DOKUMENTACE 2. PROJEKTU IPK

Scanner síťových služeb

Resumé

Jednoduchý síťový TCP/UDP scanner

Radek Duchoň, xducho07

Xducho07@stud.fit.vutbr.cz

Obsah

Popis problému a návrh řešení

Ve své podstatě je zapotřebí pouze zjistit svou a cílovou IP adresu (pokud je zadáno doménové jméno a ne samotná cílová adresa), vyplnit hlavičky packetů, odeslat packety na skenované porty (v rozmezí 0 - 65535) pomocí příslušného protokolu TCP nebo UDP a zpracovat případnou odpověď serveru pro rozhodnutí, zda je daný port na serveru otevřený, uzavření, či filtrovaný.

V případě TCP se odesílá packet s příznakem SYN, a pokud není daný port filtrovaný, vrací se data s nastavenými příznaky SYN a ACK, kde dle obsahu lze zjistit, zda je port otevřený či uzavřený.

V případě UDP to funguje ve své podstatě naopak, pokud není přijatá odpověď, port je otevřený. V případě UDP odpovědi je port otevřený, avšak taková odpověď není příliš obvyklá. Pokud se vrátí ICMP odpověď, obsahuje data buď o chybě při nedostupnosti (typ 3, kód 3) a je tedy uzavřený, anebo obsahuje jinou chybu a port je tedy filtrovaný.

Implementace a testování

Na počátku jsou v jednoduchém cyklu a sérii porovnání řetězců zpracovány argumenty příkazové řádky a do dvou vektorů (zvlášť pro TCP a UDP) jsou vloženy zadané porty pomocí funkce `ports`, která se posunuje postupně mezi čárkami (případně pomlčkou).

Mnou navržené řešení zaručuje, že v případě duplicitního zadání `-pu` nebo `-pt` budou stále zpracovány všechny porty, mnou navržená funkce taktéž dovoluje kombinaci zadávání x,y a x-y, například tedy 1,2,5-10,13-18. V případě, že člověk by člověk umazával část čísel a na konci mu zůstala nedopatřením čárka, program ji jednoduše vypustí.

Po zpracování portů jsou porty seřazeny, odstraněny duplicity a je zkontrolováno, že jsou všechny v platném rozsahu 0-65535, kde díky seřazení stačí zkontrolovat pouze krajní porty.

Pro zjištění IP adresy domény byla využita funkce `getaddrinfo` (inspirace zde), která má tu výhodu, že je jí jedno, zda člověk zadá doménové jméno či IP adresu, takže není potřeba dopředu složitě kontrolovat, zda se již náhodou nejedná o IP adresu. (Taková kontrola by nebyla obtížná pro IPv4, ale kontrolovat IPv6 je již poněkud horší.)

Následně je získáno pomocí pomocné funkce `getInterface` (inspirace příkladem zde) první aktivní běžící neloopbackové rozhraní, kde typ (IPv4 vs IPv6) odpovídá skenované adrese, nebo v případě zadaného jména rozhraní je získáno aktivní běžící rozhraní daného jména.

Dále jsou vytvořeny sokety a vytvořeny a vyplněny IP, TCP a UDP hlavičky + pomocná hlavička pro výpočet checksum u TCP. Zde byl velmi nápomocen program wireshark, který upozornil, když byly některé údaje (například délka hlavičky nebo checksum) špatně, algoritmy pro checksum byly proto vyzkoušeny 3 z různých zdrojů, pro další vyplnění byly nápomocné zdroje jako wikipedia s podrobným popisem, co hlavičky obsahují a příklady na internetu, kde je lidé nějak vyplňovali.

Nakonec se odesílají pakety na dané porty a očekává se odpověď, tato část však není i přes veškeré snahy nakonec funkční. Při kontrole pomocí wiresharku se podařilo zjistit, že ač je vybráno správné rozhraní, data se z nějakého důvodu odesílají na loopback. V případě pokusů, kdy se odstranila například podmínka na správnou rodinu při získávání rozhraní se podařilo sice data odeslat správně, nicméně jelikož byla následně údajná adresa rozhraní 2.0.0.0, nepodařilo se získat odpověď, objevili se pouze dotazy, kdo má danou adresu. Pokud by někdo, kdo toto bude číst, věděl, v čem jsem udělal chybu, prosím o sdělení na email na úvodní stránce

Zdroje:

https://en.wikipedia.org/wiki/Port_scanner

https://en.wikipedia.org/wiki/IPv4#Header

https://stackoverflow.com/questions/14235208/getifaddrs-to-parse-only-the-ip-from-etherenet-interface-eth-or-wlan-interface

https://cs.wikipedia.org/wiki/Raw_socket

https://www.tenouk.com/Module43a.html

https://en.wikipedia.org/wiki/IPv6_packet

http://man7.org/linux/man-pages/man3/

http://www.linuxhowtos.org/manpages/3/getifaddrs.htm

https://www.tcpdump.org/manpages/pcap_compile.3pcap.html

https://www.devdungeon.com/content/using-libpcap-c

https://stackoverflow.com/questions/4516436/pcap-set-rfmon-does-not-work

https://linux.die.net/man/2/setsockopt

https://www.cnblogs.com/rollenholt/articles/2590959.html

https://stackoverflow.com/questions/4583386/listening-using-pcap-with-timeout

https://www.geeksforgeeks.org/tcp-services-and-segment-structure/

https://www.tcpdump.org/pcap.html

https://www.thegeekstuff.com/2012/10/packet-sniffing-using-libpcap/

https://cs.wikipedia.org/wiki/SYN_flood

https://linux.die.net/man/7/pcap-filter

https://nmap.org/book/scan-methods-udp-scan.html