Fakulta informačních technologií
Vysoké učení technické v Brně

Počítačové komunikace a sítě – 2. projekt

Scanner síťových služeb

Marin Klobušický (xklobu03) 21.4.2019

Obsah

1 Úvod **2**

2 Implementácia **2**
2.1 Argumenty 2
2.2 Kontroly a Konverzie 2
2.3 UDP port scanner 3
2.4 TCP port scanner 3

3 Testovanie **3**

4 Záver **4**

5 Použitá literatúra **5**

1 Úvod

Program dostane zadané porty a adresu, buď v podobe doménového mena alebo IP adresy .Voliteľným argumentom je interface z ktorého bude prebiehať skenovanie. Program oskenuje zadané porty a rozhodne či sú `Open` , `Closed` alebo `Filtered`. V prípade nezadania aspoň jedného portu alebo cielovej adresy sa to považuje za chybu.

2 Implementácia

Pri implementácií som sa snažil čo najviac vecí vymyslieť sám a naštudovať si toho čo najviac a myslím že sa mi to v celku podarilo. Dosť veľa vecí má inšpiráciu na internete ale vždy som bol schopný to pochopiť a prepísať tak aby dané funkcie robili presne to čo potrebujem. Môj program obsahuje kombináciu C/C++ snažil som sa čo najväčšiu časť písať v C++ ale občas som bol nútený použiť nejakú funkciu z C.

2.1 Argumenty

Na spracovanie argumentov som použil vlastnú logiku kde v argumentoch hľadám `-i` , `-pu` a `-pt` a argumenty za nimi kontrolujem či sú v správnom formáte, tj. za `-pu` a `-pt` musí byť buť číslo alebo čísla oddelené čiarkou alebo rozmedzie čísel.

2.2 Kontroly a Konverzie

Na začiatku programu prevádzam všetky kontroly, či sú všetky zadané hodnoty vo formáte ako má byť a či je zadaný argument `-i`, ak nieje tak si pomocou funkcie `get_interface_IP` zistím adresu prvého interface ktorý nieje lookback. Ak je potrebné tak prevádzam doménové meno na IP adresu pomocou funkcie `net_dns_resolve`. Môj projekt má 3 rôzne návratové kódy:

- 10: Chyba argumentov programu
- 255: chyba pri konverziách hostname na IPV4
- 1: Chyba nadstavenia socketu alebo packetu

Tiež rozpoznávam či je zadaná adresa `IPv4` alebo `IPv6` na `IPv4` používam jednoduchý regex a na `IPv6` mám funkciu `get_interface_IPV6` ktorá vráti bool hodnotu.

2.3 UDP port scanner

Celý scanner je v cykle ktorý číta hodnoty z vektoru ktorý je naplnený všetkými portami ktoré mám skenovať až pokiaľ neprejde všetky porty. Najskôr sa vytvoria potrebné štruktúry pre IP a UDP hlavičky, následne inicializujem `pcap` ktorým budem neskôr zachytávať prichádzajúce packety. Otvorím si `RAW` socket a nadstavím IP a UDP hlavičku, pri IP robím aj checksum. Tu pri plnení hlaviček pomocou funkcii `htons` a `htonl` musím previesť čísla na Big endian s ktorým pracuje sieťovy provoz. Potom si nainicializujem packet a nadstavím filter aby som prijal len packety ktoré skutočne hľadám. Na koniec odosielam 3 packety pomocou príkazu `sendto()` a čakám na odpoveď. Ak ani raz nepríde žiadny packet tak port prehlásim za otvorený, ak dostanem nejakú odpoveď tak si packet analyzujem a opäť prevádzam z Big endian na Little endian v a zistím či je jeho `type` a `code` 3. Ak áno tak to znamená že som obdržal `ICMP` packet a port prehlásim za zatvorený.

2.4 TCP port scanner

Prvá časť prebieha dosť podobne ako `UDP`. najskôr si nainicializujem štruktúry pre IP , TCP a pseudo hlavičku. Nadstavím si `pcap` a otvorím socket, naplním IP , TCP a pseudo hlavičky s tým že teraz počítam checksum aj pre TCP hlavičku k čomu použijem pseudo hlavičku. Teraz posielam jediný packet a čakám na odpoveď. Ak nepríde žiadna odpoveď tak pošlem ďalší packet a ak ani na ten nepríde odpoveď tak port prehlásim sa filtrovaný. Ak príde na packet odpoveď tak prijatý packet analyzujem a ak má `syn` a `ack` nadstavený na 1 znamená to že port chce komunikovať a prehlásim ho za otvorený, ak má `ack` a `rst` nadstavený na 1 znamená to že port nechce komunikovať a prehlásim ho za zatvorený.

3 Testovanie

Svoj program som testoval či už v lokálnej sieti alebo na webe a svôje výstupy som si porovnával s nástrojom `Angry IP Scanner` ktorý je open source a myslím že som nenarazil na prípad kde by sa naše výstupy líšili z toho môžem usúdiť že môj program pracuje pomoerne spolahlivo. Testoval som tiež rôzne zadávania nezmyselných argumentov a snažil som sa ošetriť čo najviac vecí. Kedže som sa do poslednej chvíle snažil iplementovať komunikáciu cez `IPv6` ktorú som bohužial nezvládol tak som nemal ani moc času na testovanie ale myslím že som ošetril všetko čo som mohol vzhľadom na situáciu.

4 Záver

Na záver by som chcel povedať že pri projekte som sa neskutočne naučil a hlavne som pochopil ako skutočne tieto vecí pracujú a uvedomil som si aké je to zložité ako som už spomenul tak `IPv6` môj program síce rozpozná ale nedokáže komunikovať, pretože mi to prišlo zložitejšie a mal som príliš málo času. Rozhodne to bol veľmi časovo náročný projekt ale dúfam že veci ktoré som zvládol budú fungovať perfektne.

5 Použitá literatúra

C++ cross platform resolve hostname to ip library. Gamedev [online]. 2015 [cit. 2019-04-21]. Dostupné z: https://www.gamedev.net/forums/topic/671428-c-cross-platform-resolve-hostname-to-ip-library/

How to get IPV6 interface address using getifaddr() function. Stackoverflow [online]. 14.10.2015 [cit. 2019-04-21]. Dostupné z: https://stackoverflow.com/questions/33125710/how-to-get-ipv6-interface-address-using-getifaddr-function

Rawtcpsocket. Github [online]. 20.11.2012 [cit. 2019-04-21]. Dostupné z: https://github.com/rbaron/raw$_t cp_s ocket/blob/master/raw_t cp_s ocket.c?fbclid = IwAR2F9aeTIH5t_7 1gBgdx7i3gn4gdlhiC7J7T_y QonwrCnNJ8mjLetZ7nAts$

LINUX SOCKET PART 17. Tenuok [online]. [cit. 2019-04-21]. Dostupné z: https://www.tenouk.com/Module43a.html

C++ Check Valid IP Address IPv4 IPv6. Zedwood [online]. [cit. 2019-04-21]. Dostupné z: http://www.zedwood.com/article/cpp-is-valid-ip-address-ipv4-ipv6