Vysoké učení technické v Brně
Fakulta informačních technologií

Počítačové komunikace a sítě

Projekt č.2

21. dubna 2019 Martin Chládek (xchlad16)

Obsah

1 Zadání

Naší úlohou bylo vytvořit jednoduchý TCP/UDP scanner. Tento program si dává za úkol oskenovat zadanou IP adresu (ve formátu IPv4, IPv6 nebo doménového jména) a zadané porty. Na základě formátu příchozích packetů implementace označí dané porty jako otevřené/zavřené/filtrované a vypíše jejich stav na standardní výstup. Úloha mohla být řešena v jazyce C nebo C++. Já jsem si pro realizaci svého projektu zvolil programovací jazyk C.

2 Teorie řešení

Ke komunikaci mezi zdrojovým počítačem a cílovou stanicí je nezbytná adresace jednotlivých stran. Ta je uskutečněna pomocí IP adresy (IPv4 nebo IPv6) a portu, kde každý port má přiřazeno své identifikační číslo, které musí spadat do intervalu <1; 65535>. Některé z portů mají přiřazeny služby[1]. Stanice mezi sebou komunikují zasíláním tzv. paketů. K zobrazení odeslaných a přijímaných paketů lze použít počítačovou aplikaci Wireshark[2]. Způsoby navázání komunikace užitím protokolů TCP a UDP jsou popsány v následujících podkapitolách.

2.1 TCP skenování

Ke zjištění stavu portu pomocí TCP protokolu není nutný kompletní 3-way-handshake. Zdrojový počítač odešle paket s nastaveným příznakem SYN (synchronization flag)[4]. Existují pouze 3 scénáře, jak se může cílová stanice zachovat:

1. Odešle paket s nastavenými flagy SYN a ACK (Acknowledgment). V takovémto případě na daném poru běží nějaká služba a program označí daný port za otevřený.

2. Odešle paket s nastavenými flagy ACK a RST (Reset). Cílová stanice nás informovala, že nemá zájem participovat v komunikaci a můžeme tak daný port označit za uzavřený.

3. Program od stanice nedostane žádnou odpověď. Tehdy se aplikace zachová dle zadání tak, že zkusí celý proces opakovat a pošle ještě jeden totožný paket. Pokud se ani tentokrát nedočká odpovědi, označí daný port za filtrovaný.

2.2 UDP skenování

U UDP protokolu probíhá komunikace značně jinak. Protokol UDP nenavazuje spojení, skenování portu probíhá zasláním paketu a čekáním na odpověď. V případě, že zdrojový počítač obdrží ICMP paket typu 3 ("Destination Unreachable"), označí implementace port jako uzavřený. V opačném případě pokud nedetekuje žádný příchozí paket vyhovující požadavkům, označí port jako otevřený.

3 Popis implementace

Má implementace by se dala z hlediska funkcionality programu rozdělit do 2 logických částí: zpracování argumentů a skenování portů. Obě tyto části budou podrobněji popsány v následujících podkapitolách. V implementaci je využíváno open source aplikačního rozhraní pro odchytávání síťové komunikace `pcap`, konkrétně v implementaci pro unixové systémy – `libpcap`[3].

V programu je využíváno struktur jako jsou `ip`, `ip6_hdr`, `tcphdr`, `udphdr`, `sockaddr_in`, `sockaddr_in6` které nabízí knihovny `netinet`[6]. Struktury pseudo hlaviček pro kontrolní součet byly implementovány ručně.

Zdrojový port je v rámci celého programu neměnný a je definován jako konstanta.

[1] Seznam portů s přiřazenými službami lze nalézt zde: `<https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers>`

[2] Více informací k paketovému analyzátoru Wireshark je k dispozici na `<https://www.wireshark.org/>`

3.1 Zpracování argumentů

Zpracování argumentů je pod taktovkou funkce `parse_arguments`, která kontroluje formát vstupních parametrů a zajišťuje jejich načtení. Pro načítání a zpracování vstupních argumentů je využívána funkce `getopt_long_only`, která načítá parametry začínající pouze '-'[1]. Argumenty i případé flagy o jejich přítomnosti jsou načítány do struktury `Arguments`, se kterou je poté pracováno napříč celým programem.

3.2 Skenování

Skenování použitím obou protokolů zastřešuje funkce `scan`, přičemž využívá několik pomocných funkcí. Zajišťuje případnou konverzi zadaného doménového jména. Samotná konverze je realizována funkcí `getaddrinfo`. Očekává se, že funkce na konverzi doménového jména uloží do struktury `Arguments` cílovou IP adresu a také zda se jedná o IPv4 nebo IPv6. Následně je hledána na zadaném (případně vyhledaném) rozhraní zařízení adresa typu, který je shodný s cílovou adresou. V případě, že adresa není nalezena, funkce `scan` zažádá o vyhledání cílové adresy opozitního typu a hledá poté adresu daného typu i v rozhraní zdrojové stanice. Pokud neuspěje ani tentokrát, program vyhodnotí, že není schopen navázat komunikaci a ukončí se s chybou vypsanou na standardní chybový výstup. Následně volá funkce `TCP_scan_IPvX` a/nebo `UDP_scan_IPvX`, kde X značí verzi IP adresy.

Všechny tyto funkce pro skenování využívají pomocné funkce `open_socket`. Její hlavní úlohou je otevření a aplikování nastavení na daný socket, který předává odkazem. Implementuje přitom dvě zajímavé funkce a to sice funkci `socket` a `setsockopt`. První zmíněná vytváří koncový bod (angl. endpoint), což je síťový uzel, který slouží jako zdroj nebo cíl pro komunikaci. Druhá uvedená poté nastavuje pro daný socket konkrétní parametry[6].

3.2.1 TCP skenování

Velká část funkce pro skenování portů pomocí TCP protokolu je tvořena cyklem `while`. Jako ukončující podmínku využívá tento cyklus funkci `get_port_number`.

Tato funkce, jak už název vypovídá, vrací jeden konkrétní port z těch, co byly zadány jako argument, v případě že, již byly všechny porty projity, vrací -1. Zajímavé na ní však je, jakým způsobem porty získává. Funkce ponechává zadané porty v řetězci, jednotlivé porty získává dvojím způsobem na základě toho, zda byl zadán interval či výčet hodnot. V obou případech uchovává hodnoty, kde přestala pracovat, ve struktuře `Indexes`.

- Byl zadán interval: Funkce postupuje od spodní hranice intervalu a inkrementuje hodnotu o 1. Pozici v intervalu, kde funkce skončila, uchovává ve struktuře `Indexes`, ze které bude číst při opětovém zavolání funkce. Při každém volání je tento postup opakován až do dovršení horní hranice intervalu.
- Byl zadán výčet: S využitím funkce `atoi`, je z řetězce vytržena hodnota, vypočítán počet znaků hodnoty a původní index inkrementován právě o počet znaků + 1 (započítání oddělující čárky). Od tohoto indexu v řetězci bude příště postup opakován. Dodržení formátu obstarává jiná pomocná funkce.

V každé iteraci cyklu je naplněna TCP a IP hlavička pro aktuální parametry. Mění se pouze cílový port a s tím související kontrolní součet, který je nutný přepočítávat při každé změně v hlavičce[5]. K výpočtu kontrolního součtu u TCP hlavičky je využita pseudo TCP hlavička[7]. Vše je uloženo do pole charů `datagram`, který je funkcí `sendto` zprostředkován a odeslán na cílový socket[2].

Síťový provoz je poté filtrován pomocí filtru, jež specifikuje porty, na kterých je očekávána komunikace. Zachycení paketu zajišťuje funkce `pcap_dispatch`, která je nastavena na zachycení právě jednoho packetu a vyvolání obslužné rutiny ve formě funkce `packet_handler_TCP`, která daný paket analyzuje. V případě, že program není schopen zachytit žádnou odpověď cílové stanice, je za účelem zamezení zamrznutí programu implementován časovač, který zajišťují funkce `signal` a `alarm`.

Ve chvíli, kdy implementace není schopna zachytit paket po dobu jedné sekundy, přestává čekat na paket a je celý proces snahy o navázání spojení opakován. Pokud se ani tentokrát nedočká odpovědi ze strany cílové stanice, je port označen za filtrovaný, v opačném případě je paket analyzován a na základě nastavených flagů je port označen jako otevřený/zavřený.

3.2.2 UDP skenování

Princip implementace UDP skenování je podobný jako u TCP skenování popsaném v předchozí kapitole. Liší se pouze ve dvou ohledech. Místo TCP hlavičky jsou operace prováděny nad UDP hlavičkou a poté ve způsobu komunikace. UDP protokol očekává jako odpověď ICMP typu 3, aby mohl port prohlásit za uzavřený. Pokud žádná odpověď nepřijde, považujeme port za uzavřený. Nemůžeme tedy určit, zda je port filtrovaný.

4 Překlad aplikace

Pro překlad aplikace slouží přiložený soubor `Makefile`.. Postup lze tedy přeložit příkazem `make`:

```
$ make
```

Pro vzorové fungování aplikace lze `Makefile` využít ke spuštění následujícím příkazem:

```
$ make run
```

Pro správné spouštění je na systémech linux potřebné oprávnění správce (případné spouštění aplikace příkazem `sudo` dodá potřebná oprávnění).

5 Spuštění aplikace

Aplikace vyžaduje pro spuštění následující parametry:

- `-pt` Parametr pro skenování skrze **TCP** protokol. Musí být doplněn hodnotami portů ve formátu intervalu nebo výčtu.
- `-pu` Parametr pro skenování skrze **UDP** protokol. Musí být doplněn hodnotami portů ve formátu intervalu nebo výčtu.
- `-i` Parametr pro výběr rozhraní. Není li zvolen, zvolí se první IEEE 802 interface, který má přidělenou neloopbackovou IP adresu.
- `domain_name/IPaddress` Doménové jméno nebo IP adresa cílové stanice.

Alespoň jeden z parametrů `-pt` nebo `-pu` je povinný. Stejně tak musí být zadáno doménové jméno, nebo IP adresa (ve verzi 4 nebo 6). Rozhraní, které zadává parametr `-i` je volitelné.

Příklad spuštění:

```
$ ./ipk-scan -pt 22,23,80,1000 -pu 20-35 -i enp0s3 vutbr.cz
```

6 Testování aplikace

Aplikace byla testována ručně na operačním systému Ubuntu ve verzi 18.04.2 LTS Bionic Beaver. Pro sledování síťového provozu byl využit program Wireshark.

Použitá literatura

[1] [online].
URL <http://www.gnu.org/software/libc/manual/html_node/Getopt-Long-Options.html#Getopt-Long-Options>

[2] [online]. [cit. 2019-04-21].
URL <https://linux.die.net/man/2/>

[3] Reference Manual Pages (3PCAP). [online]. 2018 [cit. 2018-04-21].
URL <https://www.tcpdump.org/manpages/pcap.3pcap.html>

[4] TCP Flags. [online]. 2018 [cit. 2019-04-20].
URL <https://www.keycdn.com/support/tcp-flags>

[5] Tenouk. [online]. [cit. 2019-04-21].
URL <https://www.tenouk.com/Module43a.html>

[6] Kerrisk, M.: Man7 Linux manual. [online]. 2019 [cit. 2019-04-20].
URL <http://man7.org/linux/man-pages/man2/>

[7] Kozierok, C. M.: The TCP/IP Guide. [online]. 2005 [cit. 2019-04-21].
URL <http://www.tcpipguide.com/free/t_TCPChecksumCalculationandtheTCPPseudoHeader-2.htm>