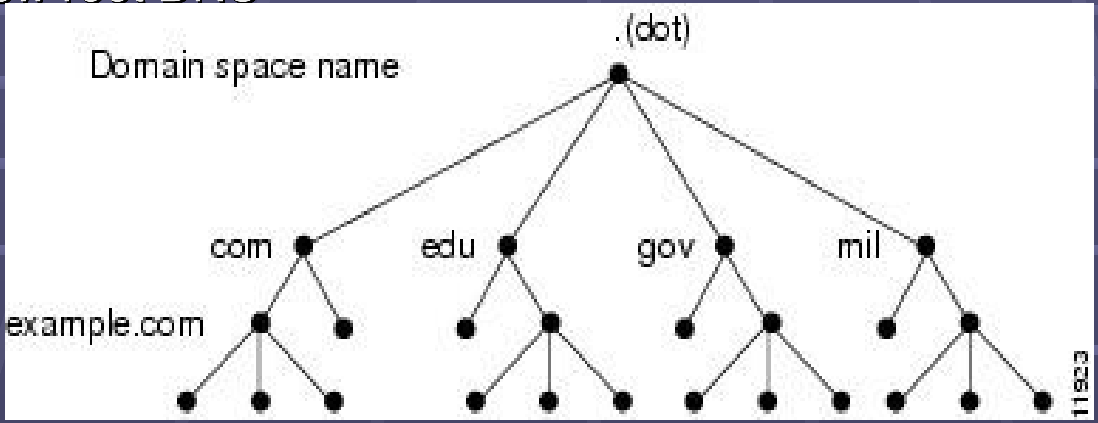
- DNS runs on port 53
- Runs on UDP
- UDP is a connectionless protocol
  - Makes it easy for spoofing
- DNS is a distributed database maintained in a hierarchical tree structure
- DNS Cache
  - To improve operational efficiency DNS servers caches the resource records
  - Positive caching
  - Negative caching

DNS Working


What is IP of www.google.com
Its IP is 3.3.3.3
What is IP of www.google.com
Try at .com its IP is 1.1.1.1
Root DNS
What is IP of www.google.com
Try at google.com authoritative DNS it IP is 2.2.2.2
1.1.1.1
TLD DNS
What is IP of www.google.com
Its IP is 3.3.3.3
2.2.2.2
Authoritative DNS

DNS Components

- Resource Records
- Internet Domain Namespace
  - Organizational
  - Geographical
  - Reverse domain
- Root DNS is at the top
- Root DNS is managed by Internet Name Registration Authority
- Top Level Domain (TLD)
  - Bellow root DNS



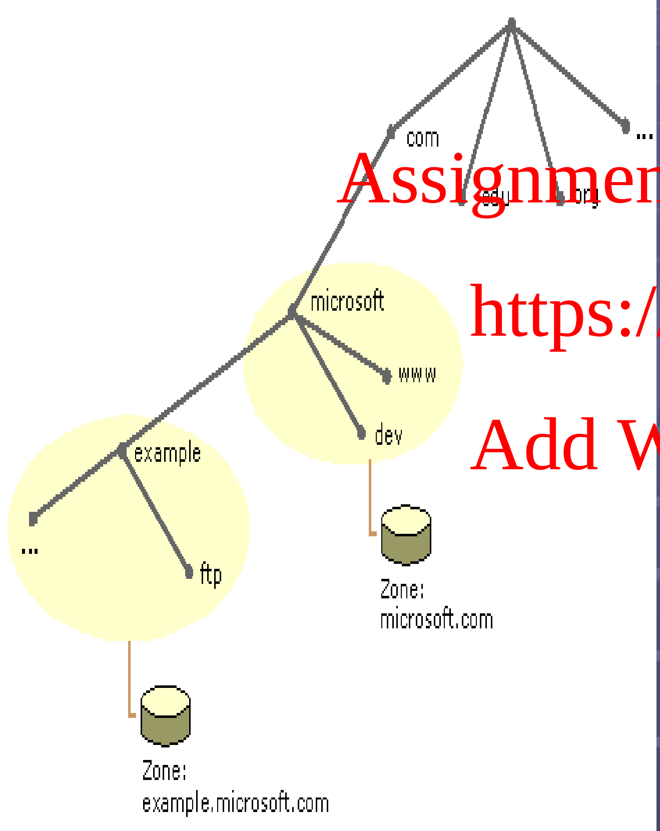
Record Types in DNS

- Important ones as there are many
  - A –Address record   name to 32 bit address
  - AAAA – Address Record name to 128 bit IPV6 address
  - CNAME – Canonical name after receiving this reply host will query with this new request

NAME	TYPE	VALUE
bar.example.com.	CNAME	foo.example.com.
foo.example.com.	A	92.0.2.23

  - NS Records – Contain IP address of authoritative name server

Zones in DNS



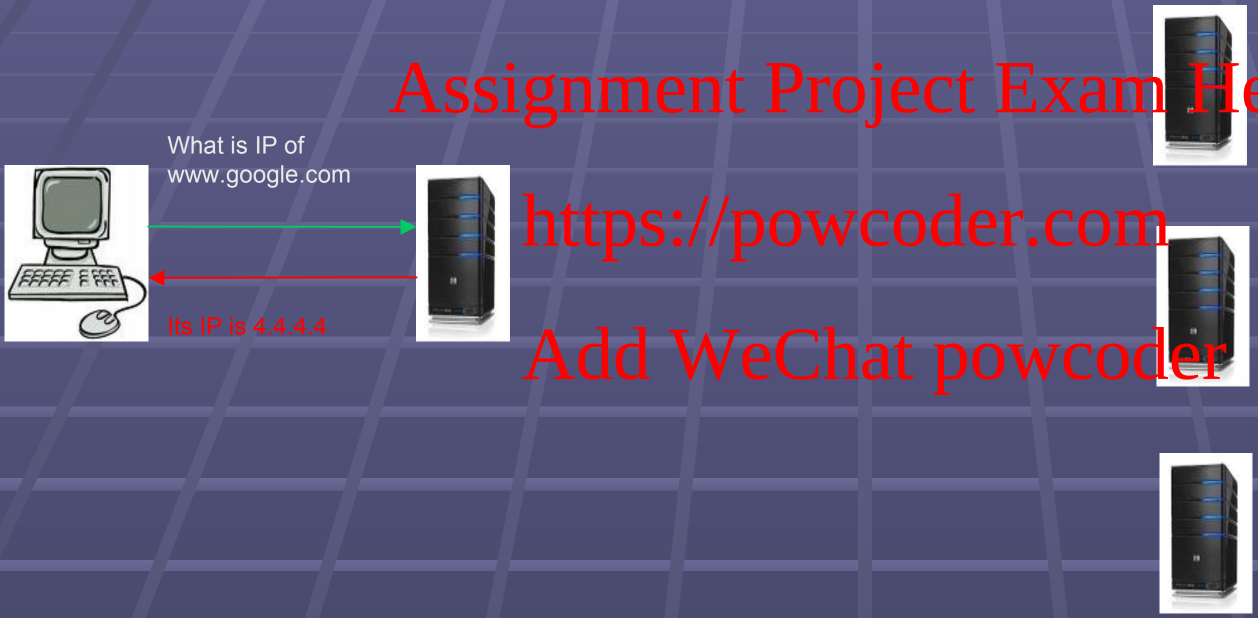
- .com is domain
- Microsoft.com is a zone
- Zone starts as a database of single domain
- If other domains are added below the domain used to create the zone
  - Subdomains can be part of same zone
    - Dev.microsoft.com
  - Belong to another zone
    - Example.microsoft.com
- Zone is a subset of domain

Zone Transfer

- When a new DNS server is added
  - For high availability and fault tolerance reasons
- It starts as a secondary DNS server
  - All zones hosted in primary are copied to secondary

DNS Vulnerability

- Getting a wrong answer from the server



What is IP of www.google.com

Its IP is 4.4.4.4

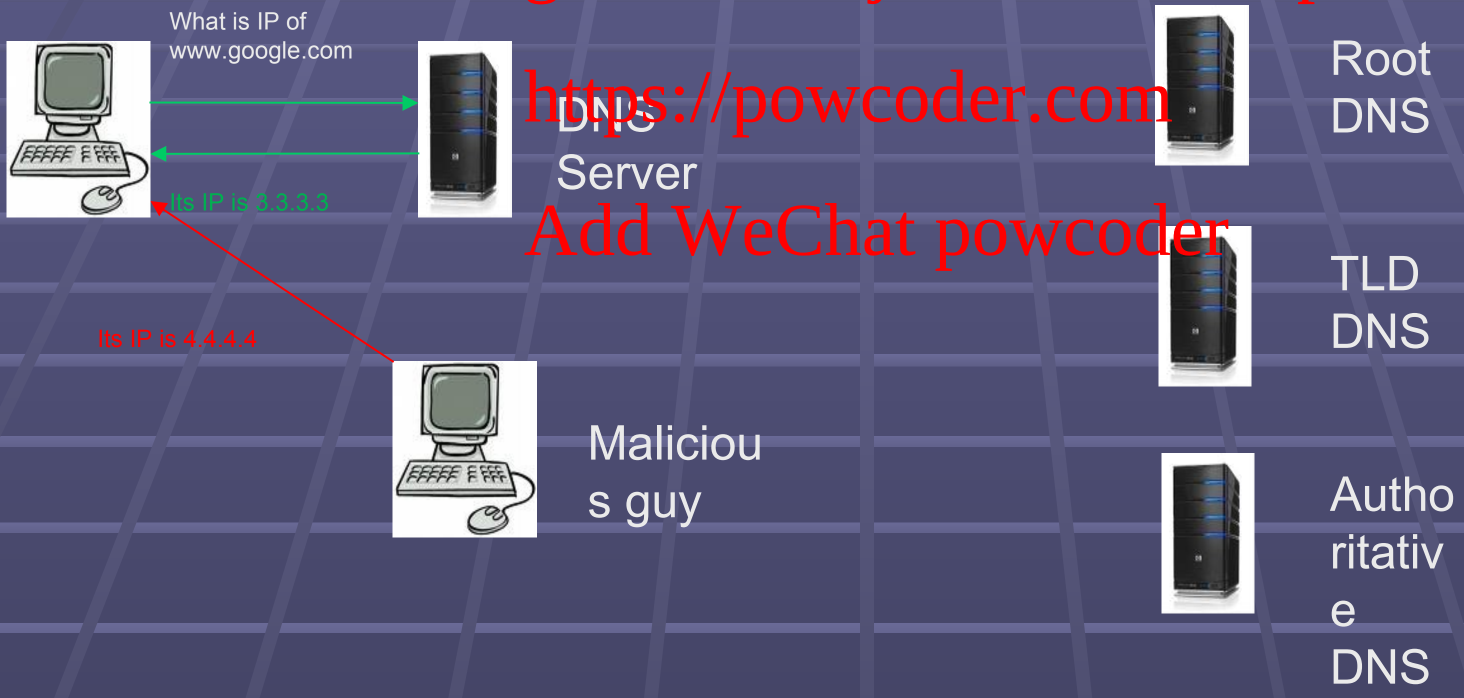
Root DNS

TLD DNS

Authoritative DNS

DNS Vulnerability

- Someone else answers to a DNS query before the one supposed to answer



What is IP of www.google.com

Its IP is 3.3.3.3

Its IP is 4.4.4.4

DNS Server

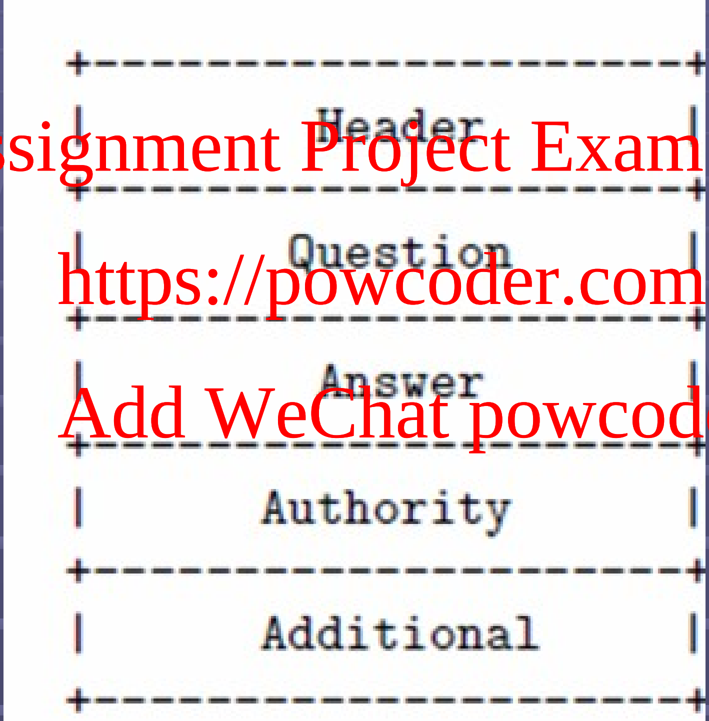
Malicious guy

Root DNS

TLD DNS

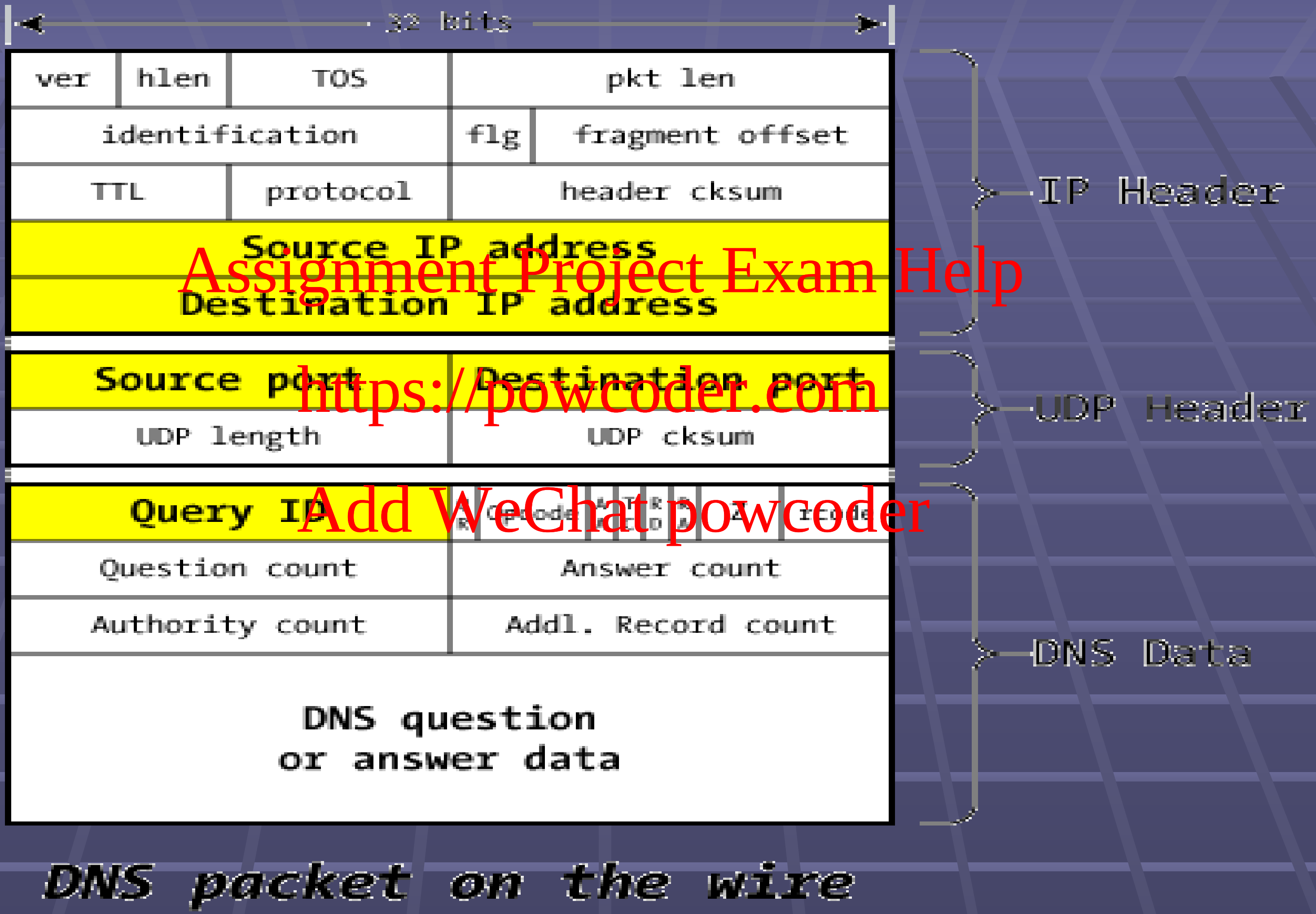
Authoritative DNS

DNS Packet Structure



```
+---------------------+
|        Header       |
+---------------------+
|       Question      |
+---------------------+
|        Answer       |
+---------------------+
|      Authority      |
+---------------------+
|      Additional     |
+---------------------+
```

DNS Packet Structure



ver | hlen | TOS | pkt len

identification | flg | fragment offset

TTL | protocol | header cksum

Source IP address

Destination IP address

Source port | Destination port

UDP length | UDP cksum

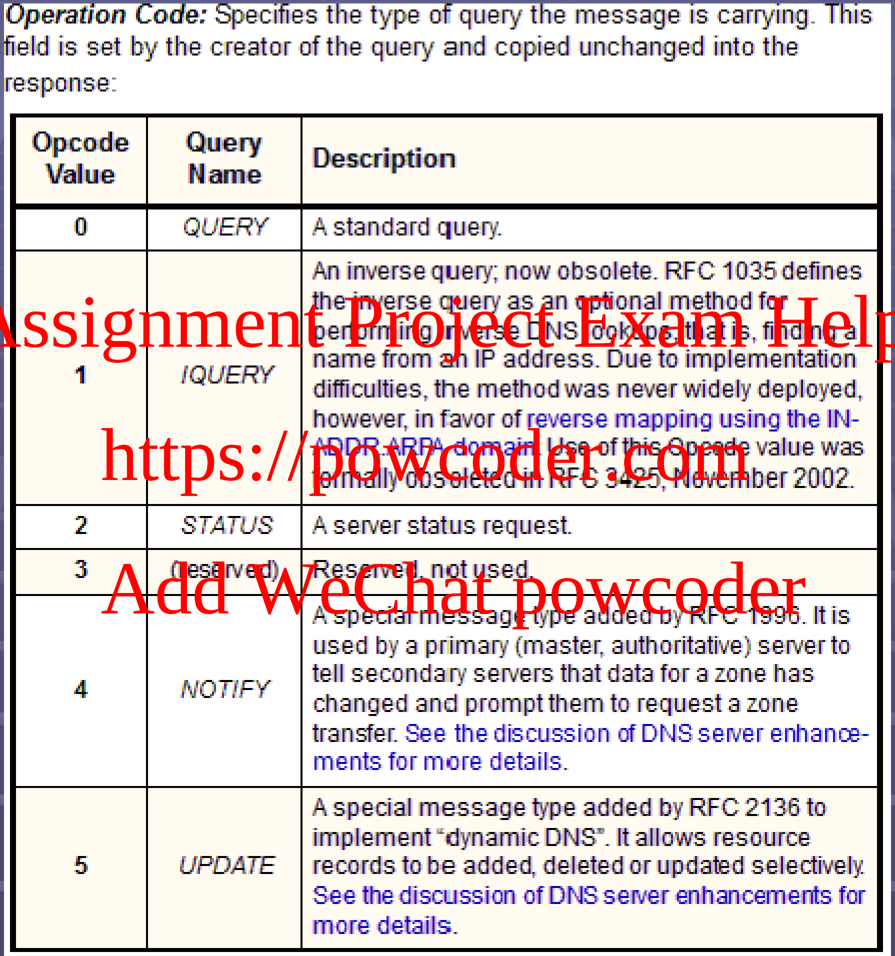
Query ID | Q R | Opcode | A A | T C | R D | R A | Z | rcode

Question count | Answer count

Authority count | Addl. Record count

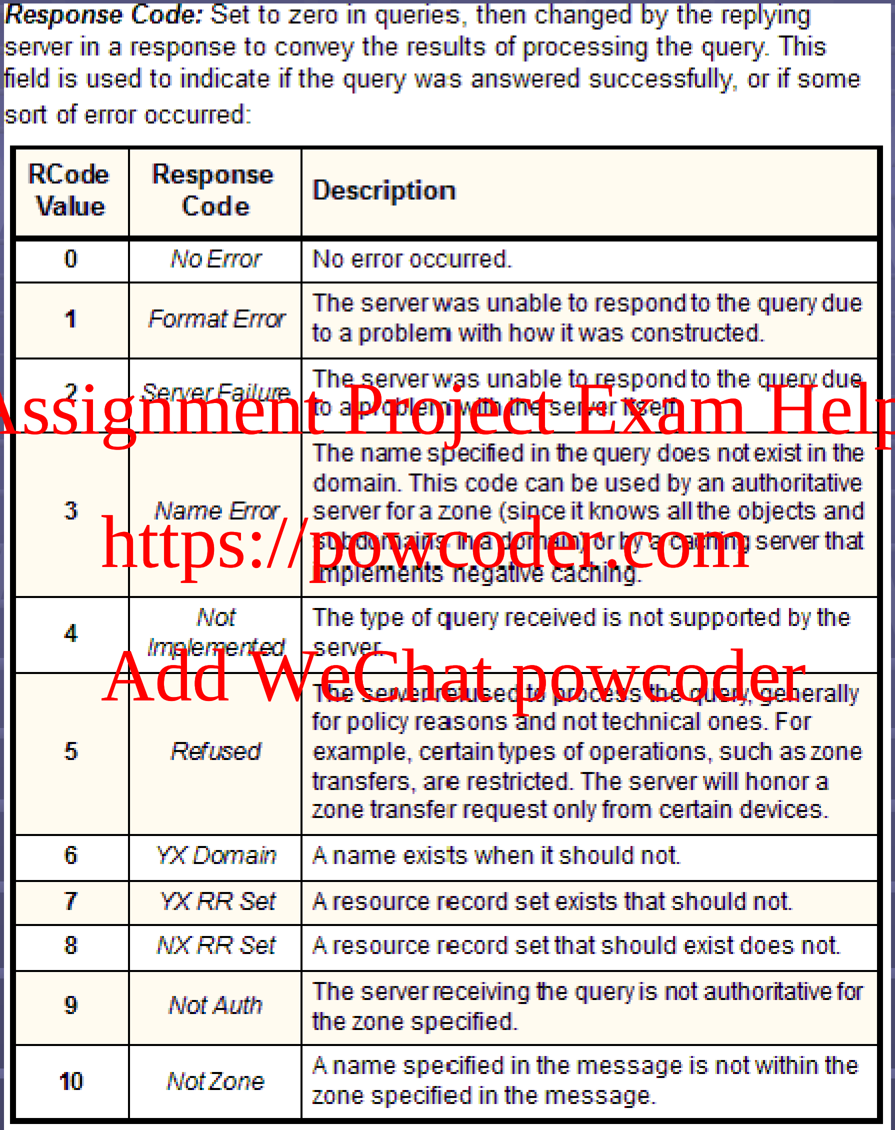
DNS question or answer data

DNS packet on the wire



Operation Code: Specifies the type of query the message is carrying. This field is set by the creator of the query and copied unchanged into the response:

Opcode Value	Query Name	Description
0	*QUERY*	A standard query.
1	*IQUERY*	An inverse query; now obsolete. RFC 1035 defines the inverse query as an optional method for performing inverse DNS lookups, that is, finding a name from an IP address. Due to implementation difficulties, the method was never widely deployed, however, in favor of reverse mapping using the IN-ADDR.ARPA domain. Use of this Opcode value was formally obsoleted in RFC 3425, November 2002.
2	*STATUS*	A server status request.
3	(reserved)	Reserved, not used.
4	*NOTIFY*	A special message type added by RFC 1996. It is used by a primary (master, authoritative) server to tell secondary servers that data for a zone has changed and prompt them to request a zone transfer. See the discussion of DNS server enhancements for more details.
5	*UPDATE*	A special message type added by RFC 2136 to implement "dynamic DNS". It allows resource records to be added, deleted or updated selectively. See the discussion of DNS server enhancements for more details.



Response Code: Set to zero in queries, then changed by the replying server in a response to convey the results of processing the query. This field is used to indicate if the query was answered successfully, or if some sort of error occurred:

RCode Value	Response Code	Description
0	*No Error*	No error occurred.
1	*Format Error*	The server was unable to respond to the query due to a problem with how it was constructed.
2	*Server Failure*	The server was unable to respond to the query due to a problem with the server itself.
3	*Name Error*	The name specified in the query does not exist in the domain. This code can be used by an authoritative server for a zone (since it knows all the objects and subdomains in a domain) or by a caching server that implements negative caching.
4	*Not Implemented*	The type of query received is not supported by the server.
5	*Refused*	The server refused to process the query, generally for policy reasons and not technical ones. For example, certain types of operations, such as zone transfers, are restricted. The server will honor a zone transfer request only from certain devices.
6	*YX Domain*	A name exists when it should not.
7	*YX RR Set*	A resource record set exists that should not.
8	*NX RR Set*	A resource record set that should exist does not.
9	*Not Auth*	The server receiving the query is not authoritative for the zone specified.
10	*Not Zone*	A name specified in the message is not within the zone specified in the message.

DNS Poisoning with Host.txt

- On a windows machine
  - Open C:\windows\system32\drivers\etc\host.txt
  - Add a line like
    - 10.10.10.10 www.iiti.ac.in
  - Open a webpage and type www.iiti.ac.in it will go elsewhere
  - Alternatively create a .bat file with
    - @echo off
    - echo 10.10.10.10 www.iiti.ac.in >> C:\windows\system32\drivers\etc\host.txt
    - exist

DNS Spoofing Tools

- Dsniff
- dnsspoof
- Example
  - abc.com IP address is 10.0.0.1
  - Make it spoof to respond 100.0.1.1
  - In the text file dnssniff.txt write
  - 100.0.1.1 abc.com
  - [gateway]# dnsspoof -i eth0 -f /etc/dnssniff.txt
  - [bash]# host abc.com
  - abc.com has address of 100.0.1.1

DNS Spoofing in Reality

- DNS Replies are verified for
  - Coming from same IP address
  - Coming to the same port from which request was sent
  - Reply is for the same record as was asked in the previous question
  - Transaction ID match

How these Verifications are Overcome

- Coming from same IP address
  - Because authorative DNS server IP address can be discovered by offline queries
- Coming on the same port from which request was sent
  - Many DNS servers used static port numbers
- Answer is the same question that was asked
  - This is easy if attacker herself initiates a request
- Transaction ID match
  - Guess it

Dan Kamnisky Attack

- Kamnisky Attack
  - Flood the recursive name server with many answers
  - One of them have to be right and it works !
  - The identifier is not fully random so one can predict

Dan Kaminisky Attack

- Ask a recursive DNS server a question which is most likely not in its cache
  - Pick a non existing domain like rnd.india.microsoft.com
- With high probability name sever will contact the authorative name server of microsoft.com domain
- Attacker send a reply with canonical name

rnd.india.microsoft.com CNAME IN www.microsoft.com

www.microsoft.com A IN 68.177.102.22

Defending DNS Spoofing

- Many solutions focus on increasing the entropy of DNS query component
  - Transaction ID
  - Port number

DNSSEC

- Security extension to DNS protocol
- It uses public key infrastructure to give a guarantee on who is sending the reply
  - Use private key to digitally sign the message
  - Use public key to verify the message
  - Works fine as long as recipient believes in public-private key pair of sender
  - What stops from someone generating her own key pair and replying
  - Chain of trust relationship

How DNSSEC Works

- Each DNSSEC zone creates one or more pairs of public/private key(s)
  - Public portion put in DNSSEC record type DNSKEY

- Zones sign all RRsets with private key(s) and resolvers use DNSKEY(s) to verify RRsets
  - Each RRset has a signature attached to it: RRSIG

- So, if a resolver has a zone’s DNSKEY(s) it can verify that RRsets are intact by verifying their RRSIGs

Chain of Trust in DNSSEC

- Introduces 3 new resource records
  - RRSIG Signature over RR set using private key
  - DNSKEY Public key, needed for verifying a RRSIG
  - DS Delegation Signer; 'Pointer' for building chains of authentication
- Authoritative DNS server sends the following with reply
  - RR containing IP URL mapping
  - RRSIG
  - DNSKEY and
  - DS
- Verification can proceed one level higher the hierarchy
  - At no point a DNS server gives a DS which is bellow it
  - Problem is effectively addressed if Root Server becomes the highest signature verifier
  - As of July 2010 there is one signed root server up and running (http://www.root-dnssec.org/)

Key References for DNSSEC

- http://www.internetsociety.org/deploy360/dnssec/basics/
- http://www.root-dnssec.org/
- http://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions