VUT
Faculty of Information Technology

DNS Monitor Manual

Author:
Denys Chernenko
xchern08

October 20, 2024

Contents

1 Theory

1.1 DNS Introduction

The Domain Name System (DNS) is often referred to as the "phonebook of the Internet." Humans access information online through domain names, such as nytimes.com or espn.com. Web browsers interact with Internet Protocol (IP) addresses, which are numerical labels assigned to devices connected to a computer network. DNS translates these human-friendly domain names into IP addresses, allowing browsers to load Internet resources effectively.

1.2 Packet Sniffing

Packet sniffing is the process of capturing network packets in real time using libraries such as `libpcap`.

- `libpcap` is a widely used C/C++ library for network packet capture. It allows for low-level access to network traffic.
- This library enables programs to monitor network interfaces and capture raw packets as they are transmitted or received.

1.3 DNS Record Types

In this section, we explore various DNS record types that are fundamental to the DNS operation.

1.3.1 A (Address Record)

Maps a domain name to an IPv4 address, allowing browsers to connect to the corresponding server. It is crucial for translating human-readable domain names into machine-friendly IP addresses.

1.3.2 AAAA (IPv6 Address Record)

Similar to the A record, but maps a domain name to an IPv6 address. It supports the newer internet protocol, ensuring compatibility with modern networks.

1.3.3 NS (Name Server Record)

Specifies the authoritative DNS servers for a domain. It delegates the responsibility for handling DNS queries to the specified name servers.

1.3.4 MX (Mail Exchange Record)

Directs email to the mail server responsible for a domain. It uses priority values to determine the order in which mail servers are contacted.

1.3.5 SOA (Start of Authority Record)

Provides administrative information about a domain, including the primary name server and contact email for the domain administrator. It defines the authority over the domain and manages DNS zone settings.

1.3.6 CNAME (Canonical Name Record)

Maps an alias domain name to the canonical (true) domain name, allowing multiple domain names to refer to the same server. This simplifies domain management by pointing several domain names to one IP address.

1.3.7 SRV (Service Locator Record)

Specifies the location of services, including the hostname and port number for a particular service. It allows for the automatic discovery of services within a network, facilitating efficient resource management.

2 Implementation

2.1 Introduction

The ‘dns-monitor‘ program is implemented in the C programming language, utilizing various libraries for functionality, including the ‘libpcap‘ library for packet capture.

2.1.1 Start of Execution

The program begins execution through the following main files:

- `dns-monitor.c`: The main program file that initiates execution.
- `arguments-parse.c`: Contains functions to parse command-line arguments.

2.1.2 Arguments Structure

The structure for command-line arguments is defined as follows:

```
typedef struct ARGUMENTS {
    char interface[MAX_LEN_IN];        // Network interface
    int verbose;                       // Verbosity flag
    char *pcap_file;                   // PCAP file for processing
    char *domain_file;                 // Domain names output file
    char *translation_file;            // Translations output file
} ARGUMENTS;
```

2.1.3 Packet Capturing

After parsing the arguments and storing them, the program starts packet capturing with the following function:

```
start_packet_capture(arguments);
```

2.2 Packet Sniffing Logic

Packet sniffing is implemented using the ‘libpcap‘ library. Based on the provided arguments, the program starts capturing packets either from a PCAP file or a specified network interface.

- **Captured Packets:**
  - The program is designed to capture only DNS packets, specifically those with a UDP header.
- **Packet Filtering:**
  - If a captured packet does not contain a UDP header, it is simply skipped.
- **Unhandled DNS Record Types:**
  - If the DNS record type is unhandled in the program (as described in the theory section), the program will write "UNKNOWN TYPE" to the output.
- **PCAP File Handling:**
  - If a PCAP file is provided, the program captures packets until it reaches the end of the file.
- **Interface Handling:**
  - If a network interface is specified, the program captures packets continuously until the user terminates the program using a signal like `CTRL+C`.

2.3 DNS Parsing Logic

After capturing DNS packets with a UDP header, the program starts parsing the information using the function `void support_dns_packet_parser`.

- **Data Structures:** The following structures are used to represent the DNS packet and its components:
  - `dns_header`: Contains the header information of the DNS packet, including the unique identifier, flags, and counts of various sections (questions, answers, authority, and additional records).

    ```
    typedef struct DNS_HEADER  {
        uint16_t id;
        uint16_t flags;
        uint16_t q_count;
        uint16_t an_count;
        uint16_t ns_count;
        uint16_t ar_count;
    } dns_header;
    ```

  - `question`: Represents a DNS query, including the domain name, query type, and query class.

    ```
    typedef struct QUESTION {
        char *qname;
        uint16_t qtype;
        uint16_t qclass;
    } question;
    ```

  - `resource_record`: Represents a resource record, including the domain name, type, class, TTL, and RDATA. This structure is extended to accommodate attributes for various DNS record types such as SOA, MX, and SRV.

    ```
    typedef struct RESOURCE_RECORD {
        char *name;
        uint16_t type;
        uint16_t a_class;
        uint32_t ttl;
        uint16_t rdlength;
        char *rdata;
    ```

```
        char *mname;
        char *rname;
        uint32_t serial_number;
        uint32_t refresh_interval;
        uint32_t retry_interval;
        uint32_t expire_limit;
        uint32_t minimum_ttl;

        uint16_t preference;
        char *mail_exchange;

        uint16_t priority;
        uint16_t weight;
        uint16_t port;
        char *target;
    } resource_record;
```

  - `dns_packet`: Represents the entire DNS packet, consisting of the DNS header, an array of questions, and arrays for answers, authority records, and additional records.

```
typedef struct DNS_PACKET {
    dns_header header;
    question *questions;
    resource_record *answers;
    resource_record *authorities;
    resource_record *additionals;
} dns_packet;
```

- **Parsing DNS Information:**

  - Inside the `support_dns_packet_parser` function, the program parses various components of the DNS packet, including:

    DNS Header, **DNS Question Section**, **DNS Answers Section**, **Authorities Section**, **Additional Records Section**

- **Name Parsing:** Within this function, `void parse_dns_name` helps handle:
  - Comprehension labels: Representing parts of a DNS domain name, structured hierarchically.
  - Simple labels: Basic domain name components that do not have any hierarchical structure.

2.4 Additional Info

- **Hash Table Implementation:** The program uses a hash table to efficiently store and manage unique domain names. After executing the program with parameters `-d` or `-t`, the output files will contain only unique domain names.
- **Hash Table Structure:**
  - `typedef struct Domain_Item` {
    - `char *domain_name;`
    - `struct Domain_Item *next;`

    } `Domain_Item;`
  - `typedef struct Hash_Domain_Table` {
    - `Domain_Item **table;`
    - `size_t size;`

    } `Hash_Domain_Table;`
- **Purpose of the Hash Table:**
  - The hash table is implemented to quickly add domain names or domain names with their IPs (based on the arguments provided).
  - It ensures the uniqueness of domain names stored, efficiently handling duplicates.
- **Storing Domain Names into File:**
  - Storing domain names or domain names with their IPs is performed after all packets have been processed and stored in the hash table.

  - The data is then written from the hash table into the specified file. If the file already exists, its contents are overwritten with the new domain names. If the file does not exist, a new file is created with the provided name, and the domain names are stored in it.

- **-h Parameter for Help:**
  - The program supports a `-h` parameter to display a help message, which includes:
  - `./dns-monitor -h`
    * Description of each command-line parameter.
    * Examples of usage.
  - After displaying the help message, the program terminates execution.

3 Usage

3.1 Parameters Description

- **Command-Line Parameters:** The program supports the following command-line parameters for configuration:
  - `-i <interface>`: Specifies the name of the network interface to capture packets from.
  - `-p <pcapfile>`: Specifies the name of the PCAP file to process.
    * **Note:** Either `-i` or `-p` must be provided, but not both or neither.
  - `-v`: Enables verbose mode for more detailed output of DNS messages.
  - `-d <domainsfile>`: Specifies the name of the file where the captured domain names will be saved.
  - `-t <translationsfile>`: Specifies the name of the file where the domain names along with their IP addresses will be saved.
  - `-h`: Displays the help message with usage information and exits the program.

3.2 Examples of Usage

- **Display Help Message:**
  - `./dns-monitor -h`
- **Capture from Network Interface with Verbose Mode:**
  - `./dns-monitor -i eth0 -v`
  - This example captures DNS packets from the `eth0` interface with detailed output.
- **Capture from a PCAP File and Save Domain Names:**
  - `./dns-monitor -p example.pcap -d domains.txt`
  - In this case, the program processes the `example.pcap` file and saves the captured domain names to `domains.txt`.
- **Capture from a PCAP File with Translation File:**
  - `./dns-monitor -p example.pcap -t translations.txt`
  - The program processes the `example.pcap` file and saves the domain names along with their IP addresses to `translations.txt`.
- **Capture from Network Interface with Domain and Translation Files:**
  - `./dns-monitor -i eth0 -d domains.txt -t translations.txt`
  - Captures DNS packets from the `eth0` interface and saves domain names to `domains.txt` and their corresponding IPs.

4 Testing

4.1 DNS A Record Resolution (non verbose)

- **Command:**

```
Resolve-DnsName -Name wikipedia.org -Type A
```

After executing the above command, I captured the network packet using Wireshark and saved it to a file named `A_test.pcap`.

- **Program Execution:**

```
sudo ./dns-monitor -p A_test.pcap
```

- **Output:**

```
2024-10-20 00:18:45 2001:1aeb:1000:1900:89b8:b31:af08:8033
-> 2001:1aeb:1000:1900:10:18ff:fe23:4744 (Q 1/0/0/0)
2024-10-20 00:18:45 2001:1aeb:1000:1900:10:18ff:fe23:4744
-> 2001:1aeb:1000:1900:89b8:b31:af08:8033
(R 1/1/0/0)
```

4.2 DNS A Record Resolution (verbose)

- **Command:**

```
Resolve-DnsName -Name wikipedia.org -Type A
```

  After executing the above command, I captured the network packet using Wireshark and saved it to a file named `A_test.pcap`.

- **Program Execution:**

```
sudo ./dns-monitor -p A_test.pcap -v
```

- **Output:**

```
Timestamp: 2024-10-20 00:18:45
SrcIP: 2001:1aeb:1000:1900:89b8:b31:af08:8033
DstIP: 2001:1aeb:1000:1900:10:18ff:fe23:4744
SrcPort: UDP/59154
DstPort: UDP/53
Identifier: 0xEFD1
Flags: QR=0, OPCODE=0, AA=0, TC=0, RD=1, RA=0, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN A
```

```
====================

Timestamp: 2024-10-20 00:18:45
SrcIP: 2001:1aeb:1000:1900:10:18ff:fe23:4744
DstIP: 2001:1aeb:1000:1900:89b8:b31:af08:8033
SrcPort: UDP/53
DstPort: UDP/59154
Identifier: 0xEFD1
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN A

[Answer Section]
wikipedia.org. 185 IN A 185.15.58.224
====================
```

4.3 DNS AAAA Record Resolution (verbose)

- **Command:**

```
Resolve-DnsName -Name wikipedia.org -Type AAAA
```

After executing the above command, I captured the network packet using Wireshark and saved it to a file named `AAAA_test.pcap`.

- **Program Execution:**

```
sudo ./dns-monitor -p AAAA_test.pcap -v
```

- **Output:**

```
Timestamp: 2024-10-13 18:36:05
SrcIP: 100.69.162.61
DstIP: 147.229.3.200
SrcPort: UDP/64382
DstPort: UDP/53
Identifier: 0x132F
```

```
Flags: QR=0, OPCODE=0, AA=0, TC=0, RD=1, RA=0, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN AAAA
====================

Timestamp: 2024-10-13 18:36:05
SrcIP: 147.229.3.200
DstIP: 100.69.162.61
SrcPort: UDP/53
DstPort: UDP/64382
Identifier: 0x132F
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN AAAA

[Answer Section]
wikipedia.org. 300 IN AAAA 2a02:ec80:600:ed1a::1
====================
```

4.4 DNS NS Record Resolution (verbose)

- **Command:**

```
Resolve-DnsName -Name wikipedia.org -Type NS
```

After executing the above command, I captured the network packet using Wireshark and saved it to a file named `NS_test.pcap`.

- **Program Execution:**

```
sudo ./dns-monitor -p NS_test.pcap -v
```

- **Output:**

```
Timestamp: 2024-10-13 18:20:42
SrcIP: 100.69.162.61
```

```
DstIP: 147.229.3.200
SrcPort: UDP/54782
DstPort: UDP/53
Identifier: 0xC87B
Flags: QR=0, OPCODE=0, AA=0, TC=0, RD=1, RA=0, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN NS
====================

Timestamp: 2024-10-13 18:20:42
SrcIP: 147.229.3.200
DstIP: 100.69.162.61
SrcPort: UDP/53
DstPort: UDP/54782
Identifier: 0xC87B
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN NS

[Answer Section]
wikipedia.org. 2578 IN NS ns1.wikimedia.org.
wikipedia.org. 2578 IN NS ns2.wikimedia.org.
wikipedia.org. 2578 IN NS ns0.wikimedia.org.

[Additional Section]
ns0.wikimedia.org. 80719 IN A 208.80.154.238
ns1.wikimedia.org. 80719 IN A 208.80.153.231
ns2.wikimedia.org. 80719 IN A 198.35.27.27
====================
```

4.5 DNS MX Record Resolution (verbose)

- **Command:**

```
Resolve-DnsName -Name wikipedia.org -Type MX
```

After executing the above command, I captured the network packet using Wireshark and saved it to a file named `MX_test.pcap`.

- **Program Execution:**

```
sudo ./dns-monitor -p MX_test.pcap -v
```

- **Output:**

```
Timestamp: 2024-10-15 12:37:37
SrcIP: 100.69.161.101
DstIP: 147.229.3.200
SrcPort: UDP/56522
DstPort: UDP/53
Identifier: 0xF008
Flags: QR=0, OPCODE=0, AA=0, TC=0, RD=1, RA=0, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN MX
====================

Timestamp: 2024-10-15 12:37:37
SrcIP: 147.229.3.200
DstIP: 100.69.161.101
SrcPort: UDP/53
DstPort: UDP/56522
Identifier: 0xF008
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN MX

[Answer Section]
wikipedia.org. 300 IN MX 10 mx-in2001.wikimedia.org.
wikipedia.org. 300 IN MX 10 mx-in1001.wiki
====================
```

4.6 DNS SOA Record Resolution (verbose)

- **Command:**

```
Resolve-DnsName -Name wikipedia.org -Type SOA
```

After executing the above command, I captured the network packet using Wireshark and saved it to a file named `SOA_test.pcap`.

- **Program Execution:**

```
sudo ./dns-monitor -p SOA_test.pcap -v
```

- **Output:**

```
Timestamp: 2024-10-13 18:42:34
SrcIP: 100.69.162.61
DstIP: 147.229.3.200
SrcPort: UDP/56452
DstPort: UDP/53
Identifier: 0x5EFE
Flags: QR=0, OPCODE=0, AA=0, TC=0, RD=1, RA=0, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN SOA
====================

Timestamp: 2024-10-13 18:42:34
SrcIP: 147.229.3.200
DstIP: 100.69.162.61
SrcPort: UDP/53
DstPort: UDP/56452
Identifier: 0x5EFE
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
wikipedia.org. IN SOA

[Answer Section]
wikipedia.org. 3571 IN SOA ns0.wikimedia.org. hostmaster.wikimedia.org.
2024072620 43200 7200 1209600 3600
====================
```

4.7 DNS CNAME Record Resolution (verbose)

- **Command:**

```
Resolve-DnsName -Name wikipedia.org -Type CNAME
```

After executing the above command, I captured the network packet using Wireshark and saved it to a file named `CNAME_test.pcap`.

- **Program Execution:**

```
sudo ./dns-monitor -p CNAME_test.pcap -v
```

- **Output:**

```
Timestamp: 2016-01-08 21:59:15
SrcIP: 4.2.2.1
DstIP: 172.16.16.154
SrcPort: UDP/53
DstPort: UDP/57434
Identifier: 0xB10B
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
www.espn.com. IN A

[Answer Section]
www.espn.com. 225 IN CNAME redir.espn.gns.go.com.
redir.espn.gns.go.com. 223 IN A 68.71.212.158
====================

Timestamp: 2016-01-08 21:59:15
SrcIP: 4.2.2.1
DstIP: 172.16.16.154
SrcPort: UDP/53
DstPort: UDP/22689
Identifier: 0xF454
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
espn.go.com. IN A

[Answer Section]
```

```
espn.go.com. 143 IN CNAME espn.gns.go.com.
espn.gns.go.com. 4 IN A 199.181.133.61
====================

Timestamp: 2016-01-08 21:59:15
SrcIP: 4.2.2.1
DstIP: 172.16.16.154
SrcPort: UDP/53
DstPort: UDP/52723
Identifier: 0x5D96
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
cdn.optimizely.com. IN A

[Answer Section]
cdn.optimizely.com. 11 IN CNAME wac.946A.edgecastcdn.net.
wac.946A.edgecastcdn.net. 1357 IN CNAME gp1.wac.v2cdn.net.
gp1.wac.v2cdn.net. 2664 IN A 72.21.91.8
====================

Timestamp: 2016-01-08 21:59:15
SrcIP: 4.2.2.1
DstIP: 172.16.16.154
SrcPort: UDP/53
DstPort: UDP/23201
Identifier: 0xCABF
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
a1.espncdn.com. IN A

[Answer Section]
a1.espncdn.com. 134 IN CNAME a.espncdn.com.edgesuite.net.
a.espncdn.com.edgesuite.net. 10943 IN CNAME a1589.g1.akamai.net.
a1589.g1.akamai.net. 13 IN A 72.246.56.35
a1589.g1.akamai.net. 13 IN A 72.246.56.83
====================

Timestamp: 2016-01-08 21:59:15
SrcIP: 4.2.2.1
```

```
DstIP: 172.16.16.154
SrcPort: UDP/53
DstPort: UDP/57920
Identifier: 0xE7B6
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
assets.espn.go.com. IN A

[Answer Section]
assets.espn.go.com. 234 IN CNAME assets.espn.go.com.edgesuite.net.
assets.espn.go.com.edgesuite.net. 18738 IN CNAME a1589.g.akamai.net.
a1589.g.akamai.net. 8 IN A 69.31.75.194
a1589.g.akamai.net. 8 IN A 69.31.75.203
====================

Timestamp: 2016-01-08 21:59:15
SrcIP: 4.2.2.1
DstIP: 172.16.16.154
SrcPort: UDP/53
DstPort: UDP/49283
Identifier: 0xBFA4
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
a2.espncdn.com. IN A

[Answer Section]
a2.espncdn.com. 284 IN CNAME a.espncdn.com.edgesuite.net.
a.espncdn.com.edgesuite.net. 11448 IN CNAME a1589.g1.akamai.net.
a1589.g1.akamai.net. 13 IN A 72.246.56.83
a1589.g1.akamai.net. 13 IN A 72.246.56.35
====================

Timestamp: 2016-01-08 21:59:15
SrcIP: 4.2.2.1
DstIP: 172.16.16.154
SrcPort: UDP/53
DstPort: UDP/21916
Identifier: 0x26E0
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0
```

```
[Question Section]
a4.espncdn.com. IN A

[Answer Section]
a4.espncdn.com. 185 IN CNAME a.espncdn.com.edgesuite.net.
a.espncdn.com.edgesuite.net. 11448 IN CNAME a1589.g1.akamai.net.
a1589.g1.akamai.net. 13 IN A 72.246.56.35
a1589.g1.akamai.net. 13 IN A 72.246.56.83
====================
```

4.8 DNS SRV Record Resolution (verbose)

- SRV was hard to create/find, so this is a random packet that appeared in Wireshark, which I then downloaded into a file `SRV_test.pcap`. Also UNKNOWN TYPE is record type 41, which is unhanddled in my program.
- **Program Execution:**

  ```
  sudo ./dns-monitor -p SRV_test.pcap -v
  ```

- **Output:**

  ```
  Timestamp: 2024-10-16 17:43:07
  SrcIP: 100.69.167.117
  DstIP: 147.229.3.200
  SrcPort: UDP/50259
  DstPort: UDP/53
  Identifier: 0x209F
  Flags: QR=0, OPCODE=0, AA=0, TC=0, RD=1, RA=0, AD=1, CD=0, RCODE=0

  [Question Section]
  _xmpp-client._tcp.jabber.org. IN SRV

  [Additional Section]
  UNKNOWN TYPE
  ====================
  ```

```
Timestamp: 2024-10-16 17:43:07
SrcIP: 100.69.167.117
DstIP: 147.229.3.100
SrcPort: UDP/50259
DstPort: UDP/53
Identifier: 0x209F
Flags: QR=0, OPCODE=0, AA=0, TC=0, RD=1, RA=0, AD=1, CD=0, RCODE=0

[Question Section]
_xmpp-client._tcp.jabber.org. IN SRV

[Additional Section]
UNKNOWN TYPE
====================

Timestamp: 2024-10-16 17:43:07
SrcIP: 147.229.3.200
DstIP: 100.69.167.117
SrcPort: UDP/53
DstPort: UDP/50259
Identifier: 0x209F
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
_xmpp-client._tcp.jabber.org. IN SRV

[Answer Section]
_xmpp-client._tcp.jabber.org. 60 IN SRV 30 30 5222 zeus.jabber.org.
_xmpp-client._tcp.jabber.org. 60 IN SRV 60 30 5222 zeus-v6.jabber.org.

[Additional Section]
zeus.jabber.org. 900 IN A 208.68.163.216
zeus-v6.jabber.org. 900 IN AAAA 2605:da00:5222:5269::2:3
UNKNOWN TYPE
====================

Timestamp: 2024-10-16 17:43:07
SrcIP: 147.229.3.100
DstIP: 100.69.167.117
SrcPort: UDP/53
DstPort: UDP/50259
```

```
Identifier: 0x209F
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=0, CD=0, RCODE=0

[Question Section]
_xmpp-client._tcp.jabber.org. IN SRV

[Answer Section]
_xmpp-client._tcp.jabber.org. 60 IN SRV 60 30 5222 zeus-v6.jabber.org.
_xmpp-client._tcp.jabber.org. 60 IN SRV 30 30 5222 zeus.jabber.org.

[Additional Section]
zeus.jabber.org. 830 IN A 208.68.163.216
zeus-v6.jabber.org. 830 IN AAAA 2605:da00:5222:5269::2:3
UNKNOWN TYPE
====================
```

4.9 SRV Resolution and Logging (non verbose)

- This section uses the same packet as shown in the previous section, so there is no need to display the verbose output again.
- **Program Execution:**

```
sudo ./dns-monitor -p SRV_test.pcap -d output.txt
```

- **Output:**

```
2024-10-16 17:43:07 100.69.167.117 -> 147.229.3.200 (Q 1/0/0/1)
2024-10-16 17:43:07 100.69.167.117 -> 147.229.3.100 (Q 1/0/0/1)
2024-10-16 17:43:07 147.229.3.200 -> 100.69.167.117 (R 1/2/0/3)
2024-10-16 17:43:07 147.229.3.100 -> 100.69.167.117 (R 1/2/0/3)
```

- **Contents of `output.txt`:**

```
_xmpp-client._tcp.jabber.org
zeus-v6.jabber.org
zeus.jabber.org
```

4.10 CNAME Resolution and Logging (non verbose)

- This section uses the same packet as shown in the previous, so there is no need to display the verbose output again.
- **Program Execution:**

```
sudo ./dns-monitor -p CNAME_test.pcap -t output.txt
```

- **Output:**

```
2016-01-08 21:59:15 4.2.2.1 -> 172.16.16.154 (R 1/2/0/0)
2016-01-08 21:59:15 4.2.2.1 -> 172.16.16.154 (R 1/2/0/0)
2016-01-08 21:59:15 4.2.2.1 -> 172.16.16.154 (R 1/3/0/0)
2016-01-08 21:59:15 4.2.2.1 -> 172.16.16.154 (R 1/4/0/0)
2016-01-08 21:59:15 4.2.2.1 -> 172.16.16.154 (R 1/4/0/0)
2016-01-08 21:59:15 4.2.2.1 -> 172.16.16.154 (R 1/4/0/0)
2016-01-08 21:59:15 4.2.2.1 -> 172.16.16.154 (R 1/4/0/0)
```

- **Contents of `output.txt`:**

```
a1589.g1.akamai.net 72.246.56.83
espn.gns.go.com 199.181.133.61
a1589.g.akamai.net 69.31.75.203
gp1.wac.v2cdn.net 72.21.91.8
a1589.g1.akamai.net 72.246.56.35
redir.espn.gns.go.com 68.71.212.158
a1589.g.akamai.net 69.31.75.194
```

4.11 AAAA Resolution and Logging (non verbose)

- This section uses the same packet as shown in the previous, so there is no need to display the verbose output again.
- **Program Execution:**

```
sudo ./dns-monitor -p AAAA_test.pcap -d domain.txt -t translate.txt
```

- **Output:**

```
2024-10-13 18:36:05 100.69.162.61 -> 147.229.3.200 (Q 1/0/0/0)
2024-10-13 18:36:05 147.229.3.200 -> 100.69.162.61 (R 1/1/0/0)
```

- **Contents of** `domain.txt`**:**

```
wikipedia.org
```

- **Contents of** `translate.txt`**:**

```
wikipedia.org 2a02:ec80:600:ed1a::1
```

4.12 MX Resolution and Memory check (non verbose)

- This section uses the same packet as shown in the previous, so there is no need to display the verbose output again.
- **Program Execution:**

```
sudo valgrind --leak-check=full --track-origins=yes
--show-leak-kinds=all
./dns-monitor -p MX_test.pcap -d domain.txt
```

- **Output:**

```
2024-10-15 12:37:37 100.69.161.101 -> 147.229.3.200 (Q 1/0/0/0)
2024-10-15 12:37:37 147.229.3.200 -> 100.69.161.101 (R 1/2/0/0)

HEAP SUMMARY:
in use at exit: 0 bytes in 0 blocks
total heap usage: 55 allocs, 55 frees,
32,550 bytes allocated

All heap blocks were freed -- no leaks are possible
```

- **Contents of** `domain.txt`**:**

```
mx-in2001.wikimedia.org
mx-in1001.wiki
wikipedia.org
```

4.13 A Resolution with Interface (verbose)

- **Command:**

```
dig @8.8.8.8 example.com A
```

- **Program Execution:**

```
sudo ./dns-monitor -i eth0 -v
UNKNOWN TYPE in that case is 41 record type (which is unhandled by the p
```

- **Output:**

```
Timestamp: 2024-10-20 11:54:41
SrcIP: 172.28.71.27
DstIP: 8.8.8.8
SrcPort: UDP/34534
DstPort: UDP/53
Identifier: 0x8B9C
Flags: QR=0, OPCODE=0, AA=0, TC=0, RD=1, RA=0, AD=1, CD=0, RCODE=0

[Question Section]
example.com. IN A

[Additional Section]
UNKNOWN TYPE
====================

Timestamp: 2024-10-20 11:54:41
SrcIP: 8.8.8.8
DstIP: 172.28.71.27
SrcPort: UDP/53
DstPort: UDP/34534
Identifier: 0x8B9C
```

```
Flags: QR=1, OPCODE=0, AA=0, TC=0, RD=1, RA=1, AD=1, CD=0, RCODE=0

[Question Section]
example.com. IN A

[Answer Section]
example.com. 2935 IN A 93.184.215.14

[Additional Section]
UNKNOWN TYPE
====================
```

4.14 Testing Summary

The program was tested with all types of packets, ensuring no memory leaks occurred. It was verified both on an interface and while handling PCAP files. The correctness of the parameters was also checked, and results were validated using Wireshark to ensure accurate DNS communication monitoring.

5 Literature

References

[1] P. Mockapetris, *Domain names - Implementation and specification*, RFC 1035, November 1987. Available: `https://datatracker.ietf.org/doc/html/rfc1035`

[2] R. A. Wright, *Domain Name System Security Extensions*, RFC 2065, January 1997. Available: `https://datatracker.ietf.org/doc/html/rfc2065`

[3] R. H. Thayer, *DNS Extensions to Support IP Version 6*, RFC 3596, October 2003. Available: `https://datatracker.ietf.org/doc/html/rfc3596`

[4] “DNS Record Types,” *PhoenixNAP*, Available: `https://phoenixnap.com/kb/dns-record-types`

[5] “Tcpdump,” *Official Tcpdump Documentation*, Available: `https://www.tcpdump.org/`

[6] “What is DNS?”, *Cloudflare*, Available: `https://www.cloudflare.com/ru-ru/learning/dns/what-is-dns/`

[7] “Domain Name System,” *Wikipedia*, Available: `https://en.wikipedia.org/wiki/Domain_Name_System`

[8] A. Ozdemir, *Hash Functions and Hash Tables*, Available: `http://www.cse.yorku.ca/~oz/hash.html`