VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

2. projekt z IPK: Scanner sieťových služieb

Dokumentácia

21. apríla 2019 Sabína Gregušová

Obsah

1 Úvod

Mojou úlohou bolo vytvoriť scanner sieťových služieb TCP a UDP v jazyku C alebo C++, ktorý na štandartný výstup vypíše v akom stave sa zadané porty nachádzajú, pričom možné stavy sú otvorený, uzavretý, alebo filtrovaný.

2 Riešenie

Rozhodla som sa použiť jazyk C++ a využiť jeho objektovú orientáciu. Nepoužila som funkciu na automatické parsovanie argumentov `getopt()`, pretože s ňou nemám dobré skúsenosti a preto som si vytvorila triedu, v ktorej si preparsujem vstupné argumenty sama. Pre lepšiu manipuláciu so vstupnými dátami som si vytvorila štruktúru `Ports`, ktorú používam pre oba protokoly TCP a UDP. Štruktúra `Ports` obsahuje zoznam portov, destination IP, source IP, názov interfacu, názov domény a flagy.

2.1 Parsovanie argumentov

Pri spracovávaní argumentov som brala do úvahy, že na poradí argumentov nezáleží, a preto som rozhodovala o význame argumentu podľa jeho prepínača. Každý argument okrem IP adresy alebo domény má svoj prepínač. Pri zadaní akéhokoľvek argumetu bol daný argument hneď spracovaný a uložený do štruktúry `Ports`. Najzaujímavejčšou časťou tejto imeplementáie je parsovanie IP adresy alebo domény a interfacu. Keďže je na užívateľovi, či zadá IP adresu alebo doménu, je potrebné zistiť čo bolo zadané a taktiež zistiť chýbajúci druhý údaj. Toto kontrolu robím pomocou regulárnych výrazov, kedy porovnám, či je daný údaj vo formáte `IPv4` alebo `IPv6` a ak ani jeden regulárny výraz neuspeje, považujem údaj za doménu a naopak. Interface je nepovinný argument, preto sa defaultne použije interface, ktorý je neloopbackový.

2.2 Odosielanie paketov

raw sockety, o ktorých ale na internete nie je veľa informácií ktoré by boli použiteľné a aj funkčné. Za najprínosnejší zdroj pre TCP protokol považujem [2], pretože sa zameriava práve na problematiku odosielania raw paketov a správnej tvorby IP a TCP hlavičky. Rovnaká stránka veľmi dobre poslúžila aj s článkom o [1] o UDP, podľa ktorej som vyplnila IP a UDP hlavičku.

2.3 TCP skenovanie

Najskôr vyplním IP hlavičku potrebnými údajmi a nadstavím check sum na 0. Po jej vyplnení vypočítam check sum a vyplním TCP hlavičku. V TCP hlavičke dopočítam check sum a ďalej je potrebné nadstaviť príznak `SYN` na 1, čo značí záujem o začiatok komunikácie. Obe hlavičky vložím do jedného buffera pomocou ukazateľov a pošlem pomocou raw socketu. Keďže protokol TCP očakáva odpoveď, môžu nastať nasledujúce situácie:

- príde odpoveď s nadstavenými flagmi pre `SYN` a `ACK`, čo značí, že daný port chce pokračovať v handshake-u a je teda otvorený (open)
- príde odpoveď s nadstavenými flagmi pre `RST` a `ACK`, čo značí, že daný port nechce pokračovať v handhsake-u a je teda uzavretý (closed)
- odpoveď nepríde vrámci nadstaveného timeoutu (3 sekundy), pošle sa paket znova a buď príde odpoveď, ktorá sa spracuje podľa bodov uvedených vyššie, alebo odpoveď ani po druhom timeoute (3 sekundy) nedorazí, a port je teda filtrovaný (filtered)

2.4 UDP skenovanie

Najskôr vyplním IP hlavičku potrebnými údajmi a nadstavím check sum na 0. Po jej vyplnení vypočítam check sum a vyplním UDP hlavičku. Keďže je protokol UDP jednoduchší, aj hlavička je jednoduchšia a kratšia. Protokol UDP neočakáva žiadnu odpoveď, a preto rozlišujeme iba 2 možné situácie:

- príde odpoveď protokolu ICMP a návratovým kódom 3 (destination unreachable), a daný port je označený za uzavretý (closed)
- nepríde odpoveď vrámci timeoutu (3 sekundy) a keďže protokol UDP neočakáva odpoveď, považujeme daný port za otvorený (open)

Vrámci protokolu UDP teda zisťujeme, či je daný port otvorený alebo uzvaretý, ale nevieme zistiť, či je filtrovaný.

2.5 Zachytávanie paketov

Na zachytávanie paketov som využila knižnicu `pcap` a logickú štruktúru funkcií som prebrala z [3]. Na začiatku je potrebné úspešne inicializovať a nadstaviť filter. Vrámci filtra je možné nadstaviť výraz, na základe ktorého sa budú zachytávať pakety. Rozhodla som sa nadstaviť tento výraz tak, aby ak filter niečo zachytí, bude to určite paket, ktorý očakávam. Toto nadstavenie filtra mi teda dovoluje využiť funkciu `pcap_next`, ktorá očakáva chytenie iba jedného paketu. Po využití filtra sú všetky alokované zdroje patrične uvoľnené a každé nadstavenie filtra je kontrolované pre prípadný neúspech.

3 Testovanie

Kvôli náročnosti projektu mi už bohužiaľ nezostal čas na vytvorenie automatických testov. Projekt som však aktívne testovala už počas implementácie. Pri odosielaní a prijímaní paketov som najčastejšie využívala program `Wireshark`, kde som sledovala správnosť paketov, úspečnosť ich odoslania a taktiež to, či daná doména aj odpovedá. Pri overovaní správnosti vyhodnotenia portov som využívala program `Nmap` od Gordona Lyona, ktorý prijíma na vstupe doménu a porty ktoré chceme vyhodnotiť a určí či sú dané porty otvorené, filtrované, alebo uzavreté.

4 Ukážka

Keďže je potrebné spúšťať tento program ako administrátor, nasledujúca ukážka je zo spustenia na mojom počítači s operačným systémom Ubuntu.

```
vstup:
sudo ./ipk-scan -pt 21,22,143 -pu 53,67 wis.fit.vutbr.cz

výstup:
Interesting ports on wis.fit.vutbr.cz (147.229.9.21):
PORT STATE
21/tcp closed
22/tcp open
143/tcp closed
53/udp open
67/udp open
```

5 Záver

Tento projekt považujem za jeden z najnáročnejších projektov na FIT VUT. So sieťami som pred tým nemala vôbec žiadne skúsenosti a práve o tejto špecifickej problematike použitia raw socketov bolo náročné nájsť použiteľné informácie. Napriek náročnosti projektu som rada, že som to na začiatku nevzdala, pretože mi jeho implementácia veľmi podrobne priblížila ako siete skutočne fungujú a určite som sa z neho mnohé naučila.

Použité zdroje

[1] LINUX SOCKET PART 17 Advanced TCP/IP - THE RAW SOCKET PROGRAM EXAMPLES. [online], [vid. 2019-04-12]. Dostupné z: https://www.tenouk.com/Module43a.html

[2] LINUX SOCKET PART 18 Advanced TCP/IP - OTHER TCP/IP INFO. [online], [vid. 2019-04-12]. Dostupné z: https://www.tenouk.com/Module43b.html

[3] NanoDano: Using libpcap in C. [online], rev. 2015-08-14, [vid. 2019-04-16]. Dostupné z: https://www.devdungeon.com/content/using-libpcap-c