VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové komunikace a sítě
2019/2020

# OMEGA: Scanner síťových služeb

1. května 2020 xmusko00

# 1 Zadání

Vytvořte jednoduchý síťový TCP, UDP skener v C/C++/C#. Program oskenuje zadanou IP adresu a porty. Na standardní výstup vypíše, v jakém stavu se porty nacházejí (otevřený, filtrovaný, uzavřený).

U UDP skenování můžete uvažovat, že daný počítač při zavřeném portu odpoví ICMP zprávou typu 3, kódu 3 (port unreachable). Ostatní porty považujte za otevřené.

V případě TCP posílá scan pouze SYN pakety. Neprovádí tedy kompletní 3-way-handshake. Pokud přijde odpověď RST - port je označen jako uzavřený. Pokud po daný časový interval nepřijde ze skenovaného portu odpověď, je nutno ověřit dalším paketem a teprve potom port označit jako filtrovaný. Pokud na daném portu běží nějáká služba, je port označen jako otevřený.

# 2 Úvod do problematiky

Při síťové komunikaci se využívá k adresaci IP adresa a port. Každý port má unikátní číslo a často je přižazen ke specifické službě - tzv. well-known ports. Skecnováním portů je obecně myšleno zjišťování stavu daného portu.[1] Toho se dá docílit pomocí dvou hlavních způsobů, skenu UDP, nebo TCP.

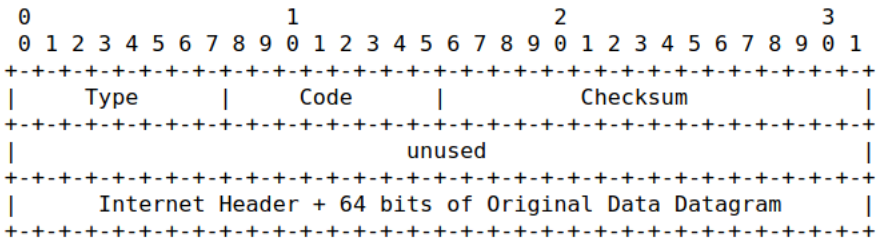
## 2.1 UDP scan

Dle zadání máme stav vyhodnotit na základě ICMP zpravy, což je podpůrný protokol složící k zasílání chybových zpráv při komunikaci s jinou IP adresou. Jedna z těchto chyb je právě uzavřený port.

Pokud je ale daný port filtrovaný (firewallem, OS, ..), neodesílá se zpět žádná zpráva a port je považován na straně scanneru za otevřený.

### ICMP

Při ICMP verze 4 nás zajímá typ 3 a kód 3 (`port unreachable`), který je zaslán v případě uzavřeného portu.

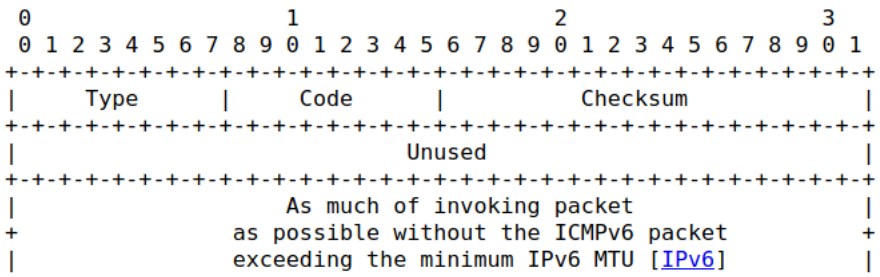



Obrázek 1: ICMP (3, 3)

ICMP s tímto kódem v sobě navíc obsahuje 64 bitů původního datagramu.[4]

**ICMPv6**

U ICMPv6 značí nedostupný port typ 1 (`Destination unreachable`) a kód 4 (`port unreachable`) [3].

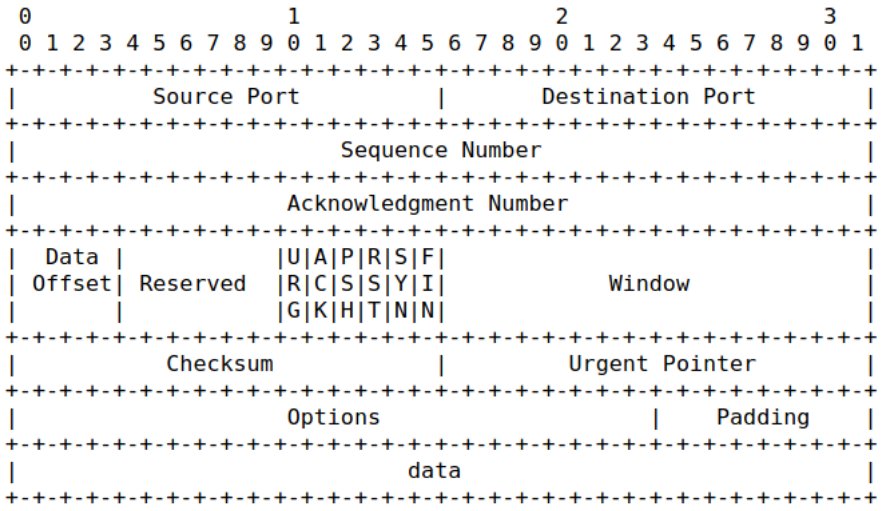



Obrázek 2: ICMPv6

## 2.2 TCP scan

U TCP protokolu se komunikace navazuje tzv. 3-way-handshakem. Každý typ tohoto hanshaku má nastavené flagy. Při navazování komunikace se odesílá SYN paket. Druhá strana odpovídá SYN-ACK paketem, pokud je port otevřený. Pokud je uzavřený, odpovídá RST paketem. V případě filtrování nepřichází odpověď žádná [2].

TCP v sobě nese zdrojový a cílový port, který nás bude zajímat [5].




Obrázek 3: TCP

# 3 Implementace

K implementaci jsem zvolila jazyk `C++`.

Dále jsem využila knihovny `pcaplib`, `sys/socket`, `netinet/in`, `netinet/udp`, `arpa/inet`, `net/ethernet` a `thread`.

## 3.1 UDP scan

Jak u IPv4, tak IPv6 jsou odesílány pakety na daný port. Pokud přijde daná ICMP zpráva, je označen za uzavřený, jinak za otevřený.

Pakety jsou zasílány standartním způsobem pomocí `sendto`.

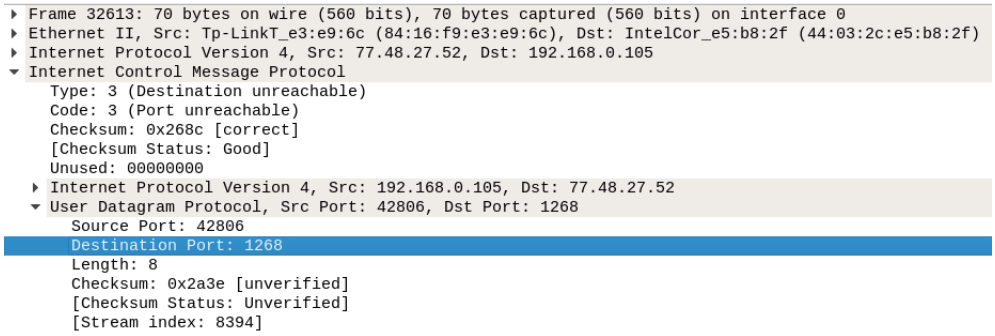
Jejich odchytávání je realizováno s pomocí knihovny `pcaplib`. Odchytává se na stejném rozhraní, ze kterého byly pakety poslány, navíc je vždy nastaven filtr pouze na dané ICMP zprávy. Zpracování paketů má na starosti `pcap_dispatch` v neblokujícím módu.

### IPv4

V ICMP samotném není obsažena informace o portu. Tento protokol ale v sobě naštěstí obsahuje kopii dat původního datagramu i s číslem cílového portu. To umožňuje poslat všechny pakety naráz a při odchytávání rozlišovat skenovaný port.

Výrazně se tím snižuje doba potřebná k oskenování. Rozdíl mezi časem potřebným k oskenování jedno portu a všech portů je prakticky zanedbatelný.

Abychom se k číslu portu dostali je potřeba znát, kde přesně se v paketu nachází. Prvních 14 bytů zabírá Ethernet. Pak následuje IP hlavička, ve které se dozvíme celkovou délku IP[6] . Za ní je samotné ICMP s tělem obsahující původní IP protokol, a nakonec datagram se zdrojovým a cílovým portem.




Obrázek 4: Struktura paketu zachyceného WireSharkem

### IPv6

U ICMPv6 už ale nelze port nijak vyčíst. Pakety se skenují po jednom, což znamená odeslání paketu, zahájení odchytávání a počkání danou dobu na přijetí.

Skenování je proto mnohem pomalejší a závisí na počtu skenovaných portů.

## 3.2 TCP scan

Pro nekompletní hanshake je třeba odesílat RAW pakety a dostat se tedy k TCP vrstvě. Toho lze docílit nastavením soketu na typ `SOCK_RAW` a protokol `IPPROTO_TCP`. U IPv4 se nám tím zpřístupní i IP vrstva.

SYN pakety jsou odesílány v jedné vlně.

K odchytavání paketů jsem použila opět knihovnu `pcaplib`. Tentokrát je ale použito dvou "odchytávačů"se dvěma různými filtry. Jeden pro SYN-ACK pakety (otevřený port), druhý pro RST (uzavřený port).

Jak už bylo řečeno, TCP hlavička v sobě obsahuje číslo portu. Stačí se tedy u příchozí zprávy podívat dovnitř paketu, podobně jako u UDP skenování. Opět to velmi urychluje skenování a není prakticky žádný rozdíl v časové náročnosti scanu jednoho nebo tisíce portů.

Pokud k některým portům nepřišla odpověď, je proces zopakován. Bez jakékoliv odpovědi je port označen za filtrovaný.

Implementována byla jen verze s IPv4.

# 4 Testování

Testování probíhalo na domácí síti a domácím serveru.
Ke sledování provozu jsem použila WireShark.

## Zdroje


[1] *Port* [online]. Dostupné na: https://en.wikipedia.org/wiki/Port_(computer_networking).

[2] *Port scanner* [online]. Dostupné na: https://en.wikipedia.org/wiki/Port_scanner.

[3] *RFC 4443* [online]. Dostupné na: https://tools.ietf.org/html/rfc4443#page-8.

[4] *RFC 792* [online]. Dostupné na: https://tools.ietf.org/html/rfc792.

[5] *RFC 793* [online]. Dostupné na: https://tools.ietf.org/html/rfc793.

[6] *Using libpcap in C* [online]. Dostupné na: https://www.devdungeon.com/content/using-libpcap-c.