Dokumentace k projektu 2

Varianta ZETA - Packet Sniffer

Tomáš Švondr – xsvond00

IPK – Počítačová komunikace a síťě

2021-2022

23. dubna 2022

Obsah

Úvod

Cílem vybrané varianty projektu ZETA byl navrhnout a implementovat síťový analyzátor, který bude schopný na určitém síťovém rozhraním zachytávat a filtrovat pakety podle protokolu, portu a rozhraní. Packety podporují zachytávání z protokolu TCP, UDP. Jako vybraný jazyk pro vypracování byl zvolen C++, zdrojový kód je uložený v souboru sniffer.cpp.

2. Použití

Překlad pomocí makefile

Získání spustitelného souboru se provede spuštění přeloženého makefile pomocí příkazu make all. Ten zdrojový kód (sniffer.cpp) přeloží za pomoci g++ (je třeba mít g++ nainstalovaný na zařízení). Příkazy ovládání makefile jsou následující:

- **make all** – spustí makefile a kompiluje zdrojový kód, výstupem je nový spustitelný soubor s názvem ipk-sniffer
- **make clean –** při spuštění příkazu „vyčistí“ současný adresář od souborů vzniklých spouštěním makefile

Spuštění programu

Samotný analyzátor se spouští zadáním následujícího řádku do konzole s administrátorskými právy (případně spustit s příznakem sudo):

./ipk-sniffer [-i *rozhraní* | --interface *rozhraní*] {-p *port*} {[--tcp|-t] [--udp|-u] [--arp] [--icmp] } {-n *num*}

Přičemž jednotlivé argumenty vykonávají následující:

- **-i | --interface** – podstoupením tohoto argumentu programu s příslušnou hodnotou (např. eth0) lze definovat právě jedno rozhraní, na které bude analyzátor odposlouchávat. Pokud tento argument není zadán, nebo mu není přiřazena hodnota hodnoty, vypíše na výstup seznam aktivních rozhraní
- **-p** – zádání tohoto argumentu s hodnotou zajistí filtrování packetů podle vybrané hodnoty portu, jak v části source, tak i v části destinaton. Pokud se zadá tento argument bez hodnoty, uvažují se všechny porty.
- **--tcp | -t** – Nastaví filtrování paketů TCP protokolu
- **--udp | -u** – Nastaví filtrování paketů UDP protokolu

- **-icmp –** Nastaví filtrování paketů ICMPv4 a ICMPv6
- **-arp** – Filtr propustí pouze pakety ARP rámce
- **-n –** určuje, kolik paketů má program zpracovat. Pokud tento argument není uvedený, zobrazí se defaultně pouze jeden packet. Pokud je hodnota n rovna 0, cyklus neustále hledá a zpracovává další pakety.

2. Implementace a popis zdrojového kódu

Využívané knihovny

Kromě knihoven datových struktur jako je například vector.h, nebo string.h využívá standardních síťových knihoven jako je např. netinet a arpa.

Nejvýznamnější používanou knihovnou je knihovna libpcap (pcap.h), která zajištuje páteřní funkce pro chod programu, především pro zachycení paketů. Při implementaci základních struktur analyzátoru jsem využíval návod „Programming with PCAP“ od autora Tim Carstens (Carstens).

Zpracování argumentů programu

K účelu uchování argumentů předaných programu při spuštění je vytvořená jednoduchá struktura settings, která uchovává informace o rozhraní, o filtrech protokolů, žádaném počtu paketů ke zpracování a o požadovaném portu pro poslech analyzátoru. Tuto strukturu plní daty funkce parseargs, která přebírá argumenty z argv a přiděluje jejich hodnoty do již zmíněné struktury.

Zpracování paketů

Zatímco o odchytávání paketů se starají funkce z knihovny pcap, následující zpracování těchto paketů obstarává funkce got_packet, která z paketu vyčte a vypíše informace o nich jako je např. zdrojová MAC adresa, IP adresa, a v případě, že paket přenáší nějaká data, vypíše i je v hexadecimálním formátu a pokud jsou data tisknutelná, vypíše je i v textovém řetězci.

Přejaté funkce

getDateFromHeader

Tuto funkci, která slouží k převodu časového razítka paketu na požadovaný čas jsem se rozhodl převzít po několikahodinovém snažení o vlastní implementaci funkce plnící tento záměr. Funkce je převzaná z internetového fóra Stack Overflow z odpovědi na otázku s titulem **Struct timeval to printable format** (Hildebrand, 2015). Autorem kódu je uživatel tohoto fóra Joe Hildebrand.

ipv6_to_str_unexpanded

Tato funkce převádí adresu ipv6 na formátovaný textový řetězec. Funkce byla převzata z internetového fóra Stack Overflow z otázky s titulem **expand an IPv6 address so I can print it to stdout** (nategoose, 2010)**.** Autorem této funkce je uživatel fóra nategoose.

3. Testování

Pro účely testování byla použita aplikace tcp-dump. A to tak, že se oba programy spustily naráz a následně se porovnávaly zachycené pakety (viz. Obrázky níže).

```
timestamp: 2022-04-24T20:28:25.848961Z
src MAC: 28:b2:bd:b4:f6:45
dst MAC: 28:b2:bd:b4:f6:45
frame length: 86 bytes
src IP: 192.168.165.248
dst IP: 192.168.165.1
src port: 48666
dst port: 53

0x0000:  0c  70  01  00  00  01  00  00  00  00  00  00  01  31  03  31   . p . . . . . . . . . . . 1 . 1
0x0010:  36  35  03  31  36  38  03  31  39  32  07  69  6e  2d  61  64   6 5 . 1 6 8 . 1 9 2 . i n - a d
0x0020:  64  72  04  61  72  70  61  00  00  0c  00  01                   d r . a r p a . . . . .
```

Obrázek 1 výstup analyzátoru ipk-sniffer

```
22:28:25.848963 IP 192.168.165.248.48666 > 192.168.165.1.domain: 3184+ PTR? 1.165.168.192
.in-addr.arpa. (44)
        0x0000:  4500 0048 b124 4000 4011 bd35 c0a8 a5f8  E..H.$@.@..5....
        0x0010:  c0a8 a501 be1a 0035 0034 a3ca 0c70 0100  .......5.4...p..
        0x0020:  0001 0000 0000 0000 0131 0331 3635 0331  .........1.165.1
        0x0030:  3638 0331 3932 0769 6e2d 6164 6472 0461  68.192.in-addr.a
        0x0040:  7270 6100 000c 0001                      rpa.....
```

Obrázek 2 výstup tcp-dump

4. Zdroje

Carstens, T. (nedatováno). *Programming with pcap.* Načteno z tcpdump.org: https://www.tcpdump.org/pcap.html

Hildebrand, J. (9. květen 2015). *struct timeval to printable format.* Načteno z Stack Overflow: https://stackoverflow.com/questions/2408976/struct-timeval-to-printable-format

nategoose. (16. září 2010). *expand an IPv6 address so I can print it to stdout.* Načteno z Stack overflow: https://stackoverflow.com/questions/3727421/expand-an-ipv6-address-so-i-can-print-it-to-stdout

Veselý, V. (nedatováno). IPK2021 – 03 – Transportní vrstva. Brno, Česká Republika.