Project documentation

DNS Communication Monitoring

Network Applications and Network Administration

November 18, 2024 Vladimir Aleksandrov (xaleks03)

Contents

1 Introduction

The goal of this project was to create a program that monitors DNS communication on a chosen interface or processes DNS messages from an existing record of communication in PCAP format.

The program will process DNS messages and display information about them in console. It will also be able to find domain names and translations of domain names to IPv4/6 addresses and write them to specified files.

2 Design and Implementation

The program is divided into a single source file, dns-monitor.cpp, and a corresponding header file, dns-monitor.h. This chapter briefly describes the individual parts of the implementation.

2.1 Main Program, dns-monitor.cpp

This file contains the main function that accepts command-line arguments and processes them. If argument processing fails, the program exits with an error code.

The program then creates an instance of the DNSMonitor class, passing the parsed command-line arguments to its constructor. The DNSMonitor class is responsible for capturing and processing DNS packets.

2.2 DNSMonitor Class, dns-monitor.h and dns-monitor.cpp

The DNSMonitor class is defined in the dns-monitor.h file and implemented in the dns-monitor.cpp file.

This class captures DNS packets using the libpcap library and processes them using the resolv.h library. It also provides methods for parsing and printing DNS message sections, handling individual DNS records, and adding domains and translations to output files.

The class has several member variables that store the parsed command-line arguments, including the interface or PCAP file to capture from, the output files for domains and translations, and flags for verbose mode.

2.3 Packet Capture and Processing

The DNSMonitor class uses the libpcap library to capture DNS packets from the specified interface or PCAP file. It sets up a filter to only capture UDP packets with destination port 53, which is the standard port for DNS.

The class then enters a loop where it captures packets and processes them using the process_packet method. This method extracts the DNS data from the packet and parses it using the resolv.h library.

2.4 DNS Message Parsing and Printing

After capturing a DNS packet, the DNSMonitor class parses the DNS message into its various sections (e.g., question, answer, authority, and additional sections). The parse-

AndPrintSection method is responsible for printing these sections, either to the console or to an output file, depending on the configuration.

Each section is processed by iterating over individual DNS records, which are parsed and printed in a human-readable format. This method also distinguishes between different types of DNS records and handles them appropriately.

2.5 Error Handling

The DNSMonitor class uses error handling mechanisms to catch and handle errors that occur during packet capture and processing. It also provides a signal handler to catch signals such as SIGTERM, SIGINT, and SIGQUIT, and exit the program cleanly.

3 Testing

To verify the correctness of the program, I performed several tests using different inputs and scenarios.

3.1 Testing with Live Feedback

First, I tested the program with live feedback by running it on a network interface and observing its output. I used the following command:

```
sudo ./dns-monitor -i wlp2s0 -d domains.txt -t translations.txt -v
```

This command runs the program on the wlp2s0 interface, stores domain names in domains.txt, stores translations in translations.txt, and enables verbose mode.

I observed the program's output and verified that it correctly displays information about DNS messages, including domain names and translations.

3.2 Testing with PCAP Files

Next, I tested the program with PCAP files to verify its ability to process recorded DNS traffic. I used the following command:

```
./dns-monitor -p test_data/test1.pcap -d domains.txt -t translations.txt
```

This command runs the program on the test1.pcap file, stores domain names in domains.txt, stores translations in translations.txt, and enables verbose mode.

I compared the program's output with the expected results and verified that it correctly processes DNS messages from the PCAP file.

3.3 Verification with Wireshark

To further verify the program's correctness, I used Wireshark to capture and analyze DNS traffic on the same interface. I compared the program's output with Wireshark's output and verified that they match.

3.4 Self-Made Tests

Finally, I created several self-made tests to verify the program's behavior in different scenarios. These tests included:

Testing the program with different command-line arguments and options Testing the program with different types of DNS messages (e.g., A, AAAA, NS, etc.) Testing the program with different types of errors (e.g., invalid input, network errors, etc.)

I verified that the program correctly handles these scenarios and produces the expected output.

3.5 Handling of ANY Type in Question Section

During testing, I encountered DNS messages with a question section containing the ANY type, such as:

[Question Section] widget.intercom.io IN ANY

However, the project specification does not explicitly mention how to handle this type of query. The specification only defines the following types of records as required:

A, AAAA, NS, MX, SOA, CNAME, SRV

Since the specification does not provide guidance on handling the ANY type, I chose not to display this type of query in the program's output when running with the -v option. This decision is consistent with the project's focus on supporting the specified types of records.

4 Usage

After compiling the program, a dns-monitor executable is created.

4.1 Syntax and Semantics of Running the Program

```
./dns-monitor (-i <interface> | -p <pcapfile>) [-v] [-d <domainsfile>] [-t
<translationsfile>]
```

- `interface` - the name of the interface to listen on, or
- `pcapfile` - the name of the PCAP file to process;
- `-v` - verbose mode: display detailed information about DNS messages;
- `-d <domainsfile>` - the name of the file to store domain names;
- `-t <translationsfile>` - the name of the file to store translations of domain names to IPv4/6 addresses.

The program will display information about DNS messages on the standard output. If the -v option is specified, the program will display detailed information about DNS messages.

The program will also find domain names and translations of domain names to IPv4/6 addresses and store them in files if the -d and -t options are specified.

5 Conclusion

This DNS monitor project has been a successful implementation of a real-time DNS packet capture and analysis tool. Through this project, I have gained hands-on experience with libpcap and DNS, improving my C++ programming skills and deepening my understanding of network protocols.

The project's implementation was relatively straightforward, with a focus on parsing and analyzing DNS packets. I encountered challenges with DNS record handling and packet fragmentation, but overcame them through persistence and research.

This project has been a valuable learning experience, improving my programming skills and knowledge of network protocols. I am confident that these skills will serve me well in future projects.

References

[1] Domain Names - Implementation and Specification, [online], listopad 1987, [vid. 18. listopadu 2024]. Available: https://datatracker.ietf.org/doc/html/rfc1035.

[2] Domain Name System Security Extensions, [online], leden 1997, [vid. 18. listopadu 2024]. Available: https://datatracker.ietf.org/doc/html/rfc2065.

[3] DNS Extensions to Support IP Version 6, [online], říjen 2003, [vid. 18. listopadu 2024]. Available: https://datatracker.ietf.org/doc/html/rfc3596.

[4] cppreference.com, *std::signal*, [online], [vid. 18. listopadu 2024]. Available: https://en.cppreference.com/w/cpp/utility/program/signal.

[5] tcpdump.org, *tcpdump*, [online], [vid. 18. listopadu 2024]. Available: https://www.tcpdump.org/.

[6] linux.die.net, *inet_ntoa(3) - Linux man page*, [online], [vid. 18. listopadu 2024]. Available: https://linux.die.net/man/3/inet_ntoa.