VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové komunikace a sítě

Dokumentace 2. projektu

21. dubna 2019

Kamil Vojanec

Obsah

1 Úvod

Předmětem druhého projektu do předmětu Počítačové komunikace a sítě (dále jako IPK) je vytvořit jednoduchý *port scanner*. Port scanner je program, který testuje stav portů na zadaném stroji. Implementované metody skenování portů jsou *TCP-SYN* a *UDP-ICMP* (viz níže). Stavy portů mohou být následující [1]:

- Open – port je otevřený a naslouchá na něm nějaká aplikace
- Closed – port je dostupný, nenaslouchá na nich ale žádná aplikace
- Filtered – port je filtrovaný, tzn. není dostupný, pakety mířící na tento port jsou zahozeny

2 Popis řešeného problému

2.1 TCP-SYN skenování

Principem TCP-SYN skenování je zaslání TCP segmentu s nastaveným příznakem SYN a čekání na odpověď. V běžném provozu je příznak SYN využíván jako žádost o navázání TCP spojení s cílovým strojem na daném portu. Pro přijetí žádosti cíl odpoví TCP segmentem s nastavenými příznaky SYN a ACK. Pro zamítnutí žádostí cíl odpovídá segmentem s příznakem RST. Dále by následovalo potvrzení přijetí žádosti zdrojovým zařízením.[2]

Při TCP-SYN skenování však nedochází k plnému navázání spojení. Namísto toho je pouze odeslán segment s příznakem SYN a v případě, že cílový stroj odpoví segmentem s příznaky SYN, ACK, prohlásíme daný port za otevřený. Narozdíl od normální komunikace však zdroj nepotvrzuje přijetí žádosti, namísto toho posílá segment s pžíznakem RST, což oznamuje cíli, že žádost o navázání spojení se ruší. V případě, že cíl odpovídá segmentem s příznakem RST, prohlásíme port za uzavřený. Poslední možností je, že cíl vůbec neodpoví, v tom případě prohlásíme cílový port za filtrovaný.[3]

2.2 UDP-ICMP skenování

UDP je jednoduchý protokol bez záruky doručení. Vyžaduje malou režii a narozdíl od TCP nenavazuje spojení.[4]

ICMP je protokol používaný operačními systémy pro služební účely. ICMP zprávy jsou zasílány například při chybách zasílání datagramů nebo chybách směrování.[5]

Principem UDP-ICMP skenování je zaslání UDP datagramu. Na základě odpovědi (nebo absence odpovědi) můžeme rozhodnout v jakém stavu se port nachází. Jelikož UDP protokol nenavazuje spojení, neočekáváme odpověď ve formě UDP, nýbrž očekáváme ICMP zprávu. Pokud operační systém cílového stroje odpoví ICMP zprávou typu 3, kódu 3 (*ICMP port unreachable error)*, je cílový port prohlášen za uzavřený. Naopak, neodpoví-li cílový stroj žádnou zprávou, může být cílový port buď otevřený nebo filtrovaný.[6]

3 Implementace

3.1 Celkový popis implementace

Program byl implementován v jazyce *C++*, což umožňuje využití standardních knihoven operačních systémů na bázi *UNIX*. Pakety jsou odesílány pomocí *raw soketů*[7], které dovolují vyplnit hlavičku paketu na programátorské úrovni. Z důvodu nemožnosti přijímat pakety na úrovni raw soketů byla využita knihovna *libpcap*, která obsauje funkce a datové struktury nutné k odchytávání příchozích paketů.

Před skenováním, a to jak TCP-SYN, tak UDP-ICMP, je vyplněna IP hlavička. Ta se při běhu programu dále nemění. Pro její vyplnění je nejprve potřeba zjistit adresu cílového stroje, a rovněž adresu zdrojového rozhraní. Také je nastaven zdrojový port. Ten je vybrán náhodně funkcí `rand`[8] v rozmezí 49152 a 65535. Tento rozsah je zvolen jako rozsah dočasných portů specifikovaných organizací *IANA*[9]. Dále je vytvořen

raw soket a je nastaven (pomocí `setsockopt`[10]) tak, aby zamezil operačnímu systému přidávat k odeslaným paketům IP hlavičky, ty si totiž program doplní sám. Následně je inicializováno přijímání paketů funkcí `pcap_open_live`[11]. Následuje již samotné skenování.

3.2 Implementace TCP-SYN skenování

Pro TCP-SYN skenování byla implementována funkce `scan_TCP`, která skenuje vždy právě jeden port. Tato funkce je poté volána v cyklu pro každý zadaný cílový port. Při každém volání je znovu přepočítán kontrolní součet dle RFC1071[12]. Pro výpočet kontrolního součtu je použita tzv. *pseudo hlavička* pro IPv4[2] a IPv6[13]. Po sestavení hlavičky TCP segmentu, kde je vyplněn zdrojový a cílový port a také je nastaven příznak SYN, je celý paket odeslán funkcí `sendto`[14]. Před samotným odchytáváním příchozího paketu je připravena obslužná funkce signálu `SIGALRM`, která zajistí přerušení odchytu paketu po určitém čase (tzv. *timeout*).

Následně se zavolá funkce `pcap_next`[15], která zachytí vždy právě jeden paket. Zde je využito filtru příchozích paketů[16], který poskytuje knihovna libpcap. Tento filtr umožní ze všech příchozích paketů vybrat jen ty, které jsou pro program zajímavé. V tomto případě se jedná o pakety s ze zdrojovou adresou odpovídající cílovému zařízení, cílovou adresou odpovídající zdrojovému rozhraní, cílovým portem odpovídajícím zdrojovému portu aplikace a zdrojovým portem odpovídajícím cílovému portu aplikace.

V případě zachycení paketu, který filtrem projde, je tento paket prozkoumán. Program hledá stav příznaků SYN, ACK a RST. Význam těchto příznaků je popsán výše. V případě, že je vyvoláno přerušení signálem `SIGALRM`, je odchytávání paketů přerušeno, odešle se ještě jeden stejný paket a proces se opakuje. Pokud ani po odeslání druhého paketu není doručena žádná odpověď, prohlásí se paket za filtrovaný.

3.3 Implementace UDP-ICMP skenování

UDP-ICMP skenování bylo implementováno velico podobným principem, jako TCP-SYN skenování. Byla implementována funkce `scan_UDP`, která je volána v cyklu pro každý cílový port. Výpočet kontrolního součtu se případě UDP omezil jen na potřebu IPv6[13], pro IPv4 není potřeba počítat kontrolní součet UDP. Odesílání a zachytávání paketů je řešeno stejně jako v případě TCP-SYN. Rozdíl je jen ve tvaru filtru příchozích paketů, kde je kromě zdrojové a cílové adresy přidán výraz kontrolující typ a kód příchozí ICMP zprávy. V případě, že není odchycena žádná ICMP zpráva, je tento postup opakován ještě čtyřikrát. Celkem tedy může být posláno až 5 UDP datagramů. Význam přijetí nebo nepřijetí ICMP zprávy je popsán výše.

4 Testování

Program byl testován zejména na lokálním stroji `localhost`. Dále byl využit server `http://nemeckay.net/`, jehož skenování bylo povoleno majitelem serveru (jedná se o spolužáka). Problémem bylo testování IPv6 serverů, toto bylo testováno pouze na lokálním stroji.

4.1 Srovnání s nástrojem *NMAP*

Pro skenování portů existuje známý program NMAP. Tento program je šířen pod licencí GNU GPL[17] a je open source. Ve srovnání s programem vytvořeným pro projekt do předmětu IPK (dále jako *ipk-scan*) je NMAP několikanásobně výkonnější, celý rozsah portů 0-65535 dokáže prozkoumat ve velmi krátkém čase. S pomocí nástroje `time` bylo na TCP-SYN skenování všech portů `localhostu` naměřen čas 6,43 sekundy. Pro srovnání, program ipk-scan za stejnou dobu prozkoumá 6 portů.

Shodnou vlastností nástoje NMAP a programu ipk-scan je využití knihovny libpcap pro odchytávání příchozích paketů a rovněž využití implementačního jazyka *C++*.

5 Závěr

Program ipk-scan je sice silně neefektivní variantou port scanneru, je však jednoduchý a pro malý počet portů funguje stejně jako obecně známější nástroje.

Reference

[1] *Port Scanning Basics, Nmap network scanning*. [Online], [cit. 21.4.2019].
Dostupné z: `https://nmap.org/book/man-port-scanning-basics.html`

[2] Postel, J.: Transmission Control Protocol. RFC 793, RFC Editor, Září 1981, doi:10.17487/RFC0793, [Online], [cit. 21.4.2019].
Dostupné z: `https://www.rfc-editor.org/rfc/rfc793.txt`

[3] *TCP SYN (Stealth) Scan (-sS), Nmap network scanning*. [Online], [cit. 21.4.2019].
Dostupné z: `https://nmap.org/book/synscan.html`

[4] Postel, J.: User Datagram Protocol. RFC 768, RFC Editor, Srpen 1980, doi:10.17487/RFC0768, [Online], [cit. 21.4.2019].
Dostupné z: `https://www.rfc-editor.org/rfc/rfc768.txt`

[5] Postel, J.: Internet Control Message Protocol. RFC 792, RFC Editor, Září 1981, doi:10.17487/RFC0792, [Online], [cit. 21.4.2019].
Dostupné z: `https://www.rfc-editor.org/rfc/rfc792.txt`

[6] *UDP Scan (-sU), Nmap network scanning*. [Online], [cit. 21.4.2019].
Dostupné z: `https://nmap.org/book/scan-methods-udp-scan.html`

[7] *SOCKET(2) Linux Programmer's Manual*. Září 2017.

[8] *RAND(3) Linux Programmer's Manual*. Srpen 2017.

[9] Cotton, E. L. T. J. W. M., M.; Cheshire, S.: Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335, RFC Editor, Srpen 2011, doi:10.17487/RFC6335, [Online], [cit. 21.4.2019].
Dostupné z: `https://www.rfc-editor.org/rfc/rfc6335.txt`

[10] *GETSOCKOPT(2) Linux Programmer's Manual*. Září 2017.

[11] *PCAP_OPEN_LIVE(3PCAP) Manual Page*. Leden 2014.

[12] Braden, B. D., R.; Partridge, C.: Computing the Internet checksum. RFC 1071, RFC Editor, Září 1988, doi:10.17487/RFC1071, [Online], [cit. 21.4.2019].
Dostupné z: `https://www.rfc-editor.org/rfc/rfc1071.txt`

[13] Deering, S.; Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, RFC Editor, Prosinec 1998, doi:10.17487/RFC2460, [Online], [cit. 21.4.2019].
Dostupné z: `https://www.rfc-editor.org/rfc/rfc2460.txt`

[14] *SEND(3) Linux Programmer's Manual*. Září 2017.

[15] *PCAP_NEXT_EX(3PCAP) Manual Page*. Duben 2014.

[16] *PCAP-FILTER(7) Miscellaneous Information Manual*. Srpen 2018.

[17] *Legal Notices, Nmap network scanning*. [Online], [cit. 21.4.2019].
Dostupné z: `https://nmap.org/book/man-legal.html`