Skener síťových služeb.

Počítačové komunikace a sítě

Michal Šedý (xsedym02)

Obsah

Kapitola 1

Úvod

Aplikace pro skenování portů pomáhají zjistit informaci, zda na zkoumaném portu běží služba, a tedy je přístupný. Tyto programy jsou převážně používány správci sítí ke zjišťováni jejich provozu a zabezpečení. V nežádoucích případech jsou využívány útočníky pro zjišťování slabých míst počítačové sítě. Mezi nejznámější softwary skenování sítí patří nmap[1]. Jednoduše naimplementované skenování UDP bylo výrazně rychlejší ve srovnání s již zmíněným nástrojem nmap.

Následující dokumentace popisuje postup návrhu skeneru TCP a UDP portů a jeho funkcionality. Bylo provedeno srovnání s již zmíněným dostupným nástrojem nmap.

[1] Nmap (nmap.org) je nástroj sloužící k vyhledávání počítačových služeb, "mapování".

Kapitola 2

Skenování portů

2.1 TCP skenování

Při TCP skenování je využita metoda SYN skenování. V praxi se zřejmě jedná o nejčastěji využívanou metodu. Na port je zaslán SYN dotaz, a pokud přijde zpráva RST, je port uzavřen, pokud příjde SYN-ACK, je port otevřený. Metoda neprovádí kompletní 3-way-handshake. Výhoda tohoto přístupu spočívá v minimální detekci firewallem, protože není navázáno plnohodnotné TCP spojení.[3] Dále se jedná o vysoce rychlou metodu, která umožňuje přesné rozdělení portů na otevřené, zavřené a filtrované.

Vyplněný SYN paket se odešle na skenovaný port. Jakákoliv příchozí odpověď pochází z dotazovaného portu, tudíž na základě identifikace zdroje zprávy je možné určit stav portu a to i v případě, že je odesláno více dotazů na různé porty současně. V případě, že z portu nepřijde odpověď, je poslán nový dotaz. Pokud nepřijde odpověď ani poté, je port označen jako filtrovaný (není možné určit, zda je otevřen, nebo uzavřen).

2.2 UDP skenování

UDP skenování využívá v programu techniku ICMP nedostupného skenování. UDP skenování je oproti TCP v mnoha ohledech složitější a nepřesnější. Stav otevřenosti portu je zjištěn pouze na základě přijatých zpráv ICMP typu 3 (port unreachable), tedy jsou známy pouze uzavřené porty. U ostatních portů není možné s jistotu určit jejich stav. Pro vyřazení falešných otevření vyžaduje bezpečná implementace násobné zaslání dotazů na zdánlivě otevřené porty. UDP skenování je komplikované počítači, které podle RFC 1812 mohou vygenerovat pouze 80 ICMP zpráv za čtyři sekundy.[2]

ICMP zpráva sice neobsahuje informaci o portu, ze kterého byla odeslána, ale v těle obsahuje hlavičku a posledních osm bytů původního UDP packetu, který zprávu vyvolal. Z původní hlavičky lze získat cílové číslo portu, který je poté označen jako zavřený. Tímto postupem lze zachytávat větší množství zpráv a postupně rozhodovat o číslech zavřených portů.

Kapitola 3

Implementace

Program byl napsán v jazyce C standardu gnu11. Pro zachytávání síťové komunikace byla využita knihovna lpcap. Zrychlení skenování umožnila knihovna lphtread pro použití vláken a s tím i nutnou ochranu zdrojů pomocí mutexu. Skener se dělí na dvě hlavní části: TCP a UDP skenování. Zbytek tvoří podpůrné moduly. Podpora IPv6 byla testována pouze na localhost.

3.1 Argumenty spuštění

Příklad spuštění:
`./ipk-scan {-i <interface>} -pu <range> -pt <range> {<option>}`
`[<domain-name> | <IP-address>]`

Běh programu se dá ovlivňovat jeho parametry spuštění. Parametr `--pt <range>` a jeho zkrácená varianta `-t` specifikují v argumentu skenované TCP porty. `--pu <range>` a zkrácená varianta `-u` definuje v argumentu skenované UDP porty. Porty jsou zadávány číselně. Jednotlivá čísla jsou oddělena čárkami bez mezer. Místo pouhých čísel mohou být použity intervaly, např: 1,2,9 nebo 1,97-100,254. Alespoň jeden z parametrů určujících porty je povinný. Volitelný parametr `-i <interface>` specifikuje identifikátor interface, ze kterého budou posílány pakety. Pokud není zadáno, bude vybráno první neloopbackové interface. Jako poslední parametr při spouštění se udává ip adresa, nebo jméno skenovaného stroje či domény. Výpis výsledků skenování je možné ovlivnit přepínačem `--only-open`, které vypisují pouze otevřené porty, přepínačem `--only-unclosed`, který vypisuje otevřené a filtrované porty, nebo přepínačem `--only-closed`, který vypisující pouze zavřeně porty. Přepínače filtrující výsledky výstupů nelze kombinovat.

Pro zpracování vstupních parametrů programu byl vytvořen modul `arg_parser.c`, který detekuje případné neznámé, nebo chybné kombinace parametrů. Pokud je detekována chyba, je vypsána na standardní chybový výstup hlášení a program je ukončen s návratovým kódem 1.

3.2 Získání interface

Získání a zpracování interface obstarává modul `interface_ip.c`. Pokud nebyl při spuštění programu zadán explicitně identifikátor interface, vybere se první neloopbackové, jehož ip

rodina odpovídá internetovému protokolu skenovaného cíle. Pokud je ale cílem `localhost`, bude zvoleno loopbackové interface. Na základě identifikátoru interface je získaná jeho ip adresa.

3.3 TCP skener

TCP skener je implementován modulem `tcp_scanner.c`. Pomocí funkce `socket` je vytvořen pro IPv4 socket `SOCK_RAW` s protokolem `IPPROTO_RAW`, kterému bude manuálně vyplněna IP hlavička[1]. Pro IPv6 je již využit `IPPROTO_TCP` a IPv6 hlavička je vyplněna automaticky samotným systémem. Socket je nabindován na interface. Vytvoří se dva sniffery (`sniffer.c`) pro zachytávání SYN-ACK a RST odpovědí ze skenovaných portů a jsou následně spuštěny ve dvou vláknech. Pro informace o stavech skenovaných portů je zřízena globální proměnná, v níž mění sniffery hodnoty stavů jednotlivých portů v závislosti na zdrojovém portu zpětně přijaté zprávy. Jedná se o kritickou sekci přístupu ke sdílené proměnné, a proto je využit k synchronizaci mutex. V hlavním vlákně jsou najednou odeslány SYN pakety na veškeré testované porty. Příchozí odpovědi jsou zachytávány ve vedlejších vláknech. Hlavní vlákno zkontroluje, zda přišly odpovědi ze všech portů. Pokud existují takové porty, pro které nelze jednoznačně říci, zda jsou otevřené, nebo zavřené, provede program ještě jednou odeslání SYN paketů na všechny takové porty. Následně jsou ukončena vlákna snifferů a TCP skenner vypíše výsledky skenování. Ty porty, ze kterých nepřišla žádná odpověď, jsou ozačeny jako filtrované.

3.4 UDP skener

UDP skener je implementován modulem `udp_scanner.c` Pomocí funkce `socket` je vytvořen pro IPv4 socket `SOCK_RAW` s protokolem `IPPROTO_UDP`. Pro IPv6 je již využit socket `SOCK_DGRAM`. Vytvoří se sniffer detekující ICMP zprávy typu 3 (port unreachable). Spustí se vlákno snifferu a odešlou UDP pakety na všechny skenované porty. Informace o stavech skenovaných portů jsou uchovány jako v předchozím případě v globální proměnné. Pokud sniffer detekuje příchozí ICMP zprávu typu 3, z původní UDP hlavičky nacházející se v těle ICMP zprávy zjistí port, na kterém byla zpráva vyvolána, a nastaví daný port v globální proměnné jako uzavřený. Po oskenování všech portů je vlákno snifferu ukončeno a hlavní program vypíše výsledky. Všechny porty, které nebyly označeny jako uzavřené, jsou chápany jako otevřené.

Nevýhodou tohoto jinak velice rychlého přístupu je možnost vysokého počtu falešně otevřených portů. Pro přesnější skenování by se měl port, ze kterého nepřišla odpověď, otestovat ještě několikrát, protože předchozí dotaz se mohl pouze ztratit a port je uzavřen. Dalším, a již vážným problémem, je možnost zahlcení starších systémů, které podle RFC 1812 vygenerují pouze 80 ICMP 3 zpráv za 4 sekundy, a tím dojde k falešnému označení velkého množství portů jako otevřených.

Kapitola 4

Struktura programu

4.1 Použité knihovny

`lphtread` - knihovna pro zpravování vláken umožňujících rychlý běh programu.
`lpcap` - knihovna zajišťující filtrování a zachytávání příchozích paketů.

4.2 Moduly

`arg_parser.c` - zpracovávání parametrů spuštění
`interface_ip.c` - načítání a kontrola interface
`port_queue.c` - pomocný modul s implementací FIFO
`sniffer.c` - vytváření snifferů s danými filtry (ACK-SYN/RST/ICMP3)
`tcp_scanner.c` - TCP skenování portů
`udp_scanner.c` - UDP skenování portů
`ipk-scan` - hlavní program řídící skenování

4.3 Návratové kódy

0 - úspěšné provedení
1 - chyba argumentů spuštění
2 - chyba vyhledávání interface
3 - chyba skenované ip/jména (nenalezeno, špatný formát, ...)
4 - chyba během vytváření sniffru
5 - chyba vytvoření vlákna programu
6 - chyba socketu
99 - vnitřní chyba (problém s alokací paměti, ...)

Kapitola 5

Spuštění a testování

K porovnání přesnosti výsledků implementovaného skeneru a jeho rychlosti s dostupnými softwary byl zvolem nástroj `nmap`.

5.1 TCP skenování

Test TCP skenování byl proveden skenováním portů 1-2000 na doméně scanme.nmap.org

```
./ipk-scan -t 1-2000 --only-open scanme.nmap.org
PORT   STATE
22/tcp open
80/tcp open

nmap -sT -p 1-2000 scanme.nmap.org
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

Výsledky obou programů jsou rovnocenné a získané ve stejném čase.

5.2 UDP skenování

Test UDP skenování byl proveden skenováním portů 1-10000 na localhost

```
./ipk-scan -u 1-5000 --only-open localhost
PORT   STATE
68/udp open
631/udp open

nmap -sT -p 1-5000 localhost
PORT   STATE SERVICE
68/udp open  ssh
631/udp open  http
```

Skenování nástrojem `nmap` trvalo šest a půl minuty. Program `ipk-scan` zvládl úspěšně otestovat stejný počet portů za dvě sekundy.

Kapitola 6

Závěr

Implementace TCP skeneru se svou rychlostí a výsledky vyrovná dostupným programům pro skenování sítí. Ovšem UDP skener předčil svou rychlostí nástroj nmap. Implementace ovšem nepočítá s omezením některých systému podle RFC 1812, které generují pouze 80 ICMP zpráv za 4 sekundy. Skenování velkého množství portů na takto omezeném zařízení, by vedlo k falešnému označení mnoha portů jako otevřených.

Literatura

[1] *Advanced TCP/IP - The TCP/IP Stack & OSI Layer*. Dostupné z: https://www.tenouk.com/Module42.html.

[2] *The Art of Port Scanning*. Dostupné z: https://nmap.org/nmap_doc.html#port_unreach.

[3] *TCP SYN (Stealth) Scan (-sS): Nmap Network Scanning*. Dostupné z: https://nmap.org/book/synscan.html.