Introduktion till Linux och små nätverk

Grunder i nätverk

När datornätverk kommunicerar, så delar man upp kommunikationen i lager, där varje lager hos sändaren kommunicerar med motsvarande lager hos mottagaren och varje lager gör en sak.

OSI-modellen

En sådan modell som används när man diskuterar datakommunikation är en modell så heter den *OSI-modellen* (Open System Interconnection Model), och som är standardiserad av ISO (Internationella Standardiseringsorganisationen), se illustration 1.

Det är en standard för att dela upp kommunikationen från en datoranvändare/applikation till en annan och definierar vilket ansvar varje lager har. Det anger även vilken information som lagret lägger till hos sändaren och vad som då tas bort hos mottagaren.

Man kan se det som att data överförs mellan sändaren och mottagaren på samma lager, men att de använder de underliggande lagren för att göra själva överföringen av data till mottagaren. Varje lager har då en entydig uppgift.

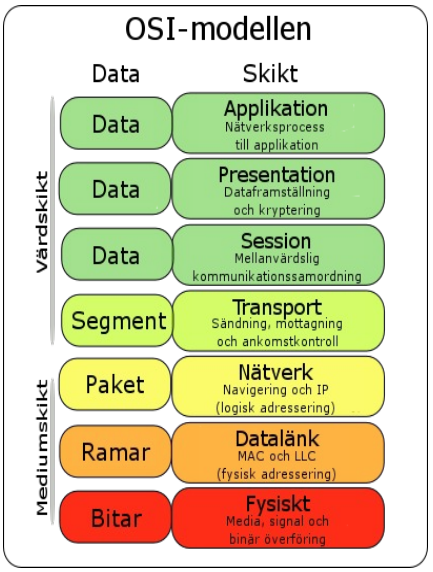

Illustration 1: OSI-modellen

Förklaring av lagren

En överföring av data mellan sändare och mottagare kan då översiktligt ske på detta sätt.

Applikationslagret genererar data som skall skickas över till mottagarens applikationslager. Detta lager anger vilken logisk namn som mottagaren har. För att överföra data, så skickas data vidare till det underliggande *presentationslagret*.

Presentationslagret konverter data det får till ett standardiserat format för att skicka över till mottagarens presentationslager. Detta gör att implementationerna kan vara olika, men om de skickar data enligt en standard, så kan de fortfarande kommunicera med varandra (ex webbläsare och webserver).
För att överföra detta till mottagarens presentationslager, så skickas det nu det konverterade data vidare ned till sändarens *sessionslager* för att transport till mottagaren.

Sessionslagret hantera en session för applikationen. En session är dialogen mellan olika datorers sessionslager, från första till sista överföringen av data. Här hanteras även eventuellt användare och lösenord eller annan liknande nödvändig information för att kunna prata med mottagarens sessionslager. Detta data skickas sedan till *transportlagret* för vidaresändning till mottagaren.

Transportlagret ser till att skicka data till rätt dator och mottagarprogram. Vilket applikation som mottar data avgörs med protokolltyp, portnummer och maskinens logiska numeriska adress. Transportlagret översätter även maskinens logiska namn från applikationslagret översättas till maskinens logiska numeriska adress. Transportlagret skickar sedan detta data vidare till *nätverkslagret* för överföring till mottagaren.

Nätverkslagret skickar data till rätt maskin utifrån mottagarens logiska numeriska adress. Om datamängden är för stort för underliggande datalänklager, så kan sändaren dela upp data i flera paket. Dessa paket sätt sedan samman hos mottagaren innan de skickas upp till dess transportlager. Här avgörs även om sändar- och mottagarmaskin sitter på samma LAN. Om så är fallet, så skickas data direkt till mottagaren. Annars så skickas det till en s.k. router som ser till att skicka data vidare till rätt LAN och rätt maskin. För att göra detta så lägger lagret till den fysiska adressen på den maskin som paketet faktiskt skall skickas till. Antingen mottagarmaskinens om den befinner sig i samma LAN, annars är det routerns fysiska adress. Noteras att den logiska numeriska adressen är kvar och opåverkad i datat som skickas vidare. Dessa paket skickas sedan av nätverkslagret till *datalänklagret* för att sändas till mottagaren.

Datalänklagret sätter ihop ramar som skall skickas över till mottagaren med den fysiska adressen från nätverkslagret. Ram är ett paket som har fysisk sändar- och mottagaradress och checksumma, förutom datat från nätverkslagret. Så datalänklagret skapar kontrollsumma till ramen och skickar vidare ramen till mottagaren på samma LAN. Mottagaren kontrollerar maskinadressen och checksumman och skickar upp det till sitt nätverkslager om dessa är korrekt och skall till den här maskinen, annars så slängs ramen. För att skicka ramen till mottagarens datalänklager, så används det *fysiska lagret* för den faktiska fysiska dataöverföringen.

Fysiska lagret konverterar ettor och nollor i ramen till en fysisk enhet, vanligen spänning, ström, radiovåg eller ljus, som mottagaren kan läsa av och i sin tur konvertera tillbaka till ettor och nollor. Dessa ettor och nollor skickas upp till mottagarens datalänklager.

Så hos mottagaren görs sedan allt detta i omvänd ordning. Data som kommer från undre lager tas emot. Det data som motsvarande lager hos sändaren lagt till kontrolleras och tas bort. Om det är korrekt, så skickas data upp till lagret ovanför. Om inte så kan data slängas eller så skickas ett felmeddelande till lagret ovanför eller så skickas felmeddelandet till motsvarande lager hos sändaren.

OSI-modellen finns inte längre som en konkret implementation, utan bara som en allmän diskussionsunderlag när man beskriver datakommunikation. Den används när man jämför olika kommunikationsprotokoll så att man snabbt skall förstå det nya protokollet.

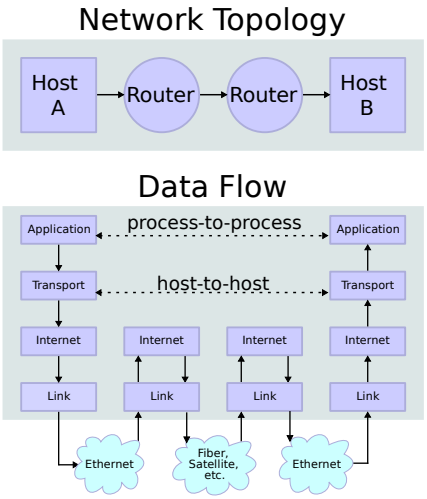

Illustration 2: Routing, förenklat

Routing

När data skickas till datorer som finns på olika LAN, så kallas processen för att skicka data till rätt LAN och dator för routing, och använder sig av s.k. router (routrar) för att skicka datat.

Hos en router så kommer data inte att gå ända upp till applikationslagret, utan så fort routern vet vart den skall skicka datat, så går det ned i lagren igen till ett annat LAN som routern är ansluten till, som man kan se illustration 2. Här använder vi en enklare modell än OSI-modellen, som mer liknar TCP/IP.

Bilden illustrerar när data skickas mellan dator A och dator B med två mellanliggande routrar som binder samman de nät som dator A sitter på med det nät som dator B sitter på.

Dator A:s program vill skicka data till dator B:s motsvarande program. Det illustreras med den

streckade linjen i figuren. Men för att göra det så måste programmen använda sig av transportlagret. Transportlagret på dator A skall då prata med transportlagret på dator B. För att göra det så behöver de använda nätverkslagret.

Nätverkslagret vet att dator B inte finns på samma nätverk, så då ber det datalänklagret att skicka datat till den första routern för vidaresändning till det LAN som B befinner sig i.

Den första routern tar emot data på datalänklagret och skickar upp det till nätverkslagret. Routerns datalänklager ser att data inte skall gå till den utan till dator B. Eftersom den inte har en direkt anslutning till mottagarens LAN, så skickar den första routern att skicka vidare data till den andra routern, som sitter närmare dator B:s LAN.

Den andra routern tar emot data och skickar upp det till nätverkslagret. Här ser nu den andra routern att den sitter ansluten till samma LAN som B, så därför kan den skickar data direkt till datorn B, som då har fått datat som dator A skickade.

Nu har vi överfört data från dator A till dator B, via två routrar, enligt den förenklade OSI-modellen.

Hur vet varje router vilken annan router som sitter närmare en dators LAN?

En router väljer till vilken den skall skicka vidare paket genom att den har kontroll vilka LAN som finns ”bakom” varje nätverksanslutning som den har. Den vet även ”långt bort” och/eller ”hur dyrt” det är att nå dessa LAN via varje anslutning.

Hur routern får reda på detta är något som denna kurs inte berör. Men det finns flera applikationer och protokoll som räknar ut detta med hjälp av data från exempelvis sina närliggande routrar. Detta kalla *routingprotokoll*, och är en hel kurs i sig själv.

Internet Protokollet

IP-stacken[1] har fem lager istället för OSI-modellens sju lager. Hur det ser när man jämför dem med varandra ser man i illustration 3. Det är i huvudsak som illustreras i bilden, men det finns undantag.

Som synes så är *Application-*, *Presentation-* och *Session*-lagren i OSI-modellen ett lager i IP-stacken. *Transportlagret* motsvara TCP och UDP, *nätverkslagret* av IP samt *data-länklagret* och delar det *fysiska lagret* av ARP, ICMP och Ethernet. Som synes av illustrationen så är det inte riktigt 1:1-förhållande mellan OSI- och IP-modellerna, men de stämmer i stort överens.

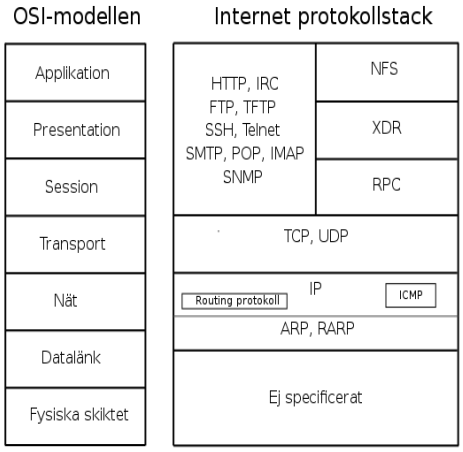

Illustration 3: OSI- och IP-stacken

Vad man kan se här är följande saker:

1. För att kommunicera med ett annat program så måste man veta vilken dator det andra programmet exekverar på. Det anges med IP-adressen (IPv4 eller IPv6).
2. När man vet det så måste man veta om det är TCP eller UDP som man skall använda för att kommunicera med det andra programmet.
3. När detta är klart så kan det finnas många program som kör exempelvis TCP på den datorn. För att veta vilket program man vill kommunicera med, så anger man ett portnummer. För

1 IP-stacken kallas även ibland för TCP/IP-stacken.

TCP och UDP består portnumret av 16 bitar och har alltså ett värde mellan 0 och 65535. De första 4096 (0-4095) portarna är reserverade för kända tjänster, som exempelvis HTTP (webserver) , SMTP (datorpost), DNS (domännamnsuppslagning) etc.

4. Nu kan ett program skapa en kontakt med ett program på den andra maskinen. Därmed kan programmen skicka kommandon enligt ett förutbestämt sätt, s.k. protokoll. Dessa protokoll skiljer sig åt, och är beroende av vilken tjänst man vill använda.

ICMP

ICMP är lite speciell och används för att skicka status mellan maskinerna. Exempelvis om man vill se om en maskin är aktiv på Internet, så kan man använda sig av ICMP Echo, vilket programmet ping(1) använder sig av. Andra användningar av ICMP är att maskinen man som man ansluter sig till kan skicka ett meddelande tillbaka, exempelvis om att det inte är tillåtet att ansluta till den TCP- eller UDP-port som man försökt ansluta till.

Transportskikten TCP och UDP

Alla data som skickas över internet delas upp till paket, som kan vara olika stora. Vanligtvis är de maximala storleken på ett paket 1480 (antal byte som kan skickas som data i ett vanligt Ethernet-ram). För att skicka mer date, så skickas bara fler paket. Problemet, vilket faktiskt är en styrka, är att paketen kan gå olika vägar genom internet. Så paket som skickas i en ordning kan komma fram till mottagaren i en annan ordning. Sedan kan paket som skickas tappas bort eller kopieras och skickas i många kopior. Så hur påverkar detta TCP och UDP?

Hur UDP fungerar

Jo, när det gäller UDP så innehåller paketet, förutom IP-nivåns huvud med avsändar- och mottagaradresser samt pakettyp(UDP), bara avsändar- och mottagar-portnumer, datalängd samt checksumma.

Detta så att mottagaren skall vem som skickade datat så att den kan returnera data, samt portnummer för att veta vilket program som skall ta emot datat. Men även för att det skall veta hur mycket data som skickas med paketet och om data överförts korrekt.

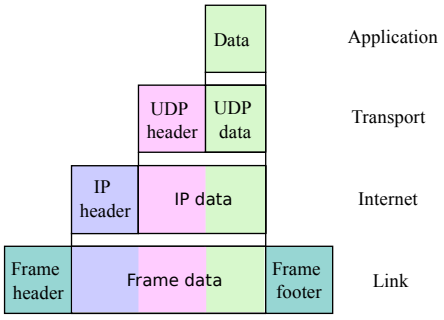

Illustration 4: Hur UDP fungerar med IP

Illustration 4 visar hur data som skickas från en applikation med UDP får mer information allt eftersom det passerar lagren. I UDP läggs ovan nämnda information, mottagar- avsändaradress, längd och checksumma, in i ett UDP-huvud framför datat. Detta skickas till Internet-lagret som lägger på en IP-huvud innan sitt data, som består av UDP-paketet. Detta skickas sedan ned till länklagret som data. Länklagret använder normalt Ethernet för överföring, så därför läggs ett Ethernet-ram-huvud med sändarens och mottagarens hårdvaruadress (MAC-adress) samt en Ethernet-svans för synkronisering runt IP-paketet. Detta skickas sedan över till mottagaren via något fysiskt medium.

Det innebär alltså att maximalt data som kan skickas mellan två datorer med ett paket är vanligen 1480-2-2-2-2 = 1472 bytes. Detta eftersom Ethernet normalt överför max 1480 bytes och portnummer, längd och checksumma är 2 bytes. Vill man skicka mer data, så måste man skicka det som flera paket.

Eftersom datornätet kan ändras mellan sändningarna, som exempelvis att en router startar eller stängs av, så innebär det att man inte vet hur dessa två paket kan komma fram. De kan komma fram i omvänd ordning, ett kan tappas bort helt och det andra kan komma fram i flera kopior.

Detta innebär att ett program som är skrivet för att använda UDP måste ta hand om detta själv.

Fördelen är att ett program direkt kan skicka data och inte behöver vänta på synkronisering med mottagaren för att kunna veta om paketet kommit fram. Detta är användbart exempelvis när man strömmar video eller ljud mellan datorer eller vid datorspel. Då är det viktigare att data kommer fram i tid än att allt kommer fram i rätt ordning.

Hur TCP fungerar

När det gäller TCP så ser TCP till att det data som skickar kommer levereras i exakt samma ordning och innehåll till mottagaren. Så förutom avsändar- och mottagarport samt storlek och checksumma så innehåller ett TCP-paket data som håller reda på ordningen på paket så att det vet att alla paket kommer fram minst en gång och alla dubbletter tas bort innan mottagarprogrammet ser det. Detta görs genom att mottagaren kontrollerar paketnummer på mottagna paket och om det saknas något så ber det sändaren att sända det igen. Om det kommit för många av samma paket så slänger mottagaren bara bort de extra paketen.

Mottagaren skickar även paket med information tillbaka till avsändaren och talar om vilka paket som har kommit fram på rätt sätt, eller om det behöver sändas om något. Dessa kallas ACK (Acknowledgment) om mottagandet gått bra och NACK (Negative ACK) om det gått dåligt. På så vis vet sändare och mottagare att datat sänds på rätt sätt mellan dem och kan korrigera om det inte går bra.

Det gör ju att ett program som använder TCP behöver inte hålla koll på ordning på paket eller om paket kommer fram som dubbletter eller inte alls. Varför använder inte alla program TCP då? Jo, därför att denna handskakning som TCP gör för att garantera att datat kommer rätt fram tar resurser och tid, vilket inte är så bra för alla saker.

Skillnader på TCP och UDP

Ta exempelvis om man har en ljudöverföring med ett IP-telefonprogram, så är är det bättre att data hoppas över än att dataöverföringen stannar en halv sekund för att ett paket tappats bort. Det tappade paketet blir bara ett litet klick i ljudet, men om all överföring stannar upp, så stannar även ljudet till för att sedan spelas snabbare för att komma ikapp. Så här gör det inte så mycket om man förlorar data, bara överföringen av data i huvudsak går i en jämn takt.

Men om man vill överföra filer eller bilder, så är det viktigt att all data kommer över korrekt, så då gör det inte så mycket om överföringshastigheten går lite ryckigt.

Så, ibland vill man ha UDP, och ibland är TCP bättre.

Man kan se UDP som en brev, där posten kan levererar breven i olika ordning. Det viktiga är att brevet levereras till mottagaren, än att det kommer fram i rätt ordning.

TCP kan då ses som ett telefonsamtal där man förbereder kommunikationen med att slå telefonnumret. Sedan väntar man på att mottagaren svarar för att sedan kunna kommunicera direkt. Sedan få höra allt som motparten säger i rätt ordning och utan att något får förlorat i samtalet.

Servrar (daemons)

De program som normalt skickar data brukar kallas klienter. Exempel på klienter kan vara en webbläsare eller datorpostläsare. De program som lyssnar brukar kallas för server eller i Linux för Daemon och exempel på dessa kan vara `apached` och `postfixd`. Ändelsen `d` på slutet indikerar att det är en server, eller sk *daemon*.

En klient-server har bestämda format, ett så kallade protokoll, som de kommunicerar med varandra. Många av dessa protokoll är textbaserade så att man lätt kan se vad program gör. Andra är binära och blir då svårare att följa. Exempel på textbaserade protokoll är `HTTP` (webbläsare och webb-server) och `SMTP` (datorpostöverföring)

Varje sådant protokoll lyssnar vanligtvis på vissa speciella portar via antingen `TCP`, `UDP` eller i några fall både `TCP` och `UDP`.

För att se vilka program/protokoll som kör och lyssnar på en dator så brukar man använda ett program som heter `nmap`. Det tar och kontrollerar vilka portar som är öppna och kan ibland lista ut vilket operativsystem som körs på datorn samt vilka program som lyssnar bakom respektive port. Mycket användbart för att se vilka tjänster som en dator delar ut till det lokala nätverket och kanske till hela internet.

Vill man sedan experimentera med protokollen för att se vad som händer, så kan man använda programmet `telnet` för att ansluta till en speciell dator och port för transportskiktet `TCP`. Då kan man själv skicka text till servern och se vad som händer när man skriver in kommandon. Det gör det lättare att förstå hur exempelvis en webbläsare eller datorpostprogram gör för att hämta webbsidor eller skicka datorpost.

Vill man se exakt vilka paket som skickas mellan datorer, så kan man med fördel använda ett program som heter `wireshark`. Detta program kan lyssna på alla paket som skickas till eller från datorn via en nätverksanslutning. Det tar även och försöker avkoda data som skickas mellan datorer. För att köra programmet så behöver man rättigheter för att avlyssna av nätverksenheten på datorn. Normalt är det rättigheten `root`, men man kan konfigurera `wireshark` så att man ger rättigheter att avlyssna för alla användare i en speciell grupp.

Dessa program är oumbärliga för en administratör som vill göra sitt nätverk så säkert som möjligt. De är även användbara verktyg för den som olovligen vill ta sig in i andras nätverk och datorer.

Så gör det `nmap`, `telnet`, `ping` och `wireshark` till farliga verktyg? Ja i orätta händer. Skall man förbjuda dessa verktyg då? Nej, för utan dessa verktyg så kan inte någon administratör göra sitt nätverk säkert och se attacker mot nätverket eller analysera hur det fungerar.

Brandvägg

Så, hur kan man skydda sin dator på nätverket mot attacker? Jo, genom att ha bra lösenord, alltid skickar data krypterat över nätverket samt se till att inte någon kommer åt fel saker på datorn. Har du exemplevis ett fildelningsprogram, som `samba` eller `nfs`, igång på datorn för dit nätverk så är några portar öppna på datorn. Detta för annars fungerar inte tjänsterna över nätverket. Vanligtvis är även dessa portar åtkomliga från hela internet om inget görs. Så för att begränsa åtkomsten till dessa port så använder man en brandvägg.

Om man tittar på brandväggsprogram till exempelvis *MS Windows*, så innehåller de mycket mer som inte strikt hör till en brandvägg. Det är exempelvis viruskontroll, kontroll av vilka program som är installerade och vad programmen får göra. Dessa saker är användbara, men strikt inte en

brandvägg, som bara håller på med nätverksanslutningar.

En brandvägg bestämmer vilka datorer som får komma åt vilka portar på datorn. Så med hjälp av en brandvägg så kan man tillåta vissa tjänster, som exempelvis `HTTP`, på maskinen från hela internet så att man kan komma åt webbsidorna som finns i webbservern på maskinen. Men man vill ju inte att hela internet exempelvis skall kunna skicka datorpost via den `SMTP`-server som kör på maskinen, utan bara de maskiner som finns i ditt lokala nätverk. Detta kan man då använda en brandvägg till.

Brandvägg i Linux definieras av något som kallas *`Netfilter`*. Dessa filtrerar alla anslutningar till och från en Linux-maskin så att administratören kan styra vad som släpps igenom eller hindras när andra datorer försöker ansluta och utföra datakommunikation.

För att kontrollerar dessa filter, så finns ett program som heter `iptables` (stödjer bara IPv4, för att ställa in brandväggen för IPv6 använder man istället `ip6tables`). Detta program är dock ganska krångligt att använda. Istället så finns det program som läser en enklare definition och gör alla de avancerade inställningarna göras med `iptables` för att få en fungerande brandvägg.

Det finns många olika program för detta. Men om man vill ställa in brandväggen i routern, så rekommenderar jag programmet `shorewall` (stödjer bara IPv4, för IPv6-brandvägg så kan man då använda `shorewall6`).

För vanliga enklare maskiner så fungerar programmet `ufw` (uncomplicated firewall, har även stöd för IPv6) eftersom det är anpassat för klientmaskiner och servrar med bara ett IP-gränssnitt.

Så hur ställer man då in en brandvägg?

Brandväggen kontrollerar data som kommer till maskinen, som skickas ut från maskinen och som skickas vidare av maskinen till andra maskiner/nätverk. D.v.s. när maskinen fungerar som en router. För varje sådan regel finns det standardinställningar.

De standardinställningar som man kan göra för paket är `ACCEPT`, om det är ett ok paket, `REJECT` om paketet skall nekas och sändaren skall få veta det via ett `ICMP`-paket eller bara `DROP` om det paketet skall kastas bort.

Standardinställningen skall vara restriktiv, d.v.s. man skall slå av alla inkommande anslutningar till maskinen. Inget skall godkännas om det inte anges explicit. Därför brukar man ange att allt inkommande skall göras `REJECT` på. Ibland rekommenderas `DROP`, men då är man inte en god internetmedborgare. Men tyvärr så kan detta ibland utnyttjas av angripare.

När dessa inställningar är gjorda, så öppnar man datorn genom att tillåta de tjänster som man faktiskt vill att andra skall komma åt. Då kan man bestämma vilka transportlager och portar som är tillåtna mellan olika maskiner eller nät. Så exempelvis så kan man ställa in så bara ett protokoll från en viss maskin får ansluta till sin maskinen. Eller så kan man göra så att bara maskiner i det lokala nätverket kan ansluta. Man kan även begränsa antalet anslutningar i minuten från en maskin, för att hindra återkommande inloggningsförsök, exempelvis med protokollet `SSH`.

Utgår man från denna metod, så kan man på ett bra sätt skydda sin dator och nätverk mot de hot som kommer från internet. Men fortfarande ha vissa tjänster tillgängliga från internet och från sitt lokala nätverk.

Referenser

- http://sv.wikipedia.org/wiki/Internetprotokoll
- http://sv.wikipedia.org/wiki/OSI-modellen
- http://sv.wikipedia.org/wiki/TCP/IP
- http://sv.wikipedia.org/wiki/Nätmskadress
- http://sv.wikipedia.org/wiki/TCP
- http://sv.wikipedia.org/wiki/UDP
- http://sv.wikipedia.org/wiki/IPv4
- http://sv.wikipedia.org/wiki/IPv6
- http://en.wikipedia.org/wiki/Netfilter
- http://en.wikipedia.org/wiki/Iptables
- http://en.wikipedia.org/wiki/Shorewall
  - http://www.shorewall.net/
  - http://wiki.debian.org/HowTo/shorewall
- http://en.wikipedia.org/wiki/Uncomplicated_Firewall
  - https://wiki.ubuntu.com/UncomplicatedFirewall
  - https://debdistrosdoneright.com/topics/16-networking/7-ubuntu-firewall