VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové komunikace a sítě – 2. projekt

Jednoduchý síťový TCP, UDP skener v C/C++

21. dubna 2019 Jakub Frejlach (xfrejl00)

Obsah

1 Úvod

Tento program byl napsán jako 2. projekt do předmětu Počítačové kumunikace a sítě. Jeho cílem bylo implementovat jednoduchý TCP, UDP skener v jazyce C/C++, který na dané doméně/adrese oskenuje požadované porty a poskytne na standardní výstup informace o jejich stavu. Pro UDP skenování může být stav jeden z dvojice open/closed, pro TCP skenování pak jeden z trojice open/closed/filtered.

1.1 Základní pojmy

- **UDP** – nebo-li User Datagram Protocol, je jeden ze standardní sady protokolů pro komunikaci přes síť. Je velmi často označován jako nespolehlivý, protože nedává záruku na doručení dat.[1]
- **TCP** – nebo-li Transmission Control Protocol, je nejpoužívanější protokol pro komunikaci přes síť, zajišťuje obousměrné a spolehlivé posílání dat.[2]
- **paket** – blok dat přenášený v počítačové síti. Skládá se z řídících a uživatelských dat.[3]
- **síťový soket** – je koncový bod při připojení přes počítačovou síť.[4]
- **raw soket** – speciální přístup k soketu, který umožňuje programátorovi přímý přístup k odesílání a přijímání paketů. [5]

2 Implementace

Většina použitých knihoven je pro jazyk C, tudíž C++ kunstrukce, jako například řetězce nebo vektory, byly použity pouze pro usnadnění práce s daty a tím pádem bylo možno částečně vynechat klasickou práci s pamětí z jazyka C.

2.1 Parsování argumentů

Na parsování argumentů byla použita funkce `getopt_long_only` a knihovna `regex.h`. Parsování bylo implemetováno pomocí konstrukce `switch` a užitím globálních flagů.

2.2 Získání IP adress cíle a rozhraní

Pomocí několika funkcí z knihoven `arpa/inet.h`, `net/if.h`, `ifaddrs.h` a `netdb.h`. Jejich užití bylo převzato z linuxových manuálových stránek.

2.3 Posílání a přijímání paketů pro IPv4 adresy

Před odesíláním paketů je nutno zinicializovat `pcap handle`, který bude zachytávat packety, tento handle je je nastaven do okamžitého módu, aby se nezacyklil při nezachycení odpovědi. Do tohoto handlu je dále vždy při zpracovávání určitého portu zkompilován filtr, který zaručí, že daný handle bude zachytávat pouze pakety z očekávané adresy, z očekávaného portu a pouze očekávaného protokolu.

Dále je podle typu protokolu vytvořen raw soket a IP/TCP/UDP hlavičky jsou naplněny daty. Tyto hlavičky jsou poskládány do paketu a je nutno spočítat kontrolní součty. Pro UDP protokol stačí spočítat kontrolní součet pro IP hlavičku, pro TCP protokol je nutno navíc spočítat i kontrolní součet pro TCP hlavičku, který je počítán s využitím pseudo hlavičky [6]. Funkce na kontrolní součet a vyplňování hlaviček bylo převzáno z doporučeného studijního materiálu v zadání. [7] Paket je odeslán pomocí funkce `sendto`, a opověď je přijímána funkcí `pcap_next_ex`.

V případě UDP skenování je při přijmutí odpovědi port označen jako closed a v případě nepřijmutí odpovědi jsou provedeny ještě další dva pokusy pro odeslání, poté je teprve paket opravdu vyhodnocen jako open.

V případě TCP skenování jsou při přijmutí odpovědi na paket namapovány Ehternet, IP a TCP hlavičky a z nich jsou zkontrolovány flagy. Konkrétně pro kombinaci flagů `SYN` a `ACK` je port označen jako open a pro kombinaci flagů `ACK` a `RST` je port označen jako closed. V případě nepřijetí odpovědi je proveden další pokus o odeslání a přijmutí a poté je port případně vyhodnocen jako filtered.

2.4 Posílání a přijímání paketů pro IPv6 adresy

Průběh je až na několik změn velmi podobný skenování IPv4 adres. Příšlusné struktury jsou nahrazeny jejich alternativami pro IPv6 adresy. Kontrolní součty jsou počítány automaticky pomocí parametru `IPV6_CHECKSUM` ve funkci `setsockopt` při vytváření soketu. Do paketu je umístěna pouze TCP/UDP hlavička, IP hlavička je doplněna automaticky kernelem. Filtry je nutno nastavit ručně a dopočítat se po jednotlivých bytech k jednolivým informacím v paketu. Vše ostatní probíhá stejně jako u IPv4.

3 Testování

Aplikace byla otestována především na localhostu a na doméně nemeckay.net, která patří spolužákovi, který testování povolil. Proti výstupům mého programu byly na ukázku postaveny výstupy prograu `nmap`. Níže je uvedeno několik ukázek testování programu.

3.1 Testování TCP IPv4

```
$ sudo ./ipk-scan -pt 20,80,1000,1001,1002,1003 nemeckay.net
Interesting ports on nemeckay.net
Target IPv4 address: 46.28.109.159
PORT     STATUS
20/tcp   filtered
80/tcp   open
1000/tcp         filtered
1001/tcp         closed
1002/tcp         closed
1003/tcp         filtered

$ sudo nmap -p 20,80,1000,1001,1002,1003 nemeckay.net

Starting Nmap 7.60 ( https://nmap.org ) at 2019-04-21 11:43 CEST
Nmap scan report for nemeckay.net (46.28.109.159)
Host is up (0.0083s latency).

PORT      STATE     SERVICE
20/tcp    filtered  ftp-data
80/tcp    open      http
1000/tcp  filtered  cadlock
1001/tcp  closed    webpush
1002/tcp  closed    windows-icfw
1003/tcp  filtered  unknown

Nmap done: 1 IP address (1 host up) scanned in 1.57 seconds
```

3.2 Testování UDP IPv4

```
$ sudo ./ipk-scan -pu 20,80,1000,1001,1002,1003 nemeckay.com
Interesting ports on nemeckay.com
Target IPv4 address: 184.168.221.49
PORT     STATUS
20/udp   open
80/udp   open
1000/udp         open
1001/udp         open
1002/udp         open
1003/udp         open

$ sudo nmap -p 20,80,1000,1001,1002,1003 -sU nemeckay.com

Starting Nmap 7.60 ( https://nmap.org ) at 2019-04-21 21:13 CEST
Nmap scan report for nemeckay.com (184.168.221.49)
Host is up (0.25s latency).
rDNS record for 184.168.221.49: ip-184-168-221-49.ip.secureserver.net

PORT      STATE          SERVICE
20/udp    open|filtered  ftp-data
80/udp    open|filtered  http
1000/udp  open|filtered  ock
1001/udp  open|filtered  unknown
1002/udp  open|filtered  unknown
1003/udp  open|filtered  unknown

Nmap done: 1 IP address (1 host up) scanned in 4.09 seconds
```

3.3 Testování TCP IPv6

```
$ sudo ./ipk-scan -pt 75-80 ::1 -i lo
Interesting ports on ::1
Target IPv6 address: ::1
PORT     STATUS
75/tcp   closed
76/tcp   closed
77/tcp   closed
78/tcp   closed
79/tcp   closed
80/tcp   open
```

```
$ sudo nmap -6 -sT -p 75-80 ::1

Starting Nmap 7.60 ( https://nmap.org ) at 2019-04-21 21:23 CEST
Nmap scan report for ip6-localhost (::1)
Host is up (0.000056s latency).

PORT   STATE  SERVICE
75/tcp closed priv-dial
76/tcp closed deos
77/tcp closed priv-rje
78/tcp closed vettcp
79/tcp closed finger
80/tcp open   http

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
```

3.4 Testování UDP IPv6

```
$ sudo ./ipk-scan -pu 1-5 ::1 -i lo
Interesting ports on ::1
Target IPv6 address: ::1
PORT     STATUS
1/UDP    closed
2/UDP    closed
3/UDP    closed
4/UDP    closed
5/UDP    closed

$ sudo nmap -6 -sU -p 1-5 ::1

Starting Nmap 7.60 ( https://nmap.org ) at 2019-04-21 21:18 CEST
Nmap scan report for ip6-localhost (::1)
Host is up (0.000040s latency).

PORT  STATE  SERVICE
1/udp closed tcpmux
2/udp closed compressnet
3/udp closed compressnet
4/udp closed unknown
5/udp closed rje

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

4 Závěr

Projekt byl časově relativně náročný a to zejména kvůli velkému množství nových knihoven, které je nutno pečlivě nastudovat. Možnost využít C++ namísto pouhého C mě velice příjemně překvapila. A i přes zdárné dokončení projektu jsem se nevyhnul frustrujícím momentům, kdy zkrátka jistá věc nefungovala a trvalo několik hodin, než se povedlo neočekávané chování programu opravit.

Zdroje

[1] Wikipedia, "User Datagram Protocol — Wikipedia, the free encyclopedia." http://cs.wikipedia.org/w/index.php?title=User%20Datagram%20Protocol&oldid=16954683, 2019. [Online; accessed 21-April-2019].

[2] Wikipedia, "Transmission Control Protocol — Wikipedia, the free encyclopedia." http://cs.wikipedia.org/w/index.php?title=Transmission%20Control%20Protocol&oldid=17119785, 2019. [Online; accessed 21-April-2019].

[3] Wikipedia, "Paket — Wikipedia, the free encyclopedia." http://cs.wikipedia.org/w/index.php?title=Paket&oldid=17077982, 2019. [Online; accessed 21-April-2019].

[4] Wikipedia, "Síťový socket — Wikipedia, the free encyclopedia." http://cs.wikipedia.org/w/index.php?title=S%C3%AD%C5%A5ov%C3%BD%20socket&oldid=16954662, 2019. [Online; accessed 21-April-2019].

[5] Wikipedia, "Raw socket — Wikipedia, the free encyclopedia." http://cs.wikipedia.org/w/index.php?title=Raw%20socket&oldid=16406576, 2019. [Online; accessed 21-April-2019].

[6] C. M. Kozierok, "Tcp checksum calculation and the tcp "pseudo header." http://www.tcpipguide.com/free/t_TCPChecksumCalculationandtheTCPPseudoHeader-2.htm, 2019. [Online; accessed 21-April-2019].

[7] Tenouk, "Linux socket part 17 advanced tcp/ip - the raw socket program examples." https://www.tenouk.com/Module43a.html, 2019. [Online; accessed 21-April-2019].