

**Experiment No.:** 01

**Aim:** To perform a multidimensional data model using SQL queries. e.g., snowflake, star and fact constellation schema.

**Kiran K Patil 211070904**

**Mayuresh Murudkar 211070903**

**Batch D**

**Protocol : DNS**

**1. Theory:**

**DNS Protocol**

The Domain Network System (DNS) protocol helps Internet users and network devices discover websites using human-readable hostnames, instead of numeric IP addresses. Domain Name System (DNS) is a hostname for IP address translation service. DNS is a distributed database implemented in a hierarchy of name servers. It is an application layer protocol for message exchange between clients and servers. It is required for the functioning of the Internet.

**Need of DNS**

Every host is identified by the IP address but remembering numbers is very difficult for people also the IP addresses are not static therefore a mapping is required to change the domain name to the IP address. So DNS is used to convert the domain name of the websites to their numerical IP address.

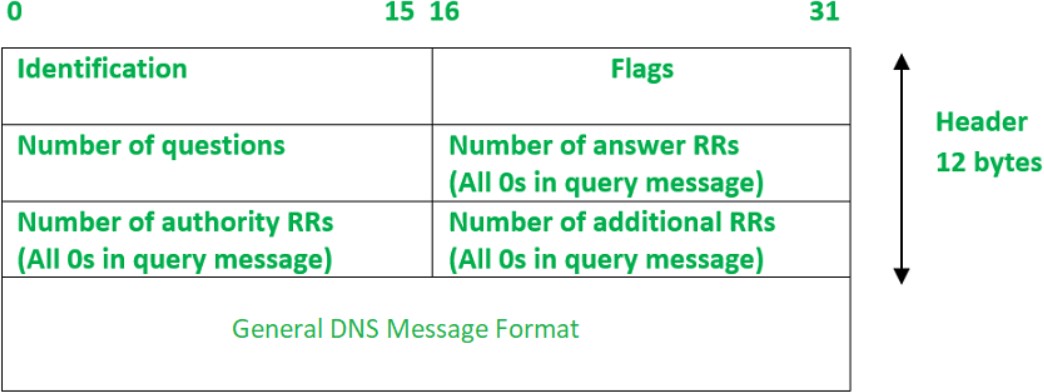
**The DNS process, simplified, works as follows:**

1. A browser, application or device called the DNS client, issues a DNS request or DNS address lookup, providing a hostname such as “example.com”.
2. The request is received by a DNS resolver, which is responsible for finding the correct IP address for that hostname. The DNS resolver looks for a DNS name server that holds the IP address for the hostname in the DNS request.
3. The resolver starts from the Internet’s root DNS server, moving down the hierarchy to Top Level Domain (TLD) DNS servers (“.com” in this case), down to the name server responsible for the specific domain “example.com”.
4. When the resolver reaches the authoritative DNS name server for “example.com”, it receives the IP address and other relevant details, and returns it to the DNS client. The DNS request is now resolved.
5. The DNS client device can connect to the server directly using the correct IP address.

**Types of Domain**

1. **Generic domains:** .com(commercial), .edu(educational), .mil(military), .org(nonprofit organization), .net(similar to commercial) all these are generic domains.
2. **Country domain:** .in (India) .us .uk
3. **Inverse domain:** if we want to know what is the domain name of the website. Ip to domain name mapping. So DNS can provide both the mapping for example to find the IP addresses of geeksforgeeks.org then we have to type

**DNS Packet format:**



* **Identification**: The identification field is made up of 16 bits which are used to match the response with the request sent from the client-side. The matching is carried out by this field as the server copies the 16-bit value of identification in the response message so the client device can match the queries with the corresponding response received from the server-side.
* **Flags**: It is 16 bits and is divided into the following Fields : -



Here is the description of each subfield of the Flags field:

* **QR (query/response):** It is a 1-bit subfield. If its value is 0, the message is of request type and if its value is 1, the message is of response type.
* **opcode**: It is a 4-bit subfield that defines the type of query carried by a message. This field value is repeated in the response. Following is the list of opcode values with a brief description:
* If the value of the opcode subfield is 0 then it is a standard query.
* The value 1 corresponds to an inverse of the query that implies finding the domain name from the IP Address.
* The value 2 refers to the server status request. The value 3 specifies the status reserved and therefore not used.
* **AA**: It is an Authoritative Answer. It is a 1-bit subfield that specifies the server is authoritative if the value is 1 otherwise it is non-authoritative for a 0 value.
* **TC**: It is Truncation. This is a 1-bit subfield that specifies if the length of the message exceeds the allowed length of 512 bytes, the message is truncated when using UDP services.
* **RD**: It is Recursion Desired. It is a 1-bit subfield that specifies if the value is set to 1 in the query message then the server needs to answer the query recursively. Its value is copied to the response message.
* **RA**: It is Recursion Available. It is a 1-bit subfield that specifies the availability of recursive response if the value is set to 1 in the response message.
* **Zero**: It is a 3-bit reserved subfield set to 0.
* **rCode**: It stands for Response Code. It is a 4-bit subfield used to denote whether the query was answered successfully or not. If not answered successfully then the status of error is provided in the response. Following is the list of values with their error status –
* The value 0 of rcode indicates no error.
* A value of 1 indicates that there is a problem with the format specification.
* Value 2 indicates server failure.
* Value 3 refers to the Name Error that implies the name given by the query does not exist in the domain.
* Value of 4 indicates that the request type is not supported by the server.
* The value 5 refers to the nonexecution of queries by the server due to policy reasons.
* **Number of Questions**- It is a 16-bit field to specify the count of questions in the Question Section of the message. It is present in both query and response messages.
* **A number of answer RRs**- It is a 16-bit field that specifies the count of answer records in the Answer section of the message. This section has a value of 0 in query messages. The server answers the query received from the client. It is available only in response messages.
* **A number of authority RRs**- It is a 16-bit field that gives the count of the resource records in the Authoritative section of the message. This section has a value of 0 in query messages. It is available only in response messages. It gives information that comprises domain names about one or more authoritative servers.

**1. The DNS protocol is as follows: -**

63f1 8180 0001 0001 0000 0000 0377 7777 0667 6f6f 676c 6503 636f 6d00 0001 0001 c00c

0001 0001 0000 0027 0004 8efa c024

**Fields in DNS Datagram: -**

1. Transaction ID: 63f1 (16)
2. Flags: 8180 (16)
3. Questions: 0001 (16)
4. Answer RRs: 0001 (16)
5. Authority RRs: 0000 (16)
6. Additional RRs: 0000 (16)
7. Queries: 0377 7777 0667 6f6f 676c 6503 636f 6d00 0001 0001 (160) [Q Name + 32 bits]
8. Answers: c00c 0001 0001 0000 0027 0004 8efa c024 (128) [This length can vary]

**2. Questionnaire: -**

**Answer the following questions regarding the given DNS packet header: -**

**63f1 8180 0001 0001 0000 0000 0377 7777 0667 6f6f 676c 6503 636f 6d00 0001 0001 c00c**

**0001 0001 0000 0027 0004 8efa c024**

**1) What is the length of the DNS header?**

**Ans**: 12 Bytes (96 Bits)

**2) What is the Transaction ID of this DNS datagram?**

**Ans**: 63f1 (1st and 2nd Byte)

**3) Identify the type of given data packet as Request or Response.**

**Ans**: The Flag field of given datagram is (8180)16. In binary = (1000 0001 1000 0000)2

The first bit represents type of message (0 -> Response; 1-> Request)

Hence, the given packet is Response packet.

**4) Is the server authoritative?**

**Ans**: No.

**Reason**: Flag = 1000 0001 1000 0000; 6th bit of the flag is 0 means server is not an authority for domain.

**5) Is the DNS Query corresponding to this response recursive?**

**Ans:** Yes.

**Reason**: The 8th bit of the Flag is 1. Hence, the Response is recursive in nature.

**6) Are there any errors?**

**Ans**: No. The last nibble (4 bits) of flags are 00002. Hence there is no error.

**7) What is the type of query and what it signifies?**

**Ans:** Last 4 bytes of Query field: (0001 0001)16. (0001 0001)16 => [Q Type(16 bit) + Q Class(16 bit)] 000116 = 110; Hence the query type is A. ‘A’ signifies Host address ipv4.

**8) What is TTL?**

**Ans:** 41st & 42nd by represents TTL. TTL = 002716 = 9910 (161 x 2 + 160 x 7)

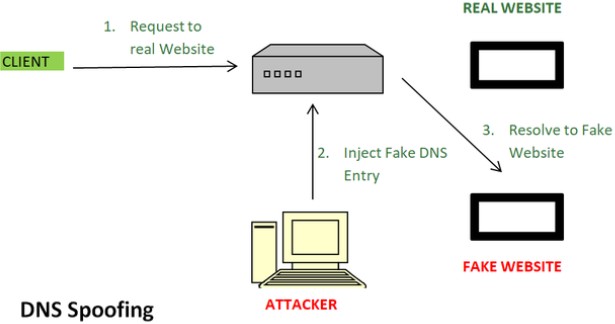
Hence, TTL is 39 seconds.

**2. Attack**

**1) DNS Spoofing**

**Attack Explanation with figure**

DNS spoofing or DNS cache poisoning is an attack in which altered DNS records are used to redirect users or data to a fraudulent website or link that is camouflaged as the actual destination.

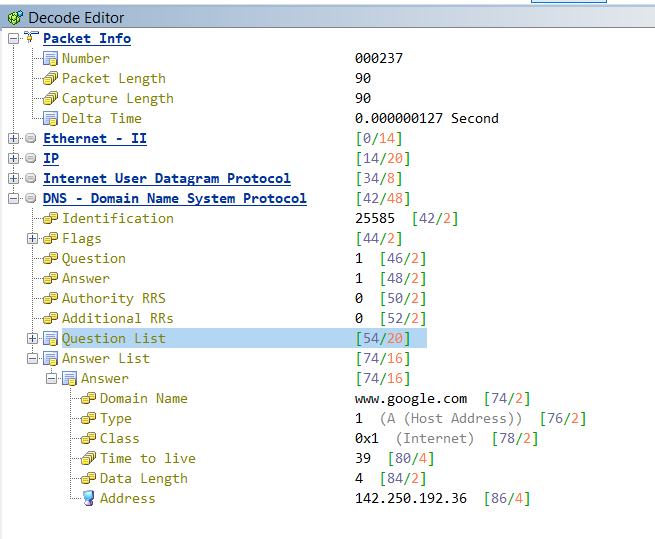


**Identify fields involved**

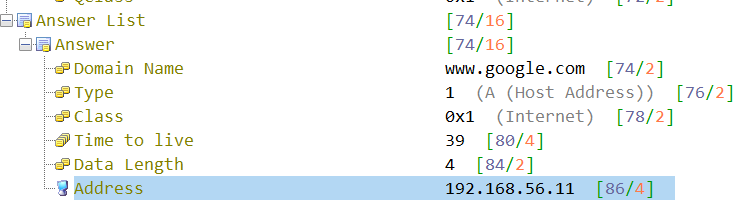
In DNS spoofing, the attacker typically sends a forged DNS response to a DNS query. The attacker may modify the IP address in the response to redirect the user to a malicious site. The modification usually occurs in the "Answer" section of the DNS response.

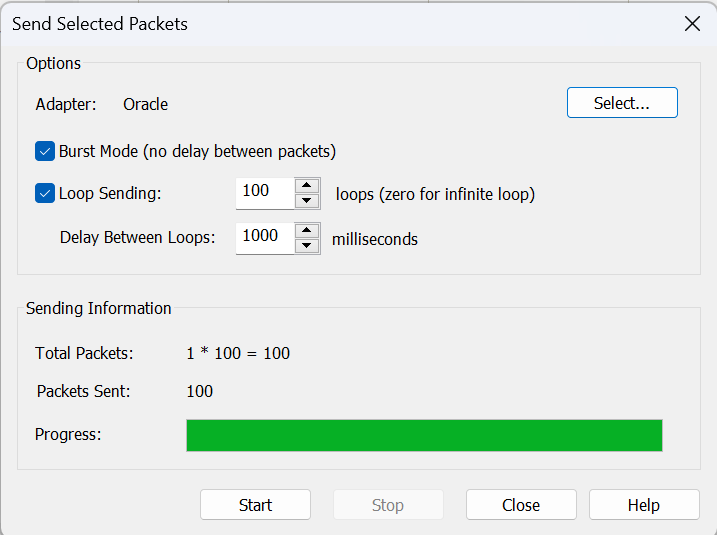
**Engage packet builder –**

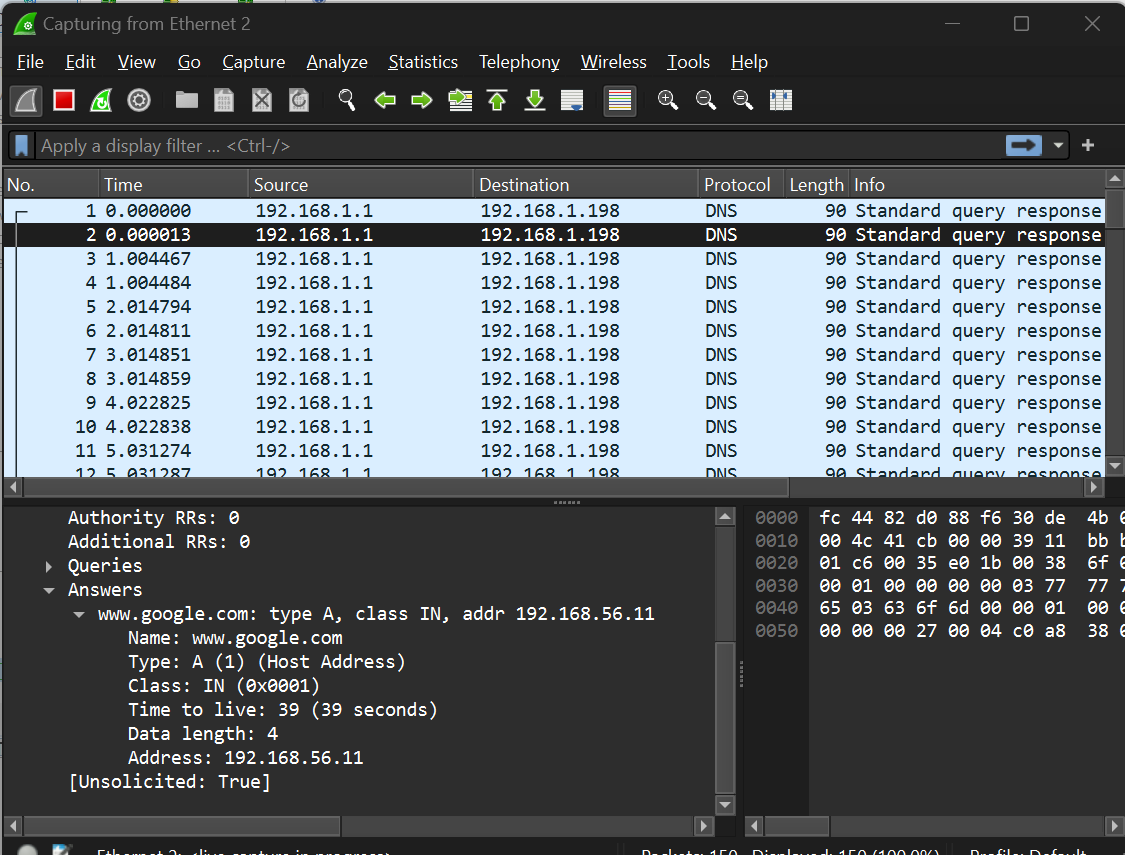
1) Original DNS Packet



2) Modified Address Field in DNS Packet:







**Algorithm**

# Initialize data structures

local\_dns\_cache = {}

threshold = 5

# Adjust as needed expected\_transaction\_ids = set() response\_anomalies = 0

# Main loop for continuous monitoring

while True:

# Capture DNS traffic

dns\_packet = capture\_dns\_packet()

# Extract relevant information from DNS response

transaction\_id, queried\_domain, response\_ip = extract\_dns\_info(dns\_packet)

# Check if the response is authoritative

is\_authoritative = is\_authoritative\_response(dns\_packet)

# Maintain local DNS cache if is\_authoritative:

local\_dns\_cache[queried\_domain] = response\_ip

# Monitor response anomalies

if transaction\_id in expected\_transaction\_ids:

if queried\_domain in local\_dns\_cache and response\_ip != local\_dns\_cache[queried\_domain]:

response\_anomalies += 1

# Threshold check

if response\_anomalies >= threshold:

trigger\_alert("Possible DNS spoofing attack detected")

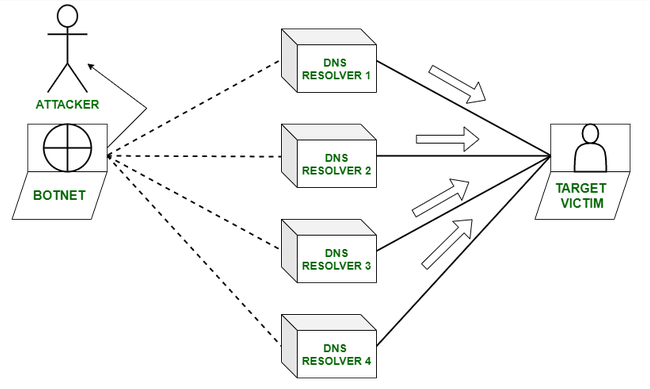
# Continue capturing and analyzing DNS traffic

**2) DNS Amplification**

**Attack Explanation with figure**

In this attack, the attacker replicates the domains and sends a large number of DNS queries to the server, this results in server sending all the records of the responses of the queries to the attacker which then gains the access over the network. For example, if the attacker generates 10 MB of DNS queries, then the server sends back about 1 TB of responses to those queries.

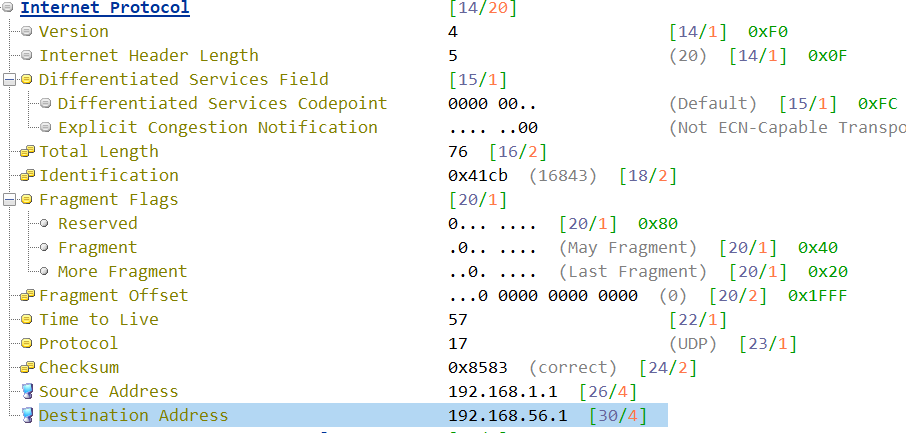
After that, the servers become so busy in handling the queries and traffic that they cannot request any other service from the legitimate users.

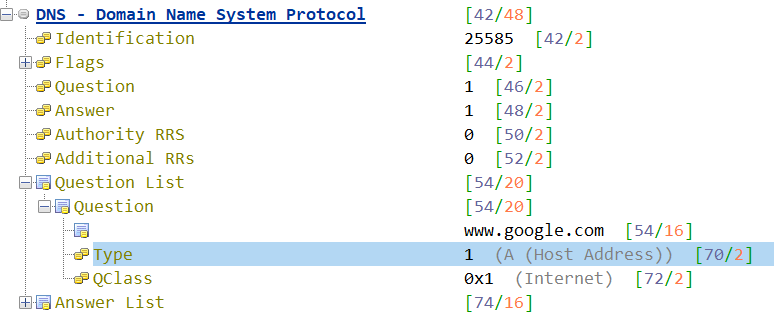


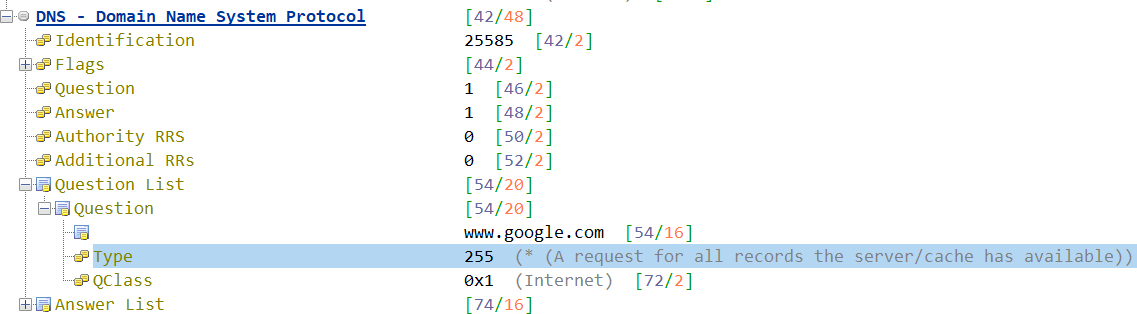
**Identify fields involved**

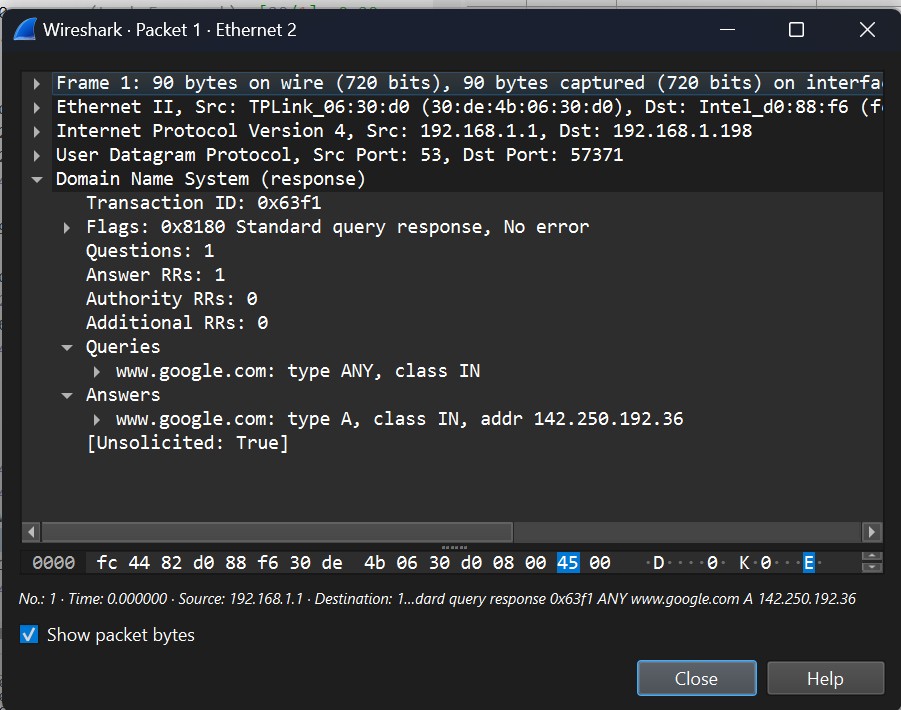
The modification involves forging the source IP address to make it appear as if the query is coming from the target (victim).

**Engage packet builder**









* **Algorithm**

# Initialize data structures dns\_amplification\_threshold = 10 # Adjust as needed dns\_amplification\_counter = 0

while True:

# Capture DNS traffic

dns\_packet = capture\_dns\_packet()

# Extract relevant information from DNS response

transaction\_id, queried\_domain, response\_ip = extract\_dns\_info(dns\_packet)

# Check for DNS amplification if len(queried\_domain) > 1:

dns\_amplification\_counter += 1

# DNS amplification threshold check

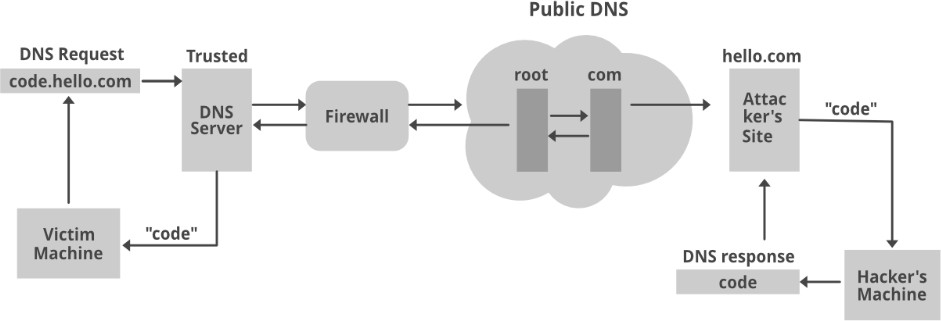
if dns\_amplification\_counter >= dns\_amplification\_threshold: trigger\_alert("Possible DNS amplification attack detected")

# Continue capturing and analyzing DNS traffic

**3) DNS Tunneling**

**Attack Explanation with figure**

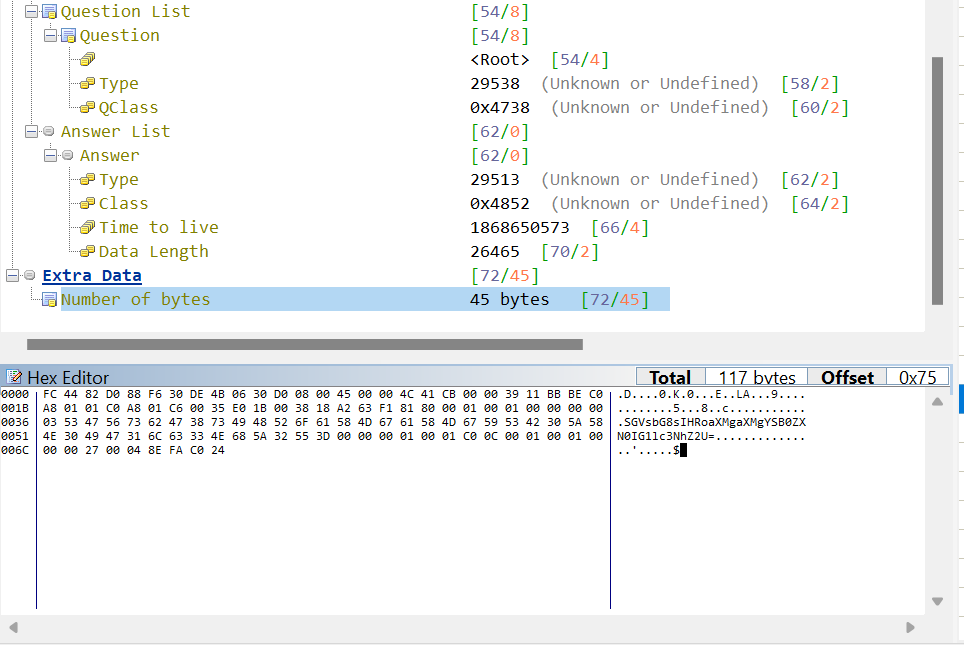
DNS Tunneling is a strategy for a digital exploit that encodes the information of different programs or protocols in DNS inquiries and responses. DNS tunnelling frequently incorporates information payloads that can be added to an exploited domain name server and used to control a distant system and applications.

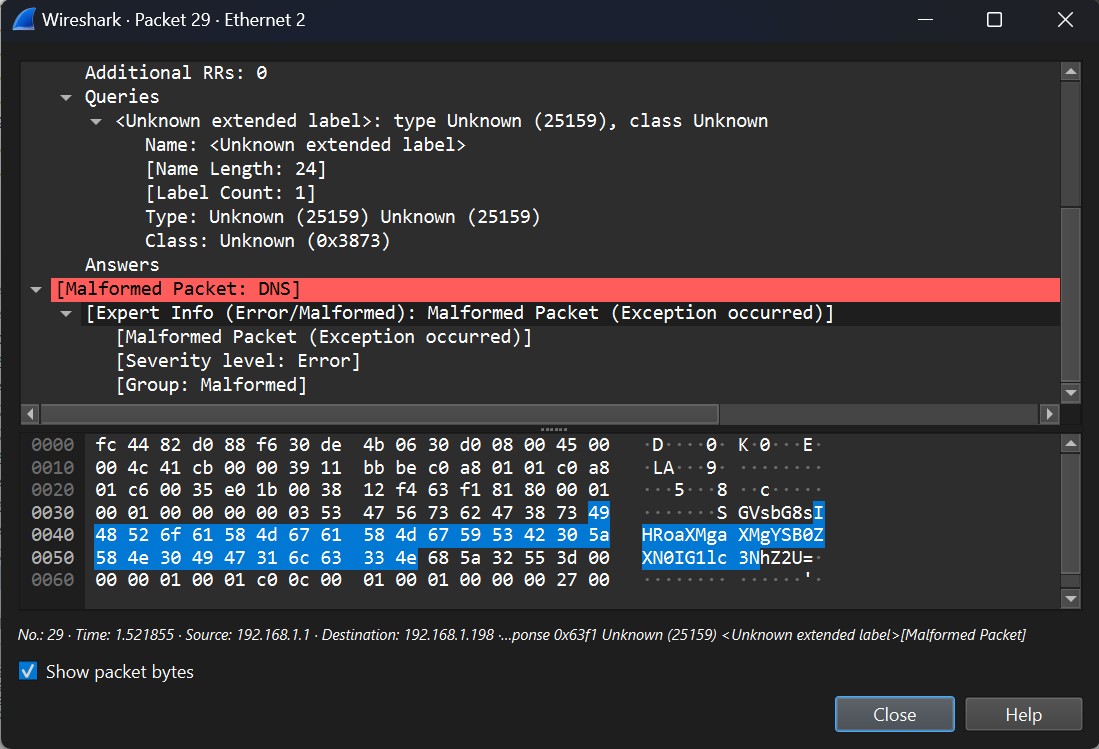


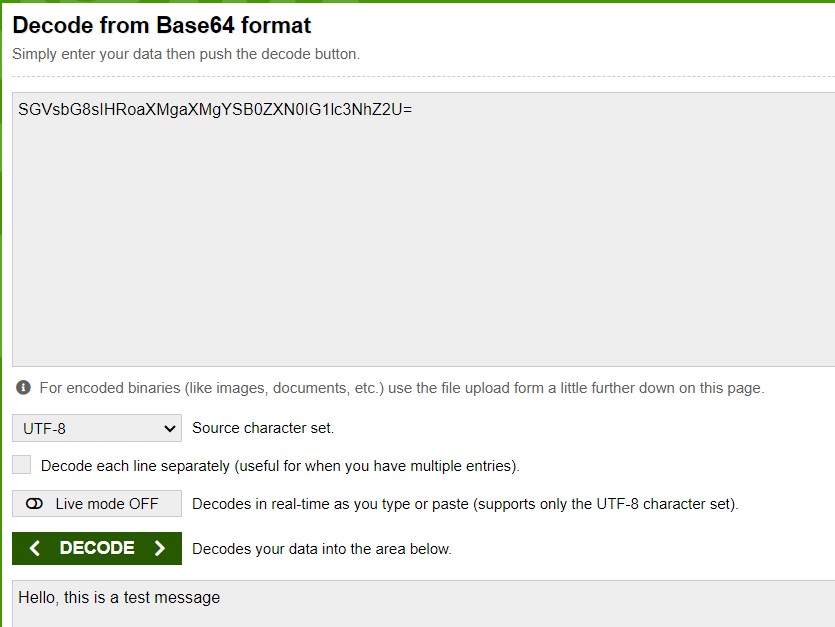
# Identify fields involved

DNS tunneling involves encoding non-DNS data within DNS queries and responses. Attackers manipulate various fields, including the "Query" and "Answer" sections, to embed data covertly. This can include payload data and control information, allowing unauthorized communication through DNS.

# Engage packet builder







* **Algorithm**

# Initialize data structures dns\_tunneling\_threshold = 5 # Adjust as needed dns\_tunneling\_counter = 0

while True:

# Capture DNS traffic

dns\_packet = capture\_dns\_packet()

# Extract relevant information from DNS response

transaction\_id, queried\_domain, response\_ip = extract\_dns\_info(dns\_packet)

# Check for DNS tunneling

if is\_unusual\_query\_pattern(queried\_domain): dns\_tunneling\_counter += 1

# DNS tunneling threshold check

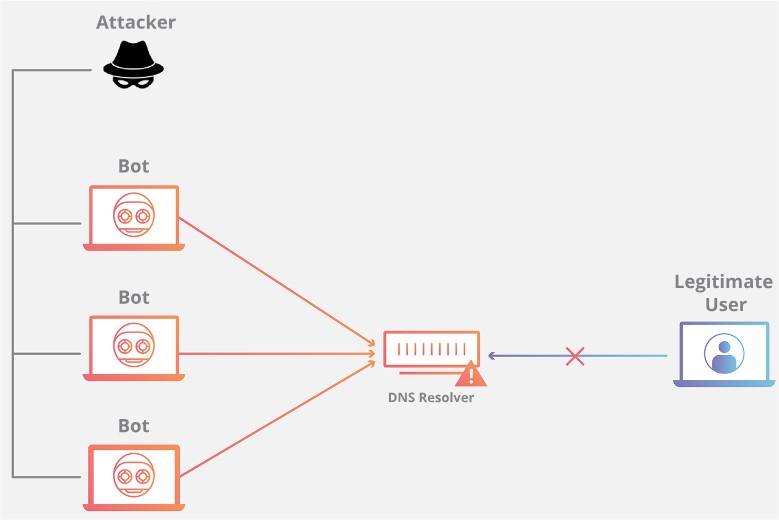
if dns\_tunneling\_counter >= dns\_tunneling\_threshold: trigger\_alert("Possible DNS tunneling detected")

# Continue capturing and analyzing DNS traffic

# 4) DDoS Attack

**Attack Explanation with figure**

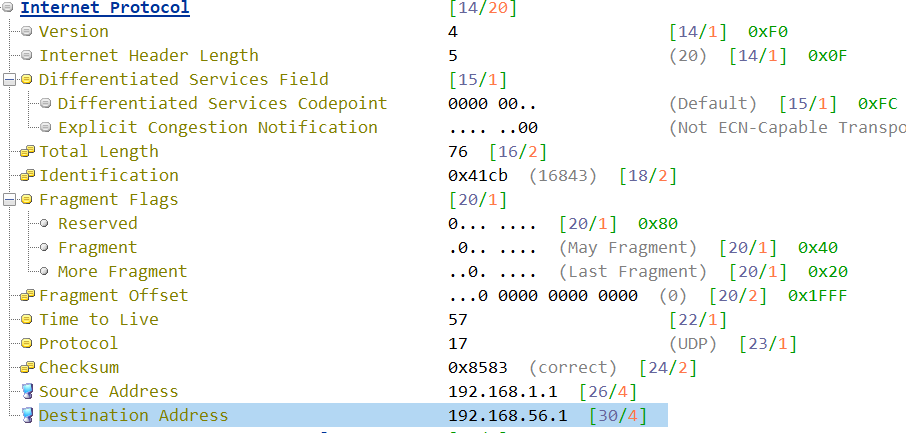
A DNS ﬂood is a type of [distributed denial-of-service attack (DDoS)](https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/) where an attacker ﬂoods a particular domain’s DNS servers in an attempt to disrupt DNS resolution for that [domain](https://www.cloudflare.com/learning/dns/glossary/what-is-a-domain-name/). If a user is unable to ﬁnd the phonebook, it cannot lookup the address in order to make the call for a particular resource. By disrupting DNS resolution, a DNS ﬂood attack will compromise a website, API, or web application's ability respond to legitimate traﬃc.

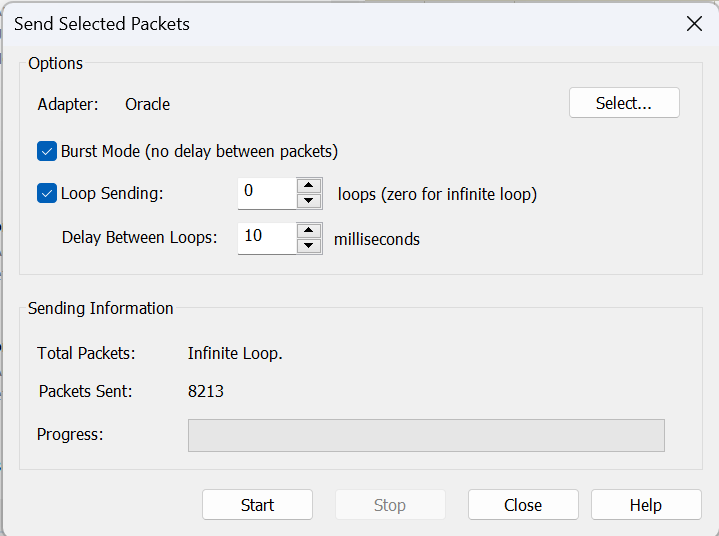


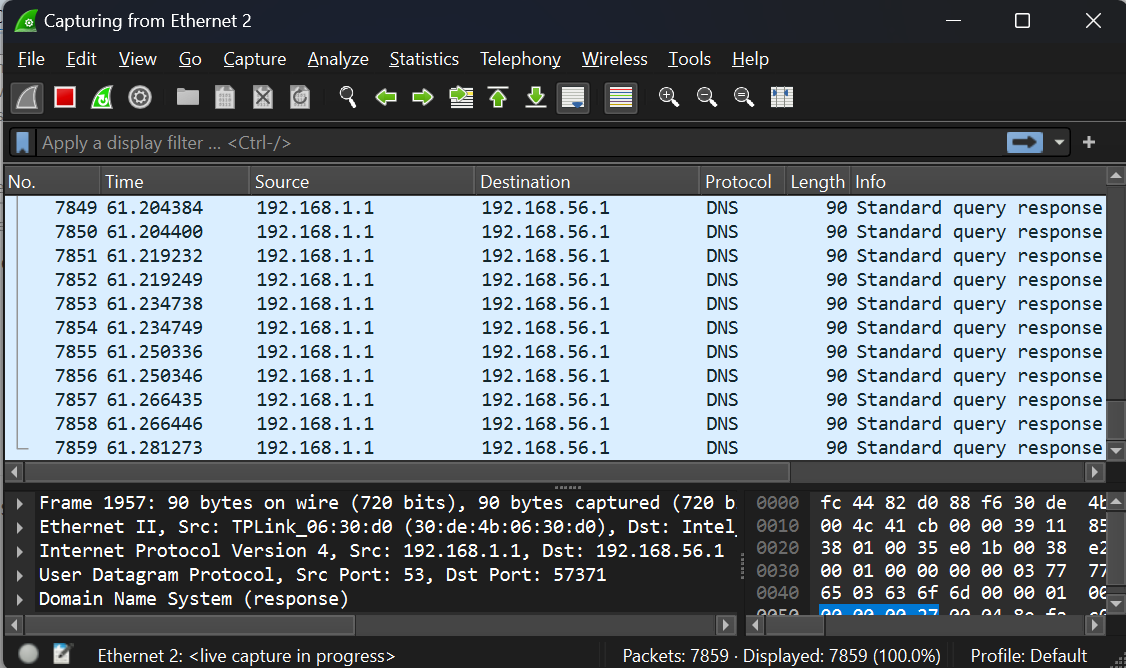
# Identify fields involved

The modification may involve forging the source IP address in these requests to make it difficult to trace the origin of the attack. Additionally, attackers may manipulate various fields in DNS packets to increase the load on DNS servers and exhaust their resources.

# Engage packet builder







* **Algorithm**

# Initialize data structures

ddos\_threshold = 100 # Adjust as needed ddos\_counter = 0

while True:

# Capture DNS traffic

dns\_packet = capture\_dns\_packet()

# Extract relevant information from DNS response

transaction\_id, queried\_domain, response\_ip = extract\_dns\_info(dns\_packet)

# Check for DDoS

if is\_multiple\_queries\_same\_domain(transaction\_id): ddos\_counter += 1

# DDoS threshold check

if ddos\_counter >= ddos\_threshold: trigger\_alert("Possible DDoS attack detected")

# Continue capturing and analyzing DNS traffic

**Conclusion**:

|  |  |  |
| --- | --- | --- |
| **Attack** | **Fields Used in Attack** | **Comments** |
| **DNS Spoofing** | DNS IP Address | Attacker modifies the DNS response to a DNS Query to redirect the target to a malicious URL. |
| **DNS Amplification** | Source IP Address | Attackers send small queries with spoofed source IP addresses to open DNS resolvers, triggering responses that are disproportionately larger, causing amplification. |
| **DNS Tunneling** | Query and Answer Sections | Attackers manipulate various fields, including the "Query" and "Answer" sections, to embed non-DNS data covertly. |
| **DDoS Attack** | Source IP Address | The attacker floods the DNS server with a massive volume of requests. |