Program pro monitorováni DNS komunikace

Autor

- Jméno: Jakub Fukala
- Login: xfukal01

- Datum vytvoření: 17.11. 2024

Obsah

1. Úvod

- Domain name system (DNS) je základ pro fungování internetu, tak jak ho známe, usnadňuje totiž převod doménových jmen na IP adresy.
- Tento dokument popisuje implementaci programu `dns-monitor`, který umožňuje sledovat komunikaci DNS na vybraném síťovém rozhraní nebo zkoumat zprávy DNS ze souboru PCAP. Sledováním této komunikace si totiz můžeme usnadnit diagnostiku problémů v síti, posuzování bezpečnosti a optimalizace výkonu sítě.
- Cílem projektu je vyvinout nástroj, který dokáže zachytávat zprávy DNS, získat z nich relevantní informace a postkytovat je uživateli ve srozumitelném formátu. Program navíc zajišťuje ukládání doménových jmen a jejich překladů na IP adresy do souborů pro následnou analýzu.

2. Přehled literatury

Pro realizaci projektu byly nastudovány následující zdroje:

- RFC 1035 [1]: "Domain names - implementation and specification" poskytuje základní informace o formátu a fungování DNS protokolu.
- RFC 2065 [2]: "Domain Name System Security Extensions" rozšiřuje DNS o bezpečnostní prvky.
- RFC 3596 [3]: "DNS Extensions to Support IP Version 6" popisuje rozšíření DNS pro podporu IPv6 adres.

[1] RFC 1035. *Domain names - implementation and specification.*
[2] RFC 2065. *Domain Name System Security Extensions.*
[3] RFC 3596. *DNS Extensions to Support IP Version 6.*

Kromě těchto standardů byly využity oficiální dokumentace a příklady k knihovnám libpcap[4] a CUnit[5], které byly použity při implementaci a testování programu.

3. Návrh aplikace

3.1 Cíle aplikace

Hlavní cíle programu dns-monitor jsou:

- Monitorovat DNS komunikaci v reálném čase nebo z existujícího záznamu.
- Zpracovávat a analyzovat DNS zprávy, včetně různých typů záznamů (A, AAAA, NS, MX, SOA, CNAME, SRV, PTR).
- Poskytovat uživateli informace ve zjednodušené nebo detailní formě.
- Umožnit ukládání spatřených doménových jmen a jejich překladů na IP adresy do souborů.

3.2 Použité technologie

- Programovací jazyk C
- Knihovna libpcap: Pro zachytávání síťových paketů.
- Knihovna CUnit: Pro jednotkové testování implementovaných funkcí.

4. Popis implementace

4.1 Struktura projektu

```
src/
    dns_monitor.c
    dns_monitor.h
    dns_parser.c
    arg_parser.c
    linked_list.c
    linked_list.h
test/
    test_main.c
pcapfiles/

Makefile
LICENSE
README.md
```

4.2 Hlavní komponenty

- `dns_monitor.c`: Obsahuje hlavní vstup programu a inicializaci zachytávání paketů.

[4] *Using libpcap in C.*
[5] *CUnit - A Unit Testing Framework for C.*

- `dns_monitor.h`: Obsahuje hlavičky funkcí sdílených mezi soubory
- `dns_parser.c`: Funkce pro zpracování DNS zpráv, včetně hlaviček a jednotlivých záznamů.
- `arg_parser.c`: Funkce pro zpracování argumentů příkazové řádky.
- `linked_list.c` / `linked_list.h`: Jednoduchá implementace vázaných seznamů pro ukládání doménových jmen a překladů.
- `test/`: Soubory pro jednotkové testy pomocí knihovny CUnit.
- `pcapfiles/` : Soubory pcap využívány pro testování

4.3 Zajímavé části implementace

4.3.1 Zpracování DNS hlavičky

Pochopení formátu hlavičky Hlavička DNS [6] je struktura pevné velikosti (12 bajtů) s následujícím uspořádáním:

Bajty	Pole	Popis
0-1	`id`	ID transakce (jedinečný identifikátor dvojice požadavek/odpověď)
2-3	`flags`	Příznaky a kódy odpovědí (QR, Opcode, ….)
4-5	`qdcount`	Počet záznamů v sekci otázek
6-7	`ancount`	Počet záznamů v sekci Odpovědi
8-9	`nscount`	Počet záznamů v oddíle Autorita
10-11	`arcount`	Počet záznamů v doplňkové sekci

```
int parse_dns_header(const unsigned char *dns_payload, uint16_t *id,
                     uint16_t *flags, uint16_t *qd_count,
                     uint16_t *an_count, uint16_t *ns_count, uint16_t *ar_count){

    // Kontrola parametrů

    // Extrakce bajtů z hlavičky
    *id = EXTRACT_16BITS(dns_payload, DNS_ID_OFFSET);
    *flags = EXTRACT_16BITS(dns_payload, DNS_FLAGS_OFFSET);
    *qd_count = EXTRACT_16BITS(dns_payload, DNS_QDCOUNT_OFFSET);
    *an_count = EXTRACT_16BITS(dns_payload, DNS_ANCOUNT_OFFSET);
    *ns_count = EXTRACT_16BITS(dns_payload, DNS_NSCOUNT_OFFSET);
    *ar_count = EXTRACT_16BITS(dns_payload, DNS_ARCOUNT_OFFSET);

    return 0;
}
```

[6] Lundrigan, Lewis. *Hands-On Network Programming with C: Learn socket programming in C and write secure and optimized network code.* O'Reilly Media, 2019.

4.3.2 Zpracování DNS záznamů

- Implementace funkce `parse_dns_rrs` umožňuje zpracovávat různé typy DNS záznamů. Pro každý podporovaný typ záznamu existuje speciální blok kódu, který extrahuje relevantní informace. Například pro záznam typu MX se zpracovává preference a název mailového serveru:

```
case 15: //MX
        {
            uint16_t preference = EXTRACT_16BITS(dns_payload, offset);
            offset += 2; // Move the offset past the PREFERENCE field
            char mx_domain_name[MAX_DOMAIN_NAME_LEN];
            int bytes_consumed = parse_domain_name(dns_payload, dns_payload_len,
                                                   offset, mx_domain_name);
            if (bytes_consumed < 0){
                fprintf(stderr, "Failed to parse MX domain name!\n");
                return -1;
            }
            if (verbose){
                printf("%u %s\n", preference, mx_domain_name);
            }
            offset += bytes_consumed;
            break;
        }
```

Podpora záznamů typu PTR - Rozšíření programu o podporu záznamů typu PTR umožňuje sledovat reverzní DNS záznamy. Implementace zahrnuje zpracování doménového jména z pole RDATA:

```
case 12: // PTR
        {
            char ptr_domain_name[MAX_DOMAIN_NAME_LEN];
            int bytes_consumed = parse_domain_name(dns_payload, dns_payload_len,
                                                   offset, ptr_domain_name);
            if (bytes_consumed < 0){
                fprintf(stderr, "Failed to parse PTR domain name!\n");
                return -1;
            }
            if (verbose){
                printf("%s\n", ptr_domain_name);
            }
            // Collect the domain name
            if (domain_list != NULL && domain_file != NULL){
                if (!(add_domain_name(domain_list, ptr_domain_name))){
                    fprintf(domain_file, "%s\n", ptr_domain_name);
                    fflush(domain_file);
                }
```

```
            }
            offset += bytes_consumed;
            break;
        }
```

Zpracování signálů - Program reaguje na signály SIGINT, SIGTERM a SIGQUIT [7] pro správné ukončení a dealokaci prostředků.

```
// Zahajení sledování singnálů v hlavním těle programu
signal(SIGINT, handle_signal);
signal(SIGTERM, handle_signal);
signal(SIGQUIT, handle_signal);

void handle_signal() {
    stop_capture = 1;
    pcap_breakloop(handle);
}
```

5. Použití programu

5.1 Kompilace Pro přeložení programu je možné využít přiložený `Makefile` v kořenovém adresáři:

```
make
```

Pro přeložení a spuštění testů je možné použít příkaz:

```
make test
```

Pozor! je však nutné mít na zařízení dostupnou knihovnu CUnit.

5.2 Spuštění

Příklady:

- Monitorování rozhraní eth0:

```
./dns-monitor -i eth0
```

- Zpracování PCAP souboru s detailním výpisem:

```
./dns-monitor -p capture.pcap -v
```

- Ukládání doménových jmen a překladů:

```
./dns-monitor -i eth0 -d domains.txt -t translations.txt
```

[7] *Handling signals in C.*

6. Testování

6.1 Jednotkové testování

- Jednotkové testování bylo prováděno pomocí jednotkových testů s knihovnou CUnit. Byly vytvořeny testy pro klíčové funkce programu:
- Arg Parser Suite: Testy ověřují správné zpracování argumentů příkazové řádky.
- DNS Header Parsing: Testy kontrolují správné parsování DNS hlaviček.
- DNS message parsing: Testy ověřují parsování různých typů DNS záznamů.
- List Suite: Testy ověřující funkci operací nad seznamy

Výstup

```
CUnit - A unit testing framework for C - Version 2.1-3
http://cunit.sourceforge.net/


Suite: DNS message parsing
  Test: Test Parse Domain Name ...passed
  Test: Test Parse Domain Name Compression ...passed
  Test: Test Parse DNS Questions ...passed
  Test: Test Parse DNS RRs A and AAAA ...passed
Suite: DNS Header parsing
  Test: Test Parse DNS Header Valid ...passed
  Test: Test Parse DNS Header Invalid ...Invalid DNS payload length!
passed
Suite: List Suite
  Test: Test Add Domain Name ...passed
  Test: Test Add Translation ...passed
Suite: Arg Parser Suite
  Test: Test Parse Arguments All Options ...passed

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      4      4    n/a      0        0
               tests      9      9      9      0        0
             asserts     26     26     26      0      n/a

Elapsed time =    0.000 seconds
```

6.2 Validace výstupu

- Pro validaci výstupu byly využívány zejmena spouštení programu se soubory pcap ze složky `pcapfiles/` s následnou ruční validací s výužitím

nástroje Wireshark [8]

7. Závěr

- Program umožňuje monitorování DNS komunikace, zpracovává požadované typy DNS záznamů a poskytuje uživateli srozumitelné výstupy.
- Pro možné zvýšené výkonu programu by bylo vhodné nahradit vázané seznamy za hašovačí tabulku a případně využít vlákna.

8. Literatura

[1] RFC 1035. *Domain names - implementation and specification.* Accessed from IETF.

[2] RFC 2065. *Domain Name System Security Extensions.* Accessed from IETF.

[3] RFC 3596. *DNS Extensions to Support IP Version 6.* Accessed from IETF.

[4] *Using libpcap in C.* DevDungeon. Accessed from DevDungeon.

[5] *CUnit - A Unit Testing Framework for C.* SourceForge. Accessed from CUnit Documentation.

[6] Lundrigan, Lewis. *Hands-On Network Programming with C: Learn socket programming in C and write secure and optimized network code.* O'Reilly Media, 2019. Accessed from O'Reilly.

[7] *Handling signals in C.* cppreference.com. Accessed from cppreference.

[8] *Wireshark - Network Protocol Analyzer.* Accessed from Wireshark.

[8] *Použití klíčového slova* `volatile` *v C.* GeeksforGeeks. Accessed from GeeksforGeeks.

[9] *Funkce* `getopt` *pro parsování argumentů.* GeeksforGeeks. Accessed from GeeksforGeeks.

[8] *Wireshark - Network Protocol Analyzer.*