VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentace k druhému projektu předmětu IPK 2018/2019

Varianta OMEGA: Scanner síťových služeb

Autor projektu: Martin Macháček
Login autora: xmacha73

20. dubna 2019

Obsah

1 Úvod

Tato dokumentace popisuje zadaný projekt, zdroje čerpání obecných informací, implementační detaily projektu, testovací výstupy, zajímavé pasáže projektu a testování. Cílem projektu je funkční program napsaný v jazyce C++, který s použitím TCP nebo UDP skenuje uživatelem zadané porty na dané IP adrese / doméně přes dané síťové rozhraní a vypíše stavy portů

2 Použitá literatura a vědomosti

Pro implementaci tohoto projektu bylo potřeba mít vědomosti o obecném principu funkčnosti paketů, využitých protokolů a jejich programovanou reprezentaci.

Informace o TCP, IP, UDP jsem čerpal nejvíce z přednášek a následujících odkazů

- https://en.wikipedia.org/wiki/Transmission_Control_Protocol
- https://en.wikipedia.org/wiki/User_Datagram_Protocol
- https://study-ccna.com/ip-header/
- https://tools.ietf.org/html/rfc792

3 Překlad a spuštění projektu

Projekt lze přeložit s využitím souboru `Makefile`. Je nutné mít nainstalovaný g++ překladač. Pro maximální zaručení funkčnosti překládejte a spouštějte aplikaci na operačním systému Linux, distribuci Ubuntu. Následující příkaz spustí aplikaci.

```
./ipk-scan <args>
```

Kde `<args>` značí argumenty, které uživatel může zadat. Takové argumenty jsou následující:

- `-pu <port-ranges> -pt <port-ranges> <address> -i <interface>`
- `-help`

V případě využití prvního způsobu spuštění je:

- `-i <interface>` volitelný příkaz pro volbu síťového rozhraní
- `-pt <port-ranges>` příkaz pro vymezení portů pro TCP skenování
- `-pu <port-ranges>` příkaz pro vymezení portů pro UDP skenování
- `<address>` název domény nebo IP adresa určena ke skenování

Tyto porty jsou omezené rozmezím 0-65535, kde formáty <port-ranges> můžou být následující (konkrétní čísla jsou pouze uvedena pro příklad):

- `65-8899`
- `1,2,98,500,1000,80`
- `8`

4 Implementace

Program je rozdělen do několika částí. Pokud program narazí na problém, vypíše odpovídající chybovou zprávu, a ukončí se s odpovídající chybovou hodnotou.

4.1 Argumenty

Program začne kontrolou argumentů. Tento krok je proveden kombinací cyklu `for` a několika podmínek kontrolující podobu argumentu. Po provedení kontroly si program roztřídí zadané porty, adresu a případné rozhraní do vlastních proměnných. Pro snazší práci s poli pro porty jsem využil `vector<int>`.

Rozhraní, pokud není zadáno argumentem `-i`, pod kterým se bude komunikovat, je přiděleno pomocí funkce `getFirstNonLoopbackI`, která najde všechny dostupné rozhraní a podle příznaku `IFF_LOOPBACK` se rozhodne, která rozhraní přiřadí.

4.2 Příprava a následné odeslání paketu

Po rozřazení argumentů se pro každý zadaný port provede funkce `connectViaTCP` nebo `connectViaUDP`. Tyto funkce nejprve převedou formát cílové adresy a adresy rozhraní na formát potřebný v hlavičkách TCP, IP, UDP. Dále si program vytvoří hlavičky a soket, které poté naplní potřebnými daty.

Struktury hlaviček jsou importované z knihoven `netinet`. Konkrétně `netinet/ip.h`, `netinet/tcp.h`, `netinet/udp.h`. Pro TCP je potřeba navíc deklarovat a vyplnit pseudo hlavičku a pseudo paket ze kterých poté program vypočítá kontrolní součet [1][2]. Program poté inicializuje proměnnou typu `pcap_t` na zachytávání příchozích paketů, nastaví jí filtr [3].

4.3 Zachycení příchozího paketu

Nyní je program připraven na odeslání paketu. Po odeslání pomocí funkce `sendto` je nastaven časový limit, do kterého musí být příchozí paket funkcí `pcap_loop` zachycen. Pokud do časového limitu paket nepřišel, spustí se funkce `handle`, která situaci vyhodnotí. Pokud vypršel čas při TCP komunikaci, je paket pro jistotu poslán znovu a teprve poté je port vyhodnocen jako filtrovaný.

Zachycený paket, pokud posíláme pakety pomocí TCP, se poté přetypuje na potřebný typ `tcphdr`, ze kterého můžeme zjistit vlastnosti paketu. Těmito vlastnostmi se myslí hlavně přepínače ACK a RST. Pokud totiž paket má příznak ACK a RST, lze port označit jako uzavřený. Pokud paket má pouze příznak ACK, lze takový port označit za otevřený. Pokud po odeslání paketu přes UDP zachytíme paket typu ICMP, lze daný port označit jako uzavřený, ve všech ostatních případech jako otevřený.

5 Zajímavé situace při implementaci

Velice zajímavým problémem bylo nalezení prvního rozhraní bez loopbackové IP. Nevěděl jsem totiž, že existuje funkce na nalezení takového rozhraní, tudíž jsem musel napsat funkci vlastní.

Další zajímavá situace se naskytla při opětovném odesílání paketu na zjištění filtrovaného portu. Situace je vyřešena přes časovač a globální `bool` proměnnou, podle které se cyklus rozhoduje, jestli má paket poslat znovu nebo ne.

6 Testování

Testování tohoto programu probíhalo porovnáváním výstupu mého projektu s open source skenerem Nmap[4]. Níže uvedené screenshoty jsou výpisy jak mého projektu, tak Nmap aplikace. Nad každým výstupem je vždy uveden příkaz, pod kterým byl tento projekt spuštěn.

Výstupy mého projektu **Výstupy Nmap**

```
sudo ./ipk-scan -pt 60,70,80,631 localhost -i lo
```

```
-----------------------------------------------------------
Interface used:         lo
Destination address:    localhost
-----------------------------------------------------------
PORT            STATE
60/tcp          Closed
70/tcp          Closed
80/tcp          Closed
631/tcp         Opened
```

```
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000071s latency).
PORT    STATE  SERVICE
60/tcp  closed unknown
70/tcp  closed gopher
80/tcp  closed http
631/tcp open   ipp

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
```

```
sudo ./pk-scan -pt 22,80,1000,1001,1002,1003 nemeckay.net
```

```
-----------------------------------------------------------
Interface used:         enp0s3
Destination address:    nemeckay.net
-----------------------------------------------------------
PORT            STATE
22/tcp          Opened
80/tcp          Opened
1000/tcp        Filtered
1001/tcp        Closed
1002/tcp        Closed
1003/tcp        Filtered
```

```
Nmap scan report for nemeckay.net (46.28.109.159)
Host is up (0.27s latency).
PORT     STATE    SERVICE
22/tcp   open     ssh
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 4.14 seconds
```

```
sudo ./ipk-scan -pt 132-140 192.168.56.1
```

```
-----------------------------------------------------------
Interface used:         enp0s3
Destination address:    192.168.56.1
-----------------------------------------------------------
PORT            STATE
132/tcp         Closed
133/tcp         Closed
134/tcp         Closed
135/tcp         Opened
136/tcp         Closed
137/tcp         Filtered
138/tcp         Closed
139/tcp         Opened
140/tcp         Closed
```

```
Nmap scan report for 192.168.56.1
Host is up (0.70s latency).
PORT    STATE    SERVICE
132/tcp closed   cisco-sys
133/tcp closed   statsrv
134/tcp closed   ingres-net
135/tcp open     msrpc
136/tcp closed   profile
137/tcp filtered netbios-ns
138/tcp closed   netbios-dgm
139/tcp open     netbios-ssn
140/tcp closed   emfis-data

Nmap done: 1 IP address (1 host up) scanned in 7.58 seconds
```

```
sudo ./ipk-scan localhost -i lo -pu 0,1,68,80,100,500,631
```

```
-----------------------------------------------------------
Interface used:         lo
Destination address:    localhost
-----------------------------------------------------------
PORT            STATE
0/udp           Closed
1/udp           Closed
68/udp          Opened
80/udp          Closed
100/udp         Closed
500/udp         Closed
631/udp         Opened
```

```
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000060s latency).
PORT    STATE         SERVICE
0/udp   closed        unknown
1/udp   closed        tcpmux
68/udp  open|filtered dhcpc
80/udp  closed        http
100/udp closed        unknown
500/udp closed        isakmp
631/udp open|filtered ipp

Nmap done: 1 IP address (1 host up) scanned in 1.49 seconds
```

7 Závěr

Na tomto projektu jsem se naučil opravdu hodně věcí, ovšem kvůli nedostatku času na ostatní projekty jsem nebyl schopen dokončit podporu IP adres verze 6. Je ale také velká škoda, že druhý projekt nebyl inspirován projektem prvním, a třeba na něj nenavázal ve větším měřítku..

Tento projekt mě opravdu donutil přemýšlet o tom, jak bych měl být vděčný za programovací jazyky jako Python, nebo za již hotové knihovny a funkce, díky kterým si nemusím dva týdny škubat vlasy a proklínat celý svět. Na druhou stranu jsem detailně pochopil to, co jsem například na přednáškách částečně nechápal, tudíž můžu říct, že tento projekt byl opravdu přínosný.

8 Reference

[1] "Silver Moon". *C Packet Sniffer Code with libpcap and linux sockets* [online] [28.4.2009] [cit. 21.4.2019]. Dostupné na: https://www.binarytides.com/raw-sockets-c-code-linux/

[2] "CNoob". *Raw Socket - Recv/Send Problem* [online] [21.1.2009][cit. 21.4.2009]. Dostupné na: http://www.cplusplus.com/forum/general/7109/

[3] "Tim Carstens, Guy Harris". *Programming with pcap* [online] [cit. 21.4.2009]. Dostupné na: https://www.tcpdump.org/pcap.html

[4] https://nmap.org/