VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MONITOROVANIE SIETE

NETWORK MONITORING

PROGRAMOVANIE SIEŤOVEJ SLUŽBY
PROGRAMMING OF NETWORK APPLICATION

AUTOR PRÁCE **JOZEF URBANOVSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN PLUSKAL**
SUPERVISOR

BRNO 2018

Abstrakt

Táto práca sa zaoberá sieťovou aplikáciou na skenovanie siete a otvorených portov. Jej realizácia je inšipirovaná voľne dostupným programom Nmap.

Abstract

This project is network application for network scanning and open ports. Realization of this work is inspired by open-source program Nmap.

Kľúčové slová

počítačové siete, monitorovanie, skenovanie, mapovanie, bezpečnosť

Keywords

networks, monitoring, scanning, mapping, security

Citácia

URBANOVSKÝ, Jozef. *Monitorovanie siete*. Brno, 2018. Programovanie sieťovej služby. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal

Monitorovanie siete

Prehlásenie

Prehlasujem, že som tento projekt vypracovával samostatne s využitím verbálnej komunikácie spolužiakov pracujúcich na tomto projekte. Uviedol som všetky literárne pramene a publikácie z ktorých som čerpal.

.....
Jozef Urbanovský
20. novembra 2017

Poďakovanie

Chcel by som poďakovať za aktivitu na fóre ako spolužiakom, tak zadávateľovi tohto projektu.

Obsah

1 Úvod **3**

2 Analýza **4**
2.1 Uvedenie do problematiky 4
2.2 Skenovanie lokálnej siete 4
2.3 Skenovanie vzdialenej siete 4
2.4 Skenovanie portov 4
2.4.1 TCP sken 5
2.4.2 UDP sken 5

3 Implementácia **6**
3.1 Spracovanie argumentov 6
3.2 Príprava skenovania 6
3.3 ARP sken 7
3.4 ICMP sken 7
3.5 TCP sken 7
3.6 UDP sken 8

4 Použitie **10**
4.1 Chybové hlášky 10
4.2 Príklady použitia a výstupu 11

5 Záver **12**

Literatúra **13**

Zoznam obrázkov

3.1 Priebeh komunikácie medzi serverom a klientom pri TCP skenovaní 8
3.2 Priebeh komunikácie medzi serverom a klientom pri UDP skenovaní 9

Kapitola 1

Úvod

Tento manuál je dokumentáciou k projektu z predmetu Sieťové aplikácie a správa sietí. Projekt sa zaoberá programovaním sieťovej služby, konkrétne monitorinovania siete. Program je inšpirovaný skenerom **Nmap** - "Network scanner", v preklade skener siete. Umožnuje objavovanie staníc a bežiacich služieb na sieti z ktorých tvorí takzvanú mapu siete. Aby bolo možné objavovať siete program konštruuje špeciálne pakety, ktoré sa odosielajú k cieľovým staniciam a na základe analýzy odpovedi určuje stav stanice a jej portov.

Projekt je riešený v jazyku C++ ako konzolová aplikácia, ktorá dostane minimálne 2 parametre na špecifikáciu siete, ktorá ma byť skenovaná. Každý ďalší prípustný argument pridáva na funkcionalite objavovať otvorené UDP alebo TCP porty, prípadne upresniť maximálny RTT packetu. Program podľa kombinácie parametrov zostrojí packety, ktoré odosiela z jednoho vlákna a príjma do druhého, z ktorého aj následne vypisuje zanalyzované výsledky.

Kapitola 2

Analýza

V tejto kapitole je analyzovaná problematika a spôsob riešenia, ktorý je použitý pri tvorbe tohto programu.

2.1 Uvedenie do problematiky

Pre ujasnenie zadania bolo nutné naštudovať ako pracuje **Nmap**, typ skenov a protokolov, ktoré používa v akých situácia a čo je cieľom analyzovať v prípade odpovedí od staníc. Špecifikácia požiadaviek pre stanice pripojené k internetu na komunikačnej vrstve[2], slúžila ako referencia pre implementáciu rôznych protokolov a analýzu odpovedí. Samotná dokumentácia a oficiálne stránky nástroju Nmap boli nápomocné na pochopenie princípov a výber protokolov[3].

2.2 Skenovanie lokálnej siete

Pre sken lokálnej siete bolo vhodné použiť protokol **ARP**, ktorý nám umožnuje objavovať stanice nezávisle na type blokovania, ktorý je v sieti aktívny. **ICMP** nie je správne riešenie, nakoľko pri existencii firewallu v lokálnej sieti by mohli byť ICMP pakety blokované a nebolo by možné spoľahlivo určiť, či je stanica aktívna alebo nie.

Na druhej strane bolo nutné rozmýšlať aj nad **ARP proxy**, ktoré môže byť prítomné na sieti, prípadne sieť, ktorá nepoužíva ethernetové rozhrania. Pokiaľ by boli použité napríklad seriálové linky implementácia ARP na objavovanie by nebola možná.

2.3 Skenovanie vzdialenej siete

Hľadanie aktívnych zariadení vo vzdialených sieťach je najefektívnejšie za použitia paketov typu ICMP, ktoré by mala podporovať každá stanica komunikujúca cez internet. ICMP sken taktiež nazývaný ping sweep pozostáva z vyslania na každú stanicu jednotlivo, nakoľko je to spoľahlivejšie ako odoslať paket len broadcastovej adrese, pretože mnoho staníc neodpovedá na broadcastové požiadavky.

2.4 Skenovanie portov

Pre objavovanie aktívnych portov na komunikačnej vrstve pre protokoly TCP a UDP bolo nutné vybrať zo širokej škály použiteľných typov objavovania v týchto protokoloch.

2.4.1 TCP sken

TCP komunikácia je založená na spoľahlivom doručovaní každého paketu, čo nám umožnuje presne vedieť, ktoré porty sú v akom stave. **TCP SYN** sken je najpopulárnejším typom skenovania otvorených portov, nakoľko je takmer neviditeľný pre zariadenia a vyžaduje odoslať len 1 paket a následne analyzovať príznaky v obdržanom pakete. Tento spôsob sa nazýva aj polootvorené skenovanie, pretože nikdy nedokončí plné TCP spojenie s cieľovou stanicou.

Pre porovnanie TCP connect sken by bol jednoduchší na implementáciu ale náročnejší na sieť, nakoľko je nutné vykonať celý "3-way handshake", v preklade trojcestné ustanovanie spojenia, ktoré by stálo o 1 odoslaný paket naviac oproti TCP SYN skenu.

2.4.2 UDP sken

UDP komunikácia je nespoľahlivá a o jej doručení nemáme žiadne informácie. Skenovanie otvorených UDP portov je postavené na spolupráci s protokolom ICMP. Pokiaľ objavovaná stanica príjme UDP paket z pohľadu skenera sa nič nestane a kvôli tomu nieje možné vedieť spoľahlivo či je port otvorený, UDP paket nebol doručený alebo odfiltrovaný. Jediné čo je možné spoľahlivo určiť je, ak skener obdrží ICMP správu typu port nedostupný.

Kapitola 3

Implementácia

Pre vytvorenie programu sú použité len základné knižnice C++. Program bol písaný v C++ kvôli efektivite práce s pamäťou a vláknami, možnosti použitia dátových typov ako reťazec a vektor. Implementácia programu je prevažne postavená na využívaní viacerých vlákien pre docielnie vyššej rýchlosti skenovania.

3.1 Spracovanie argumentov

Ako prvé `isamon` spracuje vstupné argumenty z príkazovej riadky. Pokiaľ je počet argumentov nižší ako 2 alebo je zvolená nesprávna kombinácia argumentov program je ukončený a na chybový výstup vypíše pomocnú hlášku.

V prípade správneho vstupu sú argumenty overené na validitu použitia pre program. Následne sú argumenty nahrané do globálnej štruktúry `args`, ktorá sa používa pri určení, čo a ako sa bude skenovať.

3.2 Príprava skenovania

Pre správny chod skenovania siete je nutné zistiť aké rozhrania sú použiteľné pre sken na zariadený odkiaľ sa program spúšťa. Preiterovaním cez všetky možné rozhrania určíme tie, ktoré majú IPv4 adresu a sú aktívne. Pre informácie o rozhraní ako jeho IP adresa a index rozhrania sa otvorí soket a funkcia `ioctl()` s rôznymi parametrami nám vráti potrebné dáta. Do vektoru štruktúry `if_info` je uložené aj samotné meno rozhrania na porovnanie. Zariadenia na skenovanie zahŕňajú aj prvú a poslednú IP siete, zhodne s Nmapom. Nakoľko sieť s maskou 31 v **CIDR** notácii, môže byť myslená ako peer-to-peer linka[1] a je vhodné skenovať obe stanice. Pri maske 32 alebo pri nezadaní masky vôbec sa automaticky uvažuje, že sa jedná len o jedno zariadenie.

Funkcia `hosts()`, najprv rozloží CIDR masku do štyroch oktetov a binárnej reprezentácie pre operácie ako zisťovanie IP adresy siete a broadcastu. Zadaná IP adresa siete zo vstupu je skontrolovaná či je naozaj adresou siete v kombinácii s danou maskou. Po nájdení prvej a poslednej IP adresy siete sa preiteruje cez všetky adresy, ktoré sú následne uložené do vektora v reťazcovej reprezentácii.

3.3 ARP sken

Skenovanie lokálnej siete je zabezpečené pomocou protokolu ARP, ktorý je využitý vo funkciách na odosielanie a príjmanie paketov tohto typu. Spomínané operácie sú spustené v nezávislých vláknach pre zefektívnenie skenovania. Je otvorený `RAW ARP` soket, ktorý je nastavený ako neblokujúci. Následne sa vytvorí aj pomocný soket typu `STREAM` pre zistenie MAC adresy z rozhrania momentálne používaného na sken siete.

Na odoslanie ARP paketu je nutné si zostaviť hlavičku od druhej sieťovej vrstvy OSI modelu, ktorá zahŕňa **MAC** adresu rozhrania použitého pre sken a cieľovú broadcastovú MAC adresu. Zariadenie ku ktorému posielame vykonštruovaný paket je špecifikovaný podľa cieľovej IP adresy, ktorá je nahratá do paketu. Zostavenie ARP hlavičky a jej naplnenie je implementované podľa referencie [1]. V cykle o počte staníc vektoru `ip_hosts` je odoslaný tento paket ak to jadro a vyrovnávacia pamäť momentálne umožnujú. Ak nie, funkcia sa stále pokúša odoslať paket až kým nieje uspešná a všetky pakety sú odoslané smerom k cieľovým staniciam.

V druhom vlákne je spustená funkcia na príjmanie ARP paketov. Vytvorí sa nový neblokujúci soket zhodného typu a v nekonečnom cykle sa snaží prijať odpoveď. Ak nejaký paket príjde, prečíta sa jeho hlavičku a uloží sa IP adresu do vektoru aktívnych staníc, ak má rovnakú adresu ako sieť, z ktorej zariadenia objavujeme.

3.4 ICMP sken

Vzdialené siete sú objavované protokolom ICMP, za použitia `RAW ICMP` soketu, ktorý sa vytvorí a nastaví na neblokujúci a umožní sa mu odosielať na broadcastové adresy v prípade potreby odoslať na túto adresu. Ak je zadaný aj príkaz na rozhranie, soket sa naviaže na zadané rozhranie. V hlavičke typu `icmphdr` je špecifikované, že sa má jednať o `ICMP_ECHO` správu. Nad hlavičkou sa vypočíta kontrolná hodnota, ktorá sa do nej nahrá a paket sa odošle cieľovému zariadeniu. Pokiaľ je vyrovnávacia pamäť dostupná odošle sa ihneď, ak nie, pokúša sa dookola, až kým sa odoslanie nepodarí.

Na príjmanie je otvorený zhodný paket v novom vlákne s neblokujúcim parametrom. Pokiaľ sa niečo v nekonečnom cykle niečo príjme, je využitá aritmetika ukazateľov na to aby sa z prijatej pamäti prečítala IP a ICMP hlavička a informácie v nich. Ak je zariadenie aktívne, odpovedá správou `ICMP_ECHOREPLY` a jeho IP adresa sa uloží do vektoru aktívnych staníc, pokiaľ má rovnakú adresu ako sieť, z ktorej zariadenia objavujeme.

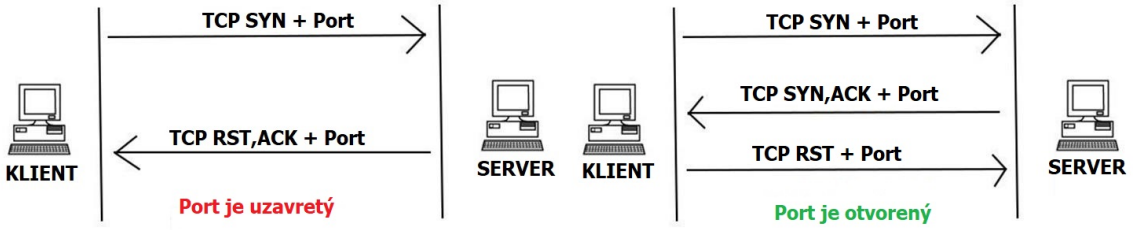
3.5 TCP sken

TCP sken je založený na odosielaní TCP SYN flagu v hlavičke na špecifikácie inicializácie komunikácie medzi zariadeniami. Na odoslanie sa vytvorí `RAW TCP` neblokujúci soket, ktorému je explicitne nastavené, aby generoval aj IP hlavičku. Pseudohlavička komunikačnej vrstvy je vytvorená aby umožnila abstrakciu nad rozdielmi v rôznych jadrách a špecifikáciach hlavičky, akú používajú. Hlavičky sa naplnia dátami ohľadom zariadenia z ktorého sa odosiela paket, stanice, ktorá je cieľová, náhodné sekvenčné číslo, protokol použitý v hlavičke a príznak SYN.

[1] https://raw.githubusercontent.com/kkrach/arp_request/master/main.c

Nad štruktúrami sa vypočíta kontrolná hodnota, ktorá sa nahrá do IP hlavičky a alokuje sa pamäť pre spojenie celého paketu na jeho odoslanie. Funkcia odošle tento paket ak je to možné vzhľadom na zdroje a vyrovnávaciu pamäť systému.

Odpoveď sa program pokúsi prijať v novom vlákne za použitia `RAW TCP` neblokujúceho soketu. Ak sa podarí prijať dáta, na základe aritmetiky ukazateľov sa rozdelí IP a TCP hlavička a vytiahnú sa potrebné dáta na určenie, či nám prišla odpoveď s príznakom `TCP SYN` a zároveň nie s `TCP RST`. Pokiaľ áno, vypisuje sa adresa z IP hlavičky a TCP port.



(a) Komunikácia pri zatvorenom porte (b) Komunikácia pri otvorenom porte

Obr. 3.1: Priebeh komunikácie medzi serverom a klientom pri TCP skenovaní

3.6 UDP sken

UDP sken je nespoľahlivý a v celku nepresný, ak je aktívne blokovanie ICMP správ, na ktorých je založený. Princíp odosielania UDP paketu je veľmi podobný TCP, s pár zmenami. Soket je vytvorený ako `RAW UDP` a žiaden kontrolný príznak sa nenastavuje, nakoľko UDP toto nastavenie nepodporuje a nepotrebuje.

Prázdny UDP paket je odoslaný na požadovaný port a adresu zariadenia v sieti, ak je to možné a sú dostupné systémové zdroje a vyrovnávacia pamäť.

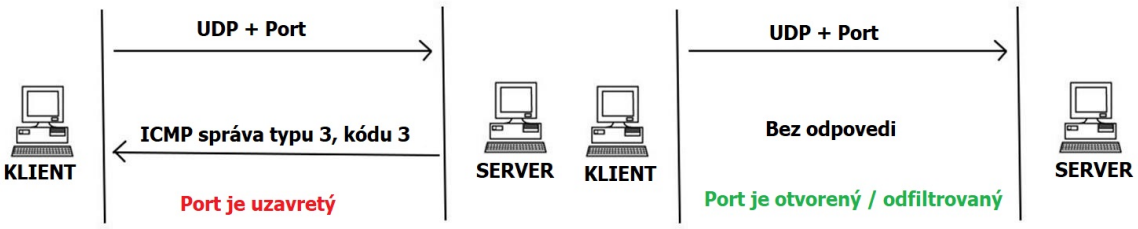
Pri príjmaní odpovede na tento paket sa vytvorí v novom vlákne `RAW ICMP` neblokujúci soket a funkcia sa pokúša prijať dáta v nekonečnom cykle. Pokiaľ `recv()` zachytí paket, je opäť využitá aritmetika ukazateľov na rozdelenie hlavičiek, ktoré sú uložené v poradí IP, ICMP, IP a UDP.

```
1           ip = (struct iphdr *) buffer;
2           icmp = (struct icmphdr *) (buffer + (ip->ihl * 4));
3           ip2 = (struct iphdr *) (icmp + 1);
4           udp = (struct udphdr *) (((uint8_t *) ip2) + (ip2->ihl * 4));
```

Výpis 3.1: Aritmetika ukazateľov nad hlavičkami protokolov

Tieto operácie sú prevedené na základe referencie[2] a sú nutné na to, aby bolo možné v jednom pakete prijať ICMP aj UDP odpoveď naraz a tak určiť kto nám odpovedal a na aký port bola táto správa určená. Ak zachytíme ICMP typu `ICMP_UNREACH` a kódu `ICMP_UNREACH_PORT` uložíme adresu z UDP odpovede do vektoru blokovaných portov. Po skončení príjmanie priterujeme cez všetky porty a vypíšeme všetky, ktoré sa nenechádzajú vo vektore blokovaných portov.

[2] https://stackoverflow.com/questions/47042355/access-udp-from-icmp-message



(a) Komunikácia pri zatvorenom porte (b) Komunikácia pri otvorenom porte

Obr. 3.2: Priebeh komunikácie medzi serverom a klientom pri UDP skenovaní

Kapitola 4

Použitie

Nastavenia a spúšťania programu:

```
isamon [-h] [-i <interface>] [-t] [-u] [-p <port>] [-w <ms>] -n <net_address/mask>
```

Kde prepínače znamenajú nasledovné:

- `-h  --help` - zobrazí nápovedu
- `-i  --interface <interface>` - rozhranie na ktorom bude zariadenie skenovať
- `-n  --network <net_address/mask>` - IP adresa s maskou siete na skenovanie
- `-t  --tcp` - použije TCP
- `-u  --udp` - použije UDP
- `-p  --port <port>` - špecifikácia skenovaného portu, inak celý rozsah
- `-w  --wait <ms>` - maximálne prípustné RTT

4.1 Chybové hlášky

Kódy, s ktorými sa program môže ukončiť pri chybe sú rôzne na základe funkcií a problémov, ktoré chybu vyvolali.

- `0 = ERR_HELP` - nesprávna špecifikácia argumentov
- `1 = ERR_PORT` - nesprávne zadaný port, mimo rozsah
- `2 = ERR_WAIT` - nesprávne zadaný časový údaj RTT
- `3 = ERR_DUPL` - duplicita argumentov
- `4 = ERR_NETW` - zadaná sieť nieje validná
- `5 = ERR_SOPT` - zlyhala funkcia setsockopt
- `6 = ERR_SOCK` - zlyhala funkcia socket
- `7 = ERR_GAIN` - zlyhala funkcia getifaddrs
- `8 = ERR_SEND` - zlyhala funkcia sendto
- `9 = ERR_IOCTL` - zlyhala funkcia ioctl
- `10 = ERR_RECV` - zlyhala funkcia recv alebo recvfrom

4.2 Príklady použitia a výstupu

`isamon -n 192.168.0.0/24` - prevedie sken siete a zobrazí aktívne zariadenia

`isamon -n 192.168.0.0/24 -t` - prevedie sken siete, zobrazí aktívne zariadenia a ich otvorené TCP porty

`isamon -n 192.168.0.0/30 -i eth0` - prevedie sken siete za použitia rozhrania eth0

`isamon -n 192.168.0.0/28 -t -u -w 10` - prevedie sken siete a a zobrazí aktívne zariadenia s ich otvorenými TCP a UDP portami s maximálnych prípustným RTT

`isamon -n 192.168.0.0/30 -t -p 80` - prevedie sken siete a vypíše aktívne zariadenia a či majú otvorený TCP port 80

Príklad výstupu:

`192.168.1.1` - aktívny klient

`192.168.1.1 TCP 80` - otvorený TCP port 80

`192.168.1.1 TCP 22` - otvorený TCP port 20

`192.168.1.1 UDP 53` - otvorený UDP port 53

Kapitola 5

Záver

Program sa dá úspešne použiť na zistenie aktívnych klientov v sieti a ich otvorené porty pre protokol TCP alebo UDP. Aplikácia je limitovaná veľkosťou vyrovnávacej pamäti a nastaveniami jadra na spracovávanie paketov.

Testovanie prebehlo úspešne na systémoch Linux, za použitia lokálneho stroja Ubuntu 17.04 a virtuálneho stroja CentOS 7 s využitím vagrantu.

Pri programovaní tejto aplikácie som sa naučil lepšie pracovať s druhou až štvrtou vrstvou ISO OSI modelu a pochopil princípom skenovania siete a portov.

Literatúra

[1] Alvaro Retana, D. W.: *Using 31-Bit Prefixes on IPv4 Point-to-Point Links*. RFC 3021, RFC Editor, December 2000.
URL `https://tools.ietf.org/html/rfc3021`

[2] Braden, R.: *Requirements for Internet Hosts – Communication Layers*. RFC 1122, RFC Editor, October 1989.
URL `https://tools.ietf.org/html/rfc1122`

[3] Lyon, G.: *Nmap Network Scanning*. Insecure Press, January 2009, ISBN 978-0-9799587-1-7.
URL `http://nmap.org/book/`