VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

POČÍTAČOVÉ KOMUNIKACE A SÍTĚ

2018/2019

2. Projekt

Scanner Síťových služeb

Jakub Sadílek (xsadil07) Brno, 21. dubna 2019

Obsah

Úvod

Zadání projektu bylo vytvořit jednoduchý TCP/UDP skener, který oskenuje zadané porty na zadaném zařízení a na standardní výstup vypíše stav těchto portů (tj. otevřený, filtrovaný, uzavřený). Pakety musí být odeslané pomocí BSD soketů a program by měl být co nejvíce multiplatformní na operačních systémech založených na unixu.

Implementace

Program je implementován v programovacím jazyce C++ ve standardu C++11. K odesílání paketů používá RAW sokety a pro jejich zachytávání externí knihovnu libpcap, kterou vyžívá například i známý program Wireshark. Program pracuje na portu 49224 a podporuje pouze IP protokol verze IPv4.

Zpracování argumentů

Jako první krok bylo potřeba zpracovat vstupní parametry programu. Jelikož program podporuje poměrně velké množství parametrů bylo by výhodné použít dostupné funkce pro jejich zpracování, ovšem já se rozhodl jít cestou podmínek, cyklů a regulárních výrazů.

V cyklu procházíme argumenty a pokud narazíme na známý přepínač, tak se rozhodneme o další akci, tj. načtení portů, interfacu či adresy vzdáleného stroje, pokud nepřecházel žádný přepínač. Ta může být zadána přímo adresou nebo jménem domény. Pro převod byla použita funkce `getaddrinfo()`, která je součástí knihovny netdb.h. Porty mohou být zadané výčtem nebo rozsahem. O který zápis se jedná je rozhodnuto regulárním výrazem a následně se volá funkce, která tyto argumenty vyčte a uloží do příslušného globálního vektoru. Interface je nepovinný argument, podle kterého volíme naší IP adresu. Pokud nebyl interface zadán, vezmeme první neloopbackovou adresu a informace uložíme do globální struktury reprezentující koncové zařízení komunikace.

TCP skenování

Pokud byl zadán aspoň jeden TCP port zavolá se funkce `TCPscan()`, která provede celé skenování. Jako první se inicializuje paket, do kterého na začátek vložíme IP hlavičku a naplníme ji informacemi (adresu příjemce a odesílatele, protokol atd..) a vypočítáme kontrolní součet. Za IP hlavičku vložíme TCP hlavičku a naplníme ji informacemi (například zdrojový a cílový port atd..) a kontrolní součet se v tomto případě prozatím nastaví na 0. Pro výpočet TCP kontrolního součtu musíme vytvořit pomocný paket, do kterého vložíme tzv. „pseudo TCP hlavičku“ a za ni nakopírujeme tu originální. Pak vypočítáme kontrolní součet nad celým pomocným paketem a výsledek uložíme do originální TCP hlavičky.

Poté co máme nastavený paket inicializujeme a nastavíme RAW soket, přes který paket odešleme. Otevřeme zachytávání paketů přes funkci `pcap_open_live()`, která je součástí knihovny pcap/pcap.h. Dále nastavíme filtr, který bude filtrovat všechny příchozí pakety, pouze na ten, který nás zajímá. Nastavíme tedy filtrování na TCP protokoly, IP odesílatele, IP příjemce a cílový port. Filtr zkompilujeme a nastavíme přes funkce

`pcap_compile()` a `pcap_setfilter()`. Pak už jen stačí paket odeslat přes funkci `sendto()` a zachytit odepověď přes `pcap_dispatch()`. Ovšem odpovědi se nemusíme dočkat, pokud server má port filtrovaný nebo se odpověď ztratí, proto čas na přijmutí paketu musíme omezit. To lze přes systémový signál `SIGALRM` a nastavíme čas na 2 sekundy. Po uplynutí tohoto intervalu voláme funkci `pcap_breakloop()`, která ukončí čekání.

Výsledek TCP skenu závisí na odpovědi serveru, ten se může zachovat třemi způsoby:

1) Odešle zpátky paket s flagy SYN a ACK, které znamenají, že server chce pokračovat v komunikaci, a proto lze port označit za otevřený.
2) Odešle zpátky paket s flagy ACK a RST, které znamenají, že server nechce pokračovat v komunikaci, a proto port lze označit za uzavřený.
3) Server neodpoví, a to může znamenat, že náš paket zahodil a je port je filtrovaný nebo se paket po cestě ztratil, a proto odešleme paket pro jistotu ještě jednou, než ho označíme za filtrovaný.

IP hlavička je naplněna pouze jednou. TCP hlavička je naplněna standartními údaji, ale některé části je potřeba měnit pro každý skenovaný port. Konkrétně to je cílový port, sekvenční číslo a kontrolní součet. Kontrolní součet je náhodně zvolené číslo, které se s každým paketem inkrementuje.

UDP skenování

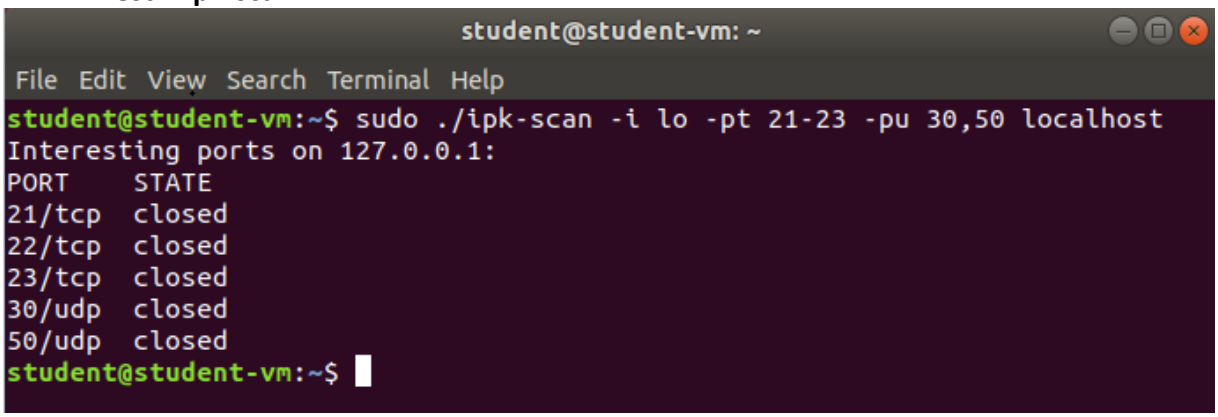
UDP skenování provádí funkce `UDPscan()` a princip je velmi podobný TCP skenování, proto zmíníme jen zajímavé odlišnosti. Jako první je naplnění UDP hlavičky, která obsahuje podstatně méně informací než TCP a nemusíme v ní počítat kontrolní součet (alespoň v případě IPv4). Dále UDP protokol nenavazuje žádné spojení, proto nelze ověřovat dostupnost portů podobným způsobem jako u TCP, ale lze využít protokolu ICMP. Tento protokol je určen pro signalizaci dostupnosti. Při zaslání našeho UDP paketu na nedostupný port, operační systém stanice generuje ICMP paket typu 3 kódem 3 (port unreachable). Proto pokud tuto zprávu neobdržíme, lze považovat port za otevřený, jinak za uzavřený. Bohužel zde není žádný způsob, jak zjistit, že port je filtrovaný. Další problém je, že ICMP zpráva je zapouzdřena přímo v jediném datagramu, a tak jako UDP nezaručuje doručení paketů, proto jak v případě TCP odešleme dotaz ještě jednou.

Výpis

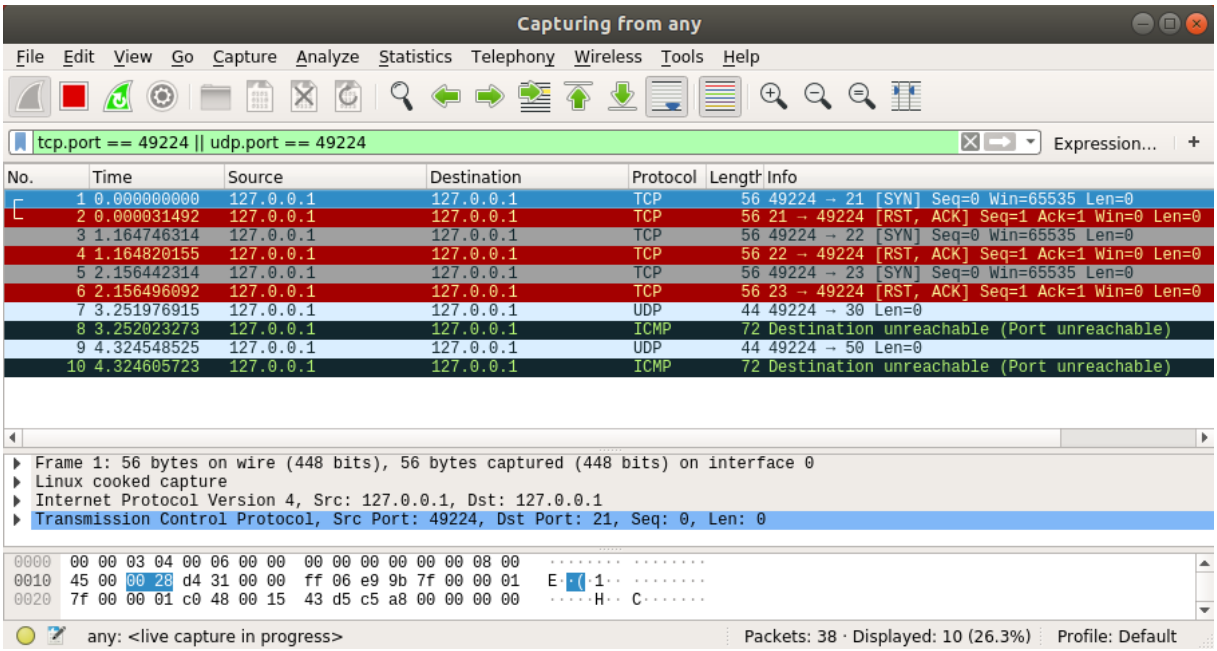
Během celého programu je potřeba zaznamenávat důležité informace od IP adresy, přes seznam portů až po jejich dostupnost. To je realizováno globálním stringem, do kterého postupně tyto informace ukládáme a na konci ho vypíšeme. Další výhodu má, že v případě chyby nejsou na standardním výstupu napůl vypsané informace. Pokud nastane chyba je na standardní chybový výstup vypsána chybová hláška a program je ukončen návratovou hodnotou 1.

Testování

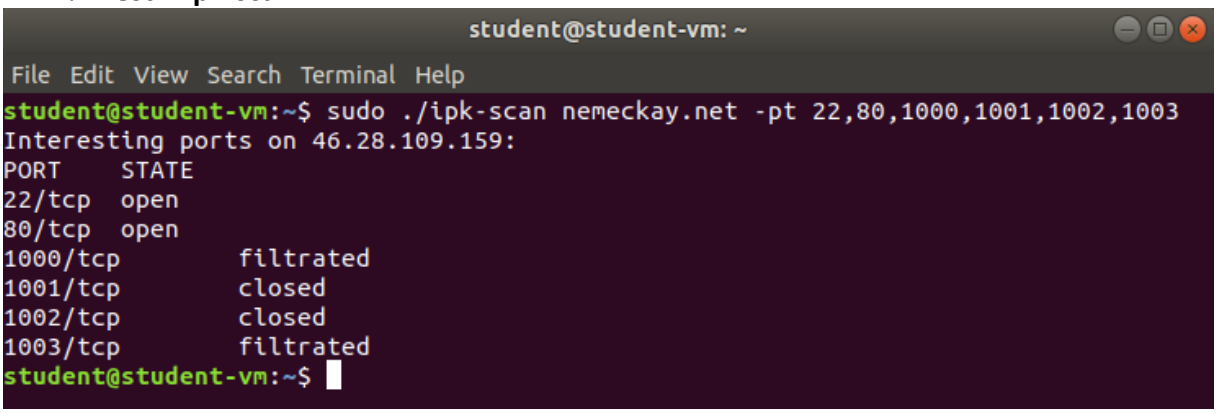
1. **Test – ipk-scan**



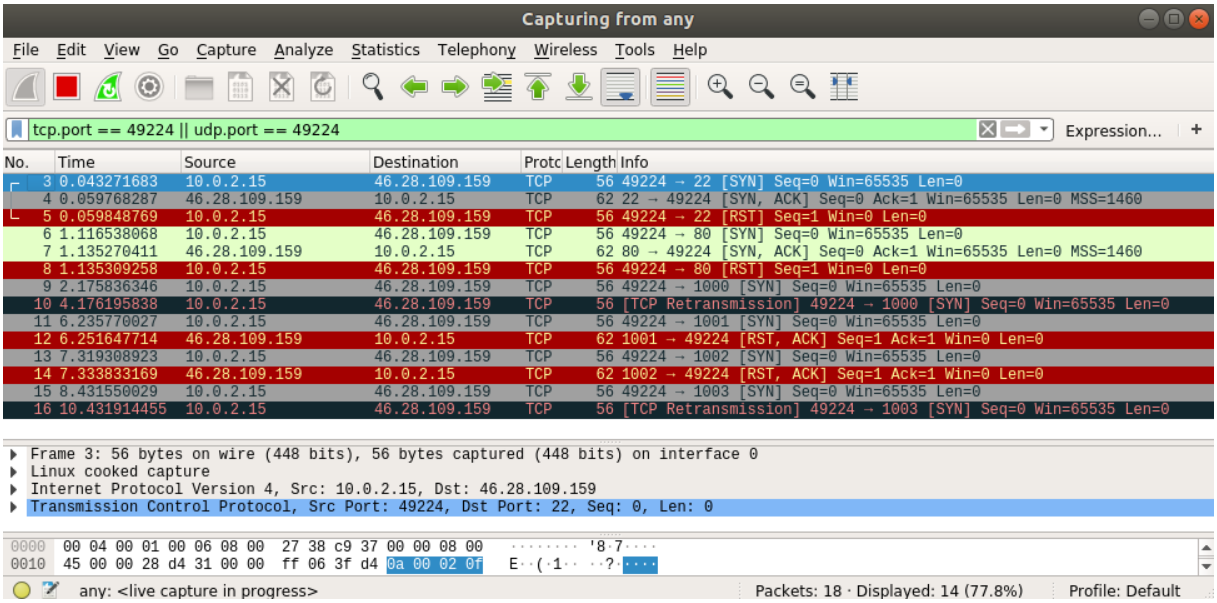
1. **Test – Wireshark**



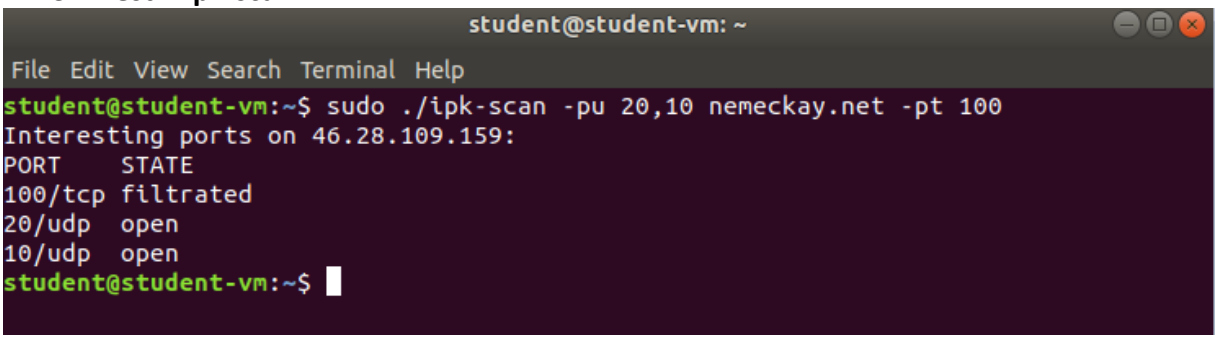
2. **Test – ipk-scan**



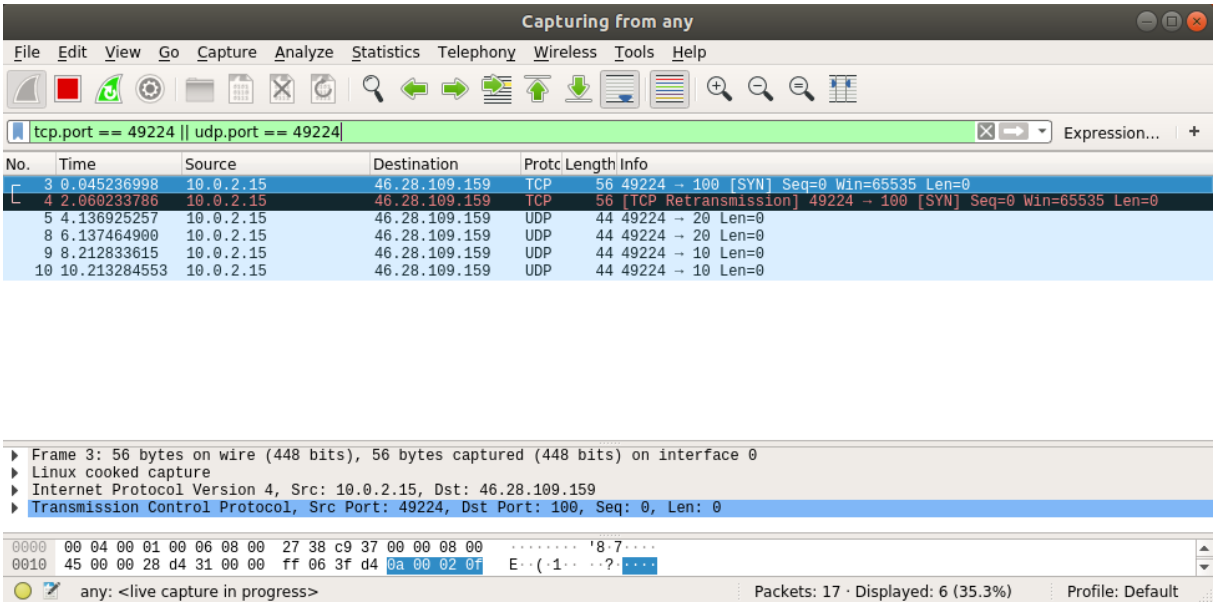
2. Test – Wireshark



3. Test – ipk-scan



3. Test – Wireshark



Závěr

Zpočátku mě projekt velmi bavil a libí se mi jeho praktická použitelnost. Díky němu se moje teoretické i praktické znalosti sítí nemálo obohatily. Jeho náročnost byla vyšší, než jsem původně očekával. Jelikož jsem měl problémy při testování IPv6, tak jsem se rozhodl ji raději vynechat než odevzdat řádně neotestovaný program.

Použitá literatura a zdroje

- Příklady RAW soketů - https://www.tenouk.com/Module43a.html
- ICMP - https://cs.wikipedia.org/wiki/ICMP
- Převod domény na IP adresu - https://stackoverflow.com/questions/5760302/when-i-do-getaddrinfo-for-localhost-i-dont-receive-127-0-0-1
- Kontrolní součet - https://stackoverflow.com/questions/8845178/c-programming-tcp-checksum
- Zpracovávání odpovědí - https://www.tcpdump.org/pcap.html
- SIGALRM - https://stackoverflow.com/questions/4583386/listening-using-pcap-with-timeout