VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové komunikace a sítě - projekt 2
Scanner síťových služeb

21. dubna 2019 Karel Hanák

Obsah

1 Zadání

Úkolem bylo vytvořit aplikaci, která provede sken zadané IP adresy a portů. Aplikace dostane doménu, ze které vyhodnotí IP adresu, nebo dostane přímo IP adresu. Dále dostane seznam či rozsah portů, které pro dané zařízení oskenovat, vyhodnotit stav daných portů a vypsat jej na standardní výstup.

2 Protokoly

2.1 Internet Protocol

Internet protocol je navržen pro komunikaci systémů a zařízení v počítačové síti. Jeho úkolem je přesun datagramů skrze množinu sítí postupně od odesílatele ke příjemci. Datagramy nebo také pakety se skládají z fragmentu a dat. Fragment obsahuje informace potřebné k doručení paketu zatímco data představují přesouvané informace [7]. Každý paket obsahuje IP adresu odesílatele i příjemce, které slouží jako jednoznačné identifikátory síťových rozhraní a používají se pro směrování [3].

2.2 Transmission Control Protocol

Transmission control protocol je jedním z TPC/IP protokolů a slouží k navázání spojení mezi počítači připojenými do sítě a k spolehlivému obousměrnému přenosu dat. Protokol zajišťuje doručení všech paketů a ve správném pořadí. Spolehlivost zajišťuje odesíláním paketů, které byly poškozeny či ztraceny, dokud skutečně nedojdou od odesílatele k příjemci nebo dokud nevyprší časový limit a spojení je ukončeno. Je vhodný pro použití, když je nutné aby všechna data byla doručena vyšší režie není problém [5].

2.3 User Datagram Protocol

User datagram protocol je také jedním z TCP/IP protokolů, který podobně jako TCP slouží k přenosu datagramů od odesílatele k příjemci s tím rozdílem, že UDP nezaručuje doručení paketů příjemci. Oproti TCP jej nezajímá, zda je paket správně doručený, pouze jej odešle pokud je v pořádku. Je vhodný pro použití, když nám jde o rychlost přenosu dat a menší režii [6].

3 Skenování portů

Skenování portů je metoda zjišťování stavu síťových portů zařízen v počítačové síti. Rozlišují se 3 základní stavy portů [4]:

1. Otevřený - Stav portu, na němž nějaká služba naslouchá a čeká na připojení klienta.
2. Zavřený - Stav portu, na němž neběží žádná služba, a tedy skrze něj nebude možné navázat spojení.
3. Filtrovaný - Stav portu, kdy host neposkytl žádnou odpověď.

V tomto projektu nás zajímá pouze SYN a UDP skenování.

3.1 SYN skenování

SYN skenování je forma TCP skenování, ale narozdíl od klasického TCP skenování nedochází k navázání skutečného spojení. Místo toho aplikace pro skenování portů vytvoří SYN packet a odešle jej. Pokud testovaný port otevřený, odešle zpět paket s příznaky SYN a ACK. Pokud se nám vrátí RST paket, tak se jedná o zavřený port. A nakonec pokud se nevrátí žádná odpověď, jde o filtrovaný port [4].

3.2 UDP skenování

UDP skenování je docela odlišné, protokol UDP není navržen pro navazování spojení a tak neexistuje UDP ekvivalent TCP SYN, pokud je však odeslán UDP paket a systém odpoví s ICMP port unreachable zprávou, testovaný port se považuje za zavřený. O tom zda je port otevřený touto metodou nelze vždy správně rozhodnout [4].

4 Implementační detaily

Ve zdrojovém kódu jsou uvedeny reference k převzatým částem kódu včetně odkazu pro bližší kontext.

4.1 Zpracování vstupních argumentů

Zpracování vstupních argumentů je prováděno pomocí funkce `getopt_long_only`[1], která narozdíl od předchozích verzí funkce registruje argumenty s – i −− a usnadňuje tak zpracování víceznakových krátkých zápisů možností. Pro kontrolu formátů zápisu argumentů s porty se používají regulární výrazy. U výčtu argumentů se poté získaný validní řetězec rozdělí podle oddělovače a do vektoru celých čísel uloží číslo portu. Obdobně je to tak i v případě rozsahu portů, s tím rozdílem že stačí kraje rozsahů použít jako řídící proměnné cyklu který vloží porty do příslušného vektoru.

4.2 IP adresa cílového zařízení

Řetězec obsahující IP adresu nebo doménu se zkontroluje regulárním výrazem, pokud se jedná přímo o IP adresu, tak se pomocí funkce `inet_aton` [2] převede na strukturu, se kterou se později pracuje. Pokud je však zadána doména, je potřeba ip adresu získat, k tomu slouží funkce `getaddrinfo` [3].

4.3 IP adresa zdrojového rozhraní

Rozhraní je volitelný urgument specifikovaný uživatelem, pokud je zadán název rozhraní, tak se pomocí funkce `getifaddrs` [4] získá seznam rozhraní a hledá se v něm rozhraní s odpovídajícím názvem. Pokud rozhraní není specifikováno, tak se použije stejná funkce k vyhledání prvního neloopbackového rozhraní.

4.4 SYN skenování

Nejpve se otevře socket pro UDP protokol a připraví se pro odeslání paketu. Dále se vytvoří 2 buffery, do kterých se budou mapovat pakety a vynulují se. Následuje nastavení zachytávání příchozích paketů, k zachytávání paketu se využívá knihovna `pcap` [5]. Nejprve se vytvoří handle pro zachytávání paketů, poté se sestaví filtrovací řetězec, který se zkompiluje do struktury, se kterou pak dokáže knihovna dále pracovat. Pro handle se nastaví zkompilovaný filtr a tím je připravený na zachytávání paketů.

Po vytvoření filtru je třeba sestavit samotný paket, který budeme odesílat. Dříve získané struktury obsahující IP adresy zdrojového rozhraní a cílového zařízení se doplní o porty a specifikuje se IPv4 protokol. Následuje vytvoření a naplnění struktur. Z knihoven `netinet/ip.h` a `netinet/tcp.h` se využijí struktury pro IP a TCP hlavičky. Hlavičky se naplní metadaty pro odeslání a je potřeba provést kontrolní výpočet pro TCP paket. V případě TCP paketu se také vytvoří pseudo-TCP hlavička, která simuluje hlavičku IP paketu [1]. Obě struktury jsou uloženy za sebou v jednom z dříve vytvořených bufferů. Z těchto dvou naplněných struktur se

[1] Linux man page getopt_long_only `https://linux.die.net/man/3/getopt_long_only`
[2] Linux man page inet_aton `https://linux.die.net/man/3/inet_aton`
[3] Linux manual page getaddrinfo `http://man7.org/linux/man-pages/man3/getaddrinfo.3.html`
[4] Linux man page getifaddrs `https://linux.die.net/man/3/getifaddrs`
[5] Manpage of PCAP `https://www.tcpdump.org/manpages/pcap.3pcap.html`

provede kontrolní výpočet a pseudo-TCP dále není potřeba. Do druhého bufferu se uloží skutečná IP hlavička a za ni skutečná TCP hlavička. Tentokrát se provede výpočet kontrolního součtu pro IP.

Po přípravě hlaviček a výpočtu kontrolních součtů se paket odešle skrze socket a předem připravený handle s filtrem následně zachytává příchozí pakety od zařízení, na které se TCP paket odesílal. Na přijetí odpovědi je timeout nastaven na 2 sekundy, pokud do té doby byl zachycen žádný paket, provede se celá iterace znovu a pokud ani podruhé nepřijde žádný paket, je port označen jako filtrovaný. Zachycený paket se namapuje zpět na struktury IP a TCP hlaviček a podle metadat se určí stav portu a vypíše se na standardní výstup. Tento postup je pak opakován pro každý port.

4.5 UDP skenování

UDP skenování probíhá velmi podobným způsobem s několika málo rozdíly. Otevíraný socket se nastaví pro protokol UDP. Zachytávání paketů probíhá stejným způsobem, ale s jiným filtrovacím řetězcem. Ve filtrovacím řetězci se specifikuje ICMP odpověď typu unreachable port, která přijde od cílového zařízení zpět na zdrojové.

Příprava hlaviček pro vytvoření paketu je opět obdobná, akorát se místo TCP hlavičky použije UDP hlavička. IP a UDP hlavičky se naplní a pro IP se vypočítá kontrolní součet. Pro UDP hlavičku u protokolu IPv4 není nutné kontrolní součet vypočítávat [2].

Po naplnění hlaviček a výpočtu kontrolního součtu se paket opět odešle krze socket a opět se čeká na příchozí data. Tentokrát čekáme na ICMP zprávu port unreachable, pokud taková zpráva dojde, port je označen jako zavřený, jinak je port označen jako otevřený a nakonec je jeho stav vypsaný na standardní výstup. Zde se paket neodesílá znovu. Tento postup je pak opakován pro každý port.

5 Testování

Testování bylo provedeno na poskytnutém virtuálním stroji a zároveň na osobním stroji. K ověření funkčnosti byl použity nástroje `Nmap` [6] a `Wireshark` [7]. Na osobním stroji se testovaly porty na localhost z loopback interface, stejně tak na virtuálním stroji. Wireshark byl použit ke sledování odchozích a příchozích paketů, Nmap pro stav portů.

[6] Nmap `https://nmap.org`
[7] Wireshark `https://www.wireshark.org`

6 Použití literatura

[1] Checksum computation, Transmission Control Protocol. [online], 2019, [cit. 2019-04-21] Dostupné z: https://en.wikipedia.org/wiki/Transmission_Control_Protocol#Checksum_computation.

[2] Checksum computation, User Datagram Protocol. [online], 2019, [cit. 2019-04-21] Dostupné z: https://en.wikipedia.org/wiki/User_Datagram_Protocol#Checksum_computation.

[3] Internet Protocol. [online], 2019, [cit. 2019-04-21] Dostupné z: https://cs.wikipedia.org/wiki/Internet_Protocol.

[4] Port Scanner. [online], 2019, [cit. 2019-04-21] Dostupné z: https://en.wikipedia.org/wiki/Port_scanner.

[5] Transmission Control Protocol. [online], 2019, [cit. 2019-04-21] Dostupné z: https://cs.wikipedia.org/wiki/Transmission_Control_Protocol.

[6] User Datagram Protocol. [online], 2019, [cit. 2019-04-21] Dostupné z: https://cs.wikipedia.org/wiki/User_Datagram_Protocol.

[7] Information Science Institute, U. o. S. C.: RFC 791: Internet Protocol. [online], 1981, [cit. 2019-04-21] Dostupné z: https://tools.ietf.org/html/rfc791.