Vysoké učení technické v Brně
Fakulta informačních technologií

Počítačové komunikácie a siete – 2. projekt
Skener sieťových služieb

23.04.2022 Adam Zvara (xzvara01)

Obsah

Úvod

Cieľom tohto projektu bolo vytvorenie aplikácie, ktorá by bola schopná oskenovať zariadenie na sieti. Skenovaním sa rozumie zistenie stavu určených portov, ktoré môžu byť otvorené, filtrované alebo zatvorené. Určenie stavu portu závisí od odpovede serveru a teda aj protokolu, ktorý bol použitý.

1 Skenovanie ako také

Skenovanie je v rámci tohto projektu rozdelené (podľa protokolov) na TCP a UDP skenovanie.

1.1 TCP skenovanie

TCP skenovanie [4] je proces, počas ktorého sú na cieľový server odosielané pakety a na základe odpovede serveru a definície TCP prokotolu [2] rozlíšené stavy portu. Odosielaný je paket, ktorý má v TCP hlavičke nastavený príznak *SYN* na nadviazanie spojenia (TCP handshake). V prípade, že je odpoveď serveru *SYN, ACK*, tak je server pripravený na danom porte zahájiť komunikáciu a port sa označí za otvorený. V prípade, že server odošle *RST*, tak server odmieta spojenie a port je zatvorený. Ostáva prípad, kedy server neodpovedá. Vtedy sa pokúsime paket odoslať znovu a v prípade, že neodpovie označíme port za filtrovaný. Celý tento proces sa nazýva *TCP Stealth Scan*, pretože odosielame iba prvý paket TCP hanshake-u.

1.2 UDP Skenovanie

UDP skenovanie [3] je jednoduchšie v tom, že nedokážeme zistiť, či je port filtrovaný. Toto skenovanie spočíva v zaslaní paketu na port skenovanému serveru a v prípade, že dostaneme odpoveď *ICMP_PORT_UNREACH* môžeme označiť port za uzatvorený. V opačnom prípade považujeme port za otvorený.

2 Postup riešenia projektu

V tomto paragrafe je popísaný celkový postup riešenia projektu, pričom jednotlivé časti sú podrobnejšie popísané v nasledujúcich podselekciách. Program je napísaný v jazyku C. Hlavná časť projektu, uložená v súbore `ipk-l4-scan.c`, začína spracovaním argumentov príkazovej riadky programu (sekcia 2.1). Po tom, čo sú argumenty spracované, sa na základe selekcie portov užívateľa volajú funkcie na TCP/UDP skenovanie. Tieto funkcie majú 3 časti:

1. odoslanie paketu (sekcia 2.2)
2. zachytenie odpovede (sekcia 2.3)
3. určenie typu odpovede (sekcia 2.4)

Po ukončení týchto funkcií sa užívateľovi vypíše stav skenovaného portu a pokračuje sa na ďalší.

2.1 Spracovanie argumentov

Spracovanie argumentov programu je realizované pomocou funkcie *getopt_long*[1]. Získané argumenty sú uložené v štruktúre *arguments*.

V ďalších častiach programu je vhodné pracovať priamo s cieľovou IP adresou a nie s doménovým menom. Preto sa kontroluje argument *domain* a v prípade, že obsahuje doménové meno je toto meno prevedené na príslušnú IP adresu (prioritne sa použije IPv4 adresa).

Ďalším problémom bolo uloženie zadaných portov. Porty môžu byť zadané jednotlivo alebo ako interval. Na rozlíšenie zadania portov existuje číselník *port_format*, ktoré rozdeľujú zadanie portov na spojité (interval) alebo diskrétne (zoznam). Pre spojitý typ je uložený začiatok a koniec intervalu, pre diskrétny sú uložené konkrétne hodnoty portov do zoznamu (oba typy uložené v štruktúre *port*).

[1] https://www.man7.org/linux/man-pages/man3/getopt.3.html

2.2 Odoslanie paketu

(IPv4) Odoslanie paketu [1] je pri oboch protokoloch v princípe veľmi podobné. Spočíva vo vytvorení raw socketu pomocou príkazu:

```
socket(AF_INET, SOCK_RAW, IPPROTO_(TCP|UDP));
```

ktorý za nás pred odoslaním paketu vyplní L2 a L3 vrstvy (Ethernet, IPv4) a na nás ostáva vyplniť hlavičku TCP alebo UDP. **(IPv6)** Zložitejšie je to pri skenovaní s využitím IPv6 adries, pretože som nenaraził na podobný spôsob vytvorenia socketu (ktorý by fungoval), preto je nutné vyplniť všetky hlavičky od L2 po L4 ručne (čo je pomerne zložité). Problémom vyplnenia Ethernetovej hlavičky je určenie cieľovej MAC adresy, ktoré sa bežne robí pomocou *Neighbor Discovery Protocol*, ktorý som neimplementoval **preto funguje skenovanie IPv6 portov len na localhoste**.

2.3 Zachytenie odpovede

Odpovede sú zachytávané pomocou knižnice *libpcap*[2]. Samotné vytvorenie *pcap handleru* prebieha už v hlavnej časti programu (`ipk-l4-scan.c`) a v jednotlivých funkciách na skenovanie sa aplikujú filtre na zachytenie odpovede.

2.4 Určenie typu odpovede

Na základe použitého protokolu sa zo získanej odpovede extrahuje TCP/UDP hlavička a kontrolujú sa hodnoty príznakov.

3 Testovanie

Program bol priebežne testovaný lokálne (localhost) ale aj na umelo vytvorenej sieti v aplikácií VirtualBox[3]

3.1 Lokálne testovanie

Na lokálne testovanie slúžil jednoduchý testovací skript `test.sh`. Tento skript otvorí vopred definované porty, ktoré sú následne oskenované pomocou vytvoreného programu. Výsledok je porovnaný s výsledkom programu *nmap*[4]

3.2 Umelo vytvorená sieť

Na testovanie skenovania iných zariadení som vytvoril jednoduchú (internú) sieť v rámci aplikácie VirtualBox. Taktiež som vytvoril 3 zariadenia v tejto sieti: **užívateľ**, ktorý skenoval ostatné zariadenia, **server** ktorý poskytoval služby na rôznych portoch a **router**, ktorý prepojoval užívateľa a server. Zariadenia obsahovali operačný systém Ubuntu Server 20.04.4. Všetkým zariadeniam som manuálne nastavil statické IP adresy a taktiež spôsoby smerovania pomocou služby Netplan[5]. Toto testovanie avšak zahrňovalo len IPv4 adresy, pretože sa mi nepodarilo ručne nakonfigurovať IPv6 sieť.

[2] Jednoduchý prehľad používania knižnice pcap https://www.tcpdump.org/pcap.html
[3] Vytváranie virtuálnych prostredí https://www.virtualbox.org/
[4] Viac o programe nmap je možne nájsť na https://nmap.org/
[5] Viac o tejto službe nájdete na https://netplan.io/examples/

Záver

Bohužiaľ tento projekt trochu zanedbáva prácu s IPv6 adresami čo sa týka samotnej implementácie (nemožno skenovať IP adresy iných zariadení) ale aj testovania (nevytvorenie siete pre testovanie IPv6).

Literatúra

[1] Buchan, D.: C Language Examples of IPv4 and IPv6 Raw Sockets for Linux. [Online], [vid. 19.04.2022].
URL https://www.pdbuchan.com/rawsock/rawsock.html

[2] Information Sciences Institute: Transmission Control Protocol. RFC 793, September 1981, doi: 10.17487/RFC0793, [vid. 18.04.2022].
URL https://www.rfc-editor.org/info/rfc793

[3] Lyon, G.: The Art of Port Scanning. [Online], [vid. 18.04.2022].
URL https://nmap.org/nmap_doc.html#port_unreach

[4] Lyon, G.: TCP SYN (Stealth) Scan (-sS). [Online], [vid. 18.04.2022].
URL https://nmap.org/book/synscan.html