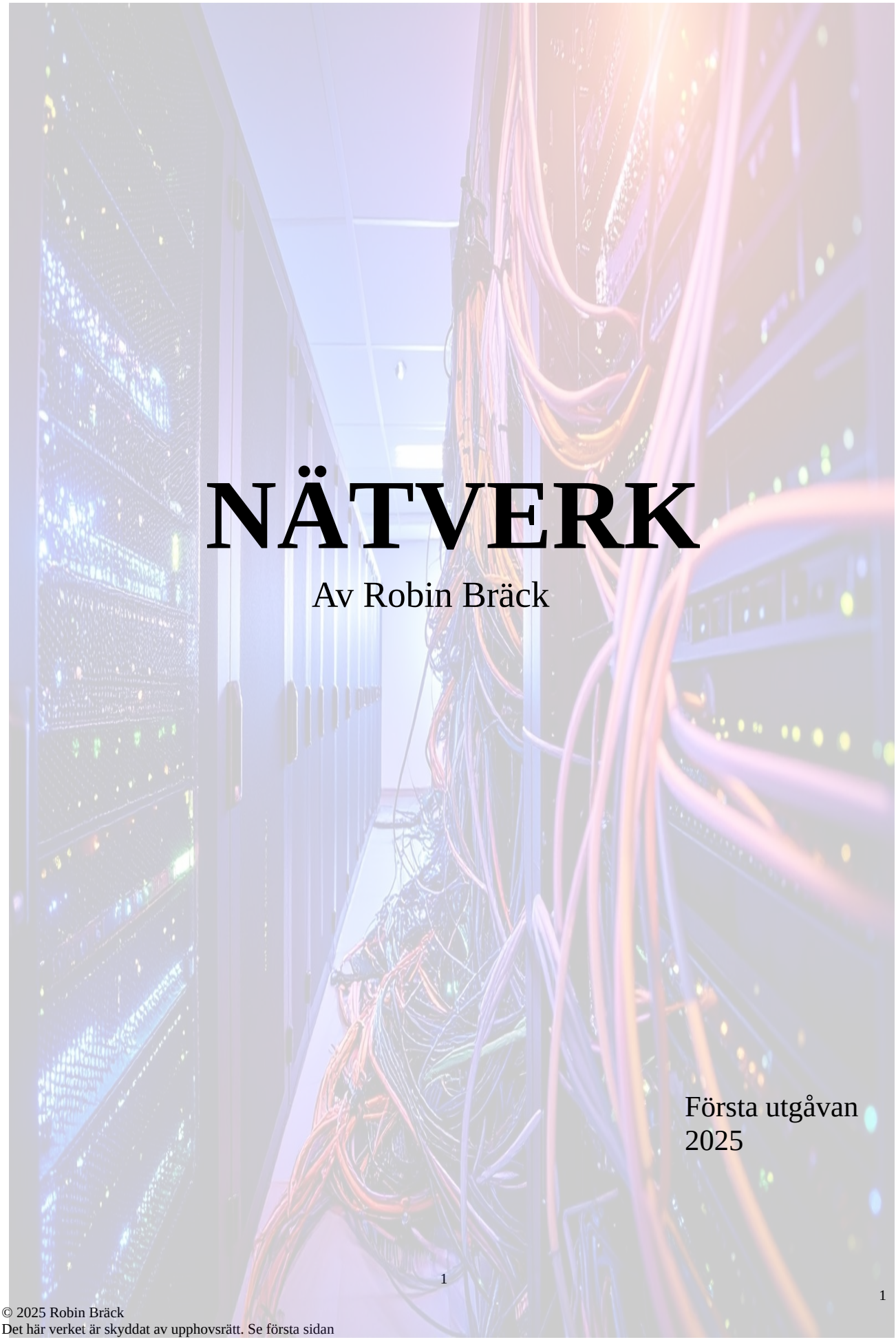

NÄTVERK

Av Robin Bräck

Första utgåvan
2025

1

Licensinformation

2

Index

3

4

5

6

7

8

9

Innan du börjar läsa

Terminologi: WLAN vs “Wi-Fi”

I den här boken använder vi **WLAN** som samlingsnamn för trådlösa lokala nät enligt **IEEE 802.11** (radiogränssnitt, SSID, autentisering, kanaler, m.m.). Begreppet **”Wi-Fi”** är egentligen ett **certifieringsprogram och varumärke** från **Wi-Fi Alliance** som garanterar kompatibilitet mellan produkter som implementerar 802.11. Många operativsystem och menyer skriver ändå “Wi-Fi” om trådlöst nätverk, men vi använder konsekvent **WLAN** för själva tekniken för att vara tydliga – inte minst för att undvika förväxling med **VLAN** (virtuella LAN), som är något helt annat.

Som orientering: **Wi-Fi 4/5/6/6E/7** motsvarar **802.11n/ac/ax/ax (6 GHz)/be**. Att en enhet är “Wi-Fi-certifierad” betyder att den klarar interoperabilitetstesterna för vissa 802.11-funktioner, inte att “Wi-Fi” är ett annat protokoll än 802.11.

10

Kapitel 1 – Vad är ett nätverk?

1.1 Vad är ett nätverk?

Ett nätverk är när två eller flera enheter kopplas samman för att kunna dela information eller resurser, till exempel en skrivare, en server eller en internetuppkoppling. Det kan handla om något så enkelt som två datorer i ett hem, eller något så avancerat som ett globalt nätverk av servrar som driver tjänster som Google, YouTube eller Microsoft Teams.

I dagens samhälle är nätverk en grundläggande del av vår infrastruktur – de finns i skolor, hem, företag, sjukhus, fabriker och nästan överallt där teknik används. Vi använder nätverk när vi skickar ett mejl, surfar på webben, jobbar i molntjänster eller spelar spel online.

1.1.2 Nätverkstyper

Ett nätverk kan se ut på många olika sätt beroende på var det används, hur stort det är och vad det ska göra. För att kunna planera, förstå och felsöka nätverk behöver vi känna till de vanligaste nätverkstyperna.

LAN – Local Area Network

Ett LAN är ett lokalt nätverk som kopplar ihop enheter i ett begränsat område, ofta inom en och samma byggnad. Det kan till exempel vara datorer i ett klassrum, skrivare i ett kontor eller en server i skolans teknikrum. LAN är snabba, stabila och har låg fördröjning.

LAN används ofta med switchar och kabeldragning, men kan även använda trådlösa tekniker (WLAN).

> ***Exempel**: I en skola har varje klassrum flera datorer anslutna till ett LAN. Datorerna kommunicerar med en central server som hanterar inloggningar och filer.*

WAN – Wide Area Network

När ett nätverk sträcker sig över stora geografiska avstånd, t.ex. mellan städer, länder eller kontinenter, kallas det ett WAN. Internet är världens största WAN – ett globalt nätverk som kopplar ihop miljoner LAN.

WAN bygger ofta på hyrda fiberförbindelser, satellit, radiolänkar eller mobilnät. Kommunikation över WAN är långsammare och dyrare än inom LAN, men möjliggör fjärrarbete, molntjänster och globala företagssystem.

> ***Exempel***: *En bank har kontor i olika länder som är ihopkopplade via ett WAN. När du loggar in på din internetbank är det detta nätverk som gör att du når din kontoinformation, oavsett var servern står.*

11

MAN – Metropolitan Area Network

MAN är en typ av nätverk som används i en hel stad eller ett större samhälle. Det är vanligast inom offentlig sektor, t.ex. när skolor, vårdcentraler eller myndigheter kopplas samman via ett stadsnät. MAN byggs ofta ut av kommunala eller kommersiella operatörer och kan fungera som bas för WAN-anslutningar.

> ***Exempel***: *En kommun har ett eget stadsnät där skolor och äldreboenden är ihopkopplade. All trafik går i hög hastighet genom detta MAN innan det går vidare ut till Internet.*

PAN – Personal Area Network

PAN är det minsta nätverket och används av en enskild person – ofta för att koppla samman mobila enheter. Det kan bestå av din mobil, surfplatta, smartwatch och trådlösa hörlurar. Kommunikation sker ofta via Bluetooth eller WLAN i ad-hoc-läge.

> ***Exempel***: När du delar internet från din mobil till en laptop via en personlig hotspot skapar du ett PAN.

1.1.3 Peer-to-peer vs klient–server

Hur enheter är organiserade i ett nätverk har stor betydelse för hur effektivt och säkert det fungerar. De två vanligaste modellerna är peer-to-peer (P2P) och klient–server.

Peer-to-peer (P2P)

I ett P2P-nätverk är alla datorer jämbördiga. Det finns ingen server, utan varje enhet kan både skicka och ta emot filer direkt från andra datorer. Det här är enkelt att sätta upp och kräver ingen särskild serverprogramvara.

Det är vanligt i små hemmiljöer eller för tillfälliga delningar – som när man kopplar ihop datorer via Bluetooth eller delar ut en mapp i Windows.

Men P2P har nackdelar:

- Det blir snabbt rörigt i större nätverk
- Det är svårt att hantera användarrättigheter
- Det är ofta osäkert, eftersom det saknas central styrning

> Tänk dig att varje dator är både chef och anställd – ingen har överblick.

Klient–server

I klient–server-nätverk finns en eller flera servrar som tillhandahåller olika tjänster: inloggning, filhantering, utskrift, e-post, databaser, och så vidare. Alla andra datorer fungerar som klienter – de begär resurser från servern.

12

Denna modell används i nästan alla professionella nätverk – från skolor och företag till sjukhus och universitet.

Fördelar:

- Central administration (konton, behörigheter, säkerhet)
- Lätt att skapa backup och underhålla systemen
- Bättre prestanda och skalbarhet

  Tänk dig en server som en bibliotekarie – klienterna frågar efter resurser, och servern bestämmer vem som får vad.

1.1.4 OSI- och TCP/IP-modellen – en introduktion

Mini-karta (protokoll → lager): ARP=Lager 2, IP/ICMP=Lager 3, TCP/UDP=Lager 4, HTTP/DNS=Lager 7.
När du öppnar en webbläsare och surfar till en hemsida sker flera dolda processer i bakgrunden. För att förstå hur dessa fungerar använder vi modeller som OSI-modellen och TCP/IP-modellen. De hjälper oss att se hur data går från en applikation till en annan – genom kablar, routrar och switchar – och hela vägen till mottagaren.

OSI-modellen

OSI-modellen är en teoretisk referensmodell som delar upp kommunikation i sju lager. Varje lager ansvarar för en specifik funktion i dataöverföringen.

Lager	Namn	Funktion
7	Applikation	Programmet du använder (t.ex. webbläsare)
6	Presentation	Kodning, kryptering, formatering
5	Session	Startar och hanterar sessioner
4	Transport	Säkerställer att data kommer fram korrekt
3	Nätverk	Sköter adressering och routing (IP)
2	Datalänk	Hanterar MAC-adresser, paketering inom LAN
1	Fysiskt	Kablar, signaler, elektriska pulser

OSI är särskilt användbar vid felsökning, eftersom den gör det möjligt att isolera problem. Om nätverkskabeln är urkopplad är det lager 1 (fysiskt) som är felkällan. Om DNS inte fungerar är det lager 7 (applikation).

TCP/IP-modellen

TCP/IP-modellen används praktiskt i dagens Internet och datanätverk. Den är enklare än OSI och innehåller fyra lager:

1. **Applikation** – Appar och protokoll (HTTP, SMTP, DNS)
2. **Transport** – Kontrollerar flöde och felhantering (TCP/UDP)

3. **Internet** – Ansvarar för routing och adresser (IP, ICMP)
4. **Nätverksåtkomst** – All fysisk och elektrisk överföring (Ethernet, WLAN)

TCP/IP är grunden för all modern kommunikation online och är därför extremt viktig att förstå i praktiken.

14

📌 1.1.5 Kontrollfrågor

1. Vad är ett nätverk?
2. Vad står LAN för, och var används det?
3. Ge ett exempel på när ett PAN används.
4. Vad skiljer peer-to-peer från klient–server?
5. Varför är klient–server säkrare än P2P?
6. Vad heter lagret som hanterar IP-adresser i OSI-modellen?
7. Vad är TCP/IP-modellen främst anpassad för?
8. Nämn två fördelar med att använda servrar.
9. Vilket lager i OSI hanterar fysiska signaler?
10. Vad är den största skillnaden mellan OSI och TCP/IP?

15

🔗 1.1.6 Fördjupningslänkar

1. https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/
2. https://www.geeksforgeeks.org/types-of-computer-network/
3. https://www.cisco.com/c/en/us/products/security/what-is-a-network.html
4. https://www.ibm.com/cloud/learn/networking-a-complete-guide
5. https://en.wikipedia.org/wiki/TCP/IP_model

16

✍️ 1.1.7 Egna anteckningar

17

1.2 Nätverkskomponenter och topologier

1.2.1 Switchar, routrar, accesspunkter och brandväggar

För att ett nätverk ska fungera behövs rätt komponenter – alltså fysisk utrustning som kopplar samman enheter och styr hur trafiken rör sig. Det här kapitlet ger dig en överblick över de viktigaste enheterna du kommer att arbeta med i ett nätverk.

Switch

En switch används för att koppla samman flera enheter inom ett lokalt nätverk (LAN). Den tar emot datapaket och skickar dem vidare till rätt mottagare genom att läsa av varje enhets MAC-adress. Switchar bygger upp en MAC-tabell för att hålla reda på vilka enheter som finns kopplade till vilka portar.

En switch arbetar på lager 2 i OSI-modellen (datalänklagret), men det finns även Layer 3-switchar som har vissa routingfunktioner.

Switchar kan vara:

- **Ohanterade (unmanaged)** – enklare, billigare och används i små nätverk.
- **Hanterade (managed)** – ger administratören kontroll över VLAN, hastigheter, portövervakning och mer.

Router

En router kopplar samman olika nätverk med varandra. Den används ofta för att ansluta ett LAN till ett WAN, till exempel när ett hemnätverk kopplas till Internet. Routern ansvarar för att hitta den bästa vägen för varje datapaket att ta, baserat på IP-adresser.

I skol- eller företagsnätverk används ofta mer avancerade routrar eller brandväggssystem (som OPNsense eller Cisco ISR), medan hemnätverk ofta använder enklare kombinerade enheter som innehåller router, switch, accesspunkt och brandvägg i samma låda.

Accesspunkt (WLAN AP)

En accesspunkt gör det möjligt för trådlösa enheter att ansluta till ett LAN genom ett WLAN (Wireless LAN). Accesspunkten tar emot data via radio och skickar vidare den till det trådbundna nätverket – eller tvärtom.

I mindre nätverk är accesspunkten ofta inbyggd i routern. I större installationer, som skolor, används istället fristående accesspunkter som är monterade i tak eller väggar och kopplade till switchar via Ethernet.

Det är viktigt att accesspunkter är rätt placerade för att ge jämn täckning, särskilt i miljöer där många klienter rör sig över stora ytor.

18

Brandvägg

En brandvägg är en komponent – fysisk eller mjukvarubaserad – som filtrerar trafiken till och från nätverket. Den kan blockera eller tillåta trafik baserat på regler som administratören definierar.

Brandväggar kan arbeta med:

- IP-adresser
- Portnummer
- Protokoll (t.ex. TCP, UDP, ICMP)
- Innehåll (t.ex. webbfilter, antivirus, IDS/IPS)

En brandvägg är ett grundläggande skydd och används både på nätverksnivå (gateway-brandvägg) och direkt i operativsystem (lokal brandvägg).

1.2.2 Nätverkskort, kablar och portar

För att kunna kommunicera i ett nätverk behöver varje enhet ett sätt att ansluta till infrastrukturen – det vill säga ett nätverkskort och en fysisk koppling via kabel eller trådlös förbindelse. Här tittar vi på grunderna.

Nätverkskort (NIC)

Ett nätverkskort (Network Interface Card) är hårdvaran som gör att en dator, server eller annan enhet kan kommunicera över ett nätverk. Nätverkskort finns i två huvudtyper:

- **Ethernetkort** – används för trådbundna anslutningar via RJ-45 och switch.
- **WLAN-kort** – används för trådlös anslutning till en accesspunkt (WLAN).

Varje nätverkskort har en inbyggd **MAC-adress** – en unik identifierare som används i LAN-kommunikation. MAC-adressen är fast programmerad och används av switchar för att avgöra vart data ska skickas.

I servermiljöer används ofta nätverkskort med flera portar eller stöd för 1/10/40/100 Gbit/s och VLAN-tagging.

Kablar

Kabeln bär den elektriska (eller optiska) signalen mellan enheter. De vanligaste kabeltyperna är:

- **Twisted Pair (TP)** – används i de flesta LAN. Kategorier som Cat5e, Cat6 och Cat6a används beroende på krav på hastighet och störningsskydd. Kablarna har åtta ledare i fyra par, som tvinnas för att minska störningar.
- **Fiberoptisk kabel** – används för längre avstånd eller höga hastigheter. Överför ljuspulser genom glas eller plastfiber. Immun mot elektromagnetiska störningar.
- **Koaxialkabel (BNC)** – förekommer fortfarande i vissa typer av nät (t.ex. CCTV och vissa stadsnät), men är inte vanligt i nya Ethernet-installationer.

19

En kabels kvalitet påverkar både **hastighet, räckvidd och störningskänslighet**. Längre kablar kräver bättre skärmning.

Portar

En port är det fysiska gränssnittet där du kopplar in en kabel i en enhet. Vanliga porttyper i nätverk är:

- **RJ-45** – används för TP-kablar. Nästan alla nätverkskort, switchar och routrar har denna port.
- **SFP/SFP+** – moduler för fiberoptiska anslutningar. Vanliga i servrar, switchar och nätverksbackbones.
- **USB-C / Lightning** – används ibland för att dela internetanslutningar eller skapa personliga nätverk (PAN), men inte i traditionella LAN.

Portar kan stödja olika hastigheter, till exempel 10/100/1000 Mbit/s (Gigabit Ethernet), och kan ha stöd för PoE (Power over Ethernet) – vilket gör det möjligt att driva t.ex. accesspunkter via nätverkskabeln.

1.2.3 Vanliga topologier

Topologi beskriver hur nätverksenheterna är organiserade – både fysiskt och logiskt. En topologi kan påverka prestanda, skalbarhet, tillförlitlighet och kostnad. Det finns flera klassiska topologier som är bra att känna till.

Stjärntopologi

I en stjärntopologi är alla enheter anslutna till en central punkt, vanligtvis en switch. Detta är den vanligaste topologin i moderna nätverk, eftersom den är lätt att bygga ut och felsöka.

Om en kabel eller en klient går sönder påverkas inte övriga enheter. Men om switchen går ner slutar hela nätverket att fungera.

> *Stjärntopologi är grunden för alla moderna Ethernet-baserade LAN.*

Bustopologi

Bustopologi innebär att alla enheter är kopplade längs en enda gemensam kabel (bussen). Denna topologi användes ofta med koaxialkabel i äldre nätverk, där varje enhet behövde en T-koppling och en terminator i vardera ände.

Den är billig, men har många nackdelar: dålig skalbarhet, svår felsökning och känslighet för avbrott.

> *Bustopologi är idag föråldrad och används endast i vissa specialmiljöer.*

20

Ringtopologi

I en ringtopologi kopplas varje enhet till två andra, vilket bildar en sluten slinga. Data skickas runt i ett bestämt varv tills det når rätt mottagare. I vissa ringnät användes **token passing** för att bestämma vem som fick skicka.

Ringtopologi var vanliga i äldre nätverk (t.ex. Token Ring), men används inte längre i moderna datanät. Dock lever idén vidare i vissa fiber- och transportnät där redundans är viktig.

Mesh-topologi

I en mesh-topologi är varje nod kopplad till flera andra. Detta ger hög redundans och tillgänglighet. Mesh används bland annat i trådlösa mesh-nätverk i hem och företag, i backbone-nät, och i datacenter.

Det finns:

- **Full mesh** – alla noder kopplade till alla andra (dyrt och komplext)
- **Partial mesh** – utvalda noder har flera förbindelser

Mesh-topologi ger god tolerans mot fel men är dyr att implementera i större skala.

Hybridtopologi

De flesta riktiga nätverk använder en blandning av flera topologier – t.ex. stjärna internt och mesh i överliggande struktur. En hybridtopologi kombinerar fördelar från flera olika modeller och används för att skapa flexibla och skalbara nät.

21

📌 Kontrollfrågor

1. Vad är skillnaden mellan en switch och en router?
2. Vilken uppgift har en accesspunkt i ett WLAN?
3. Vad är MAC-adress och varför är den viktig?
4. Vilken typ av kabel används mest i moderna LAN?
5. När används fiberoptik istället för koppar?
6. Vad är en brandvägg och vad används den till?
7. Nämn en fördel och en nackdel med stjärntopologi.
8. Varför används inte bustopologi längre i moderna nätverk?
9. Vad är skillnaden mellan full mesh och partial mesh?
10. Vad menas med hybridtopologi?

22

🔗 Fördjupningslänkar

1. https://www.geeksforgeeks.org/network-devices-hub-switch-router-gateway-bridge/
2. https://www.comptia.org/content/articles/network-topologies
3. https://www.diffen.com/difference/Router_vs_Switch
4. https://www.cisco.com/c/en/us/products/switches/what-is-a-network-switch.html
5. https://en.wikipedia.org/wiki/Network_topology

23

✍️ Egna anteckningar

24

1.3 Protokoll och adressering

1.3.1 Varför protokoll?

Protokoll är **regler** som gör att olika system kan förstå varandra. De definierar hur data ska paketeras, adresseras, överföras och kontrolleras för fel. I praktiken arbetar flera protokoll tillsammans i en **protokollstack** (t.ex. TCP/IP), där varje lager har sin uppgift: från fysisk överföring till applikationer som webbläsare och e-post.

Ett vanligt felsökningsgrepp är att "gå neråt i stacken" tills du hittar lägsta lagret där felet uppstår.

Exempel: *En webbplats laddar inte. Du kontrollerar först om applikationen (HTTP) svarar, sedan om transport (TCP/443) etablerar förbindelse, därefter om Internetlagret (IP) ruttnar rätt, och slutligen om nätverksåtkomst (Ethernet/WLAN) har länk.*

1.3.2 TCP och UDP (transportlagret)

TCP (Transmission Control Protocol) är **förbindelseorienterat**: trevägshandskakning (SYN–SYN/ACK–ACK), kvittenser och omsändning vid paketförlust. Det passar där **tillförlitlighet** är viktigt: webben (HTTP/HTTPS), e-post (SMTP/IMAP), filöverföring (SFTP/FTPS).

UDP (User Datagram Protocol) är **förbindelselöst**: inget handskak, inga kvittenser. Fördel: låg latens och mindre overhead. Vanligt för realtidsapplikationer som VoIP, vissa spel och DNS-frågor.

Portnummer identifierar vilken tjänst som ska ta emot datan. "Välkända portar" (0–1023) används av standardtjänster, "registrerade" (1024–49151) av applikationer, och "dynamiska/privata" (49152–65535) väljs oftast som källport av klienter.

Exempel: *Din klient öppnar en HTTPS-anslutning från källport 51234 till serverns port 443. Svaret går tillbaka till 51234.*

1.3.3 Vanliga infrastrukturprotokoll

Korshänvisning: se avsnitt 2.4 DHCP, DNS och NAT för detaljer och felsökning.

- **ARP (Address Resolution Protocol)** – översätter **IP → MAC** inom LAN. Nödvändigt för att skicka Ethernet-ramar till rätt nätverkskort.
- **ICMP (Internet Control Message Protocol)** – skickar kontrollmeddelanden (t.ex. *ping*). Bra för diagnos, inte för applikationsdata.
- **DHCP (Dynamic Host Configuration Protocol)** – tilldelar automatiskt IP-adress, nätmask, gateway och DNS till klienter.
- **DNS (Domain Name System)** – översätter **namn → IP** (t.ex. *example.se → 93.184.216.34*). Kritisk för nästan all applikationstrafik.
- **NTP (Network Time Protocol)** – synkroniserar tid mellan system; viktigt för loggar, Kerberos och certifikat.

25

Blockeras DNS eller DHCP i fel segment får klienter snabbt “mystiska” problem (ingen adress, eller kan inte nå resurser trots nätverkslänk).

1.3.4 IPv4-adressering: struktur och begrepp

IPv4 består av 32 bitar, skrivs i **punktnotation** (t.ex. 192.168.10.15). **Subnätmasken** avgör hur många bitar som beskriver nätet (**prefix**), t.ex. **/24** motsvarar 255.255.255.0. Resten är **värdbitar**.

Gateway (default gateway) är den routeradress inom ditt subnät som tar trafik **ut** till andra nät.

Viktiga adresser och områden:

- **Privata nät** (RFC1918): 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
- **APIPA/Link-local**: 169.254.0.0/16 (klienter ger sig själva om DHCP saknas)
- **Loopback**: 127.0.0.0/8 (t.ex. 127.0.0.1)
- **Broadcast**: sista adressen i subnätet (t.ex. 192.168.10.255 i /24)

Exempel: *I subnätet 192.168.10.0/24 kan du ofta välja 192.168.10.1 som gateway, använda 192.168.10.10–.200 för klienter, och reservera 192.168.10.210–.220 för servrar och nätverksskrivare.*

1.3.5 Subnät och CIDR

CIDR (Classless Inter-Domain Routing) använder **prefixlängd** (t.ex. /24) i stället för “klass A/B/C”. Att **subnäta** betyder att låna värdbitar till nätbitar för att skapa fler, mindre nät.

En /24 har 256 adresser (0–255), en /25 har 128, en /26 har 64, osv. Antal användbara värdadresser blir alltid totalen minus nät- och broadcastadresser.

Exempel: *192.168.10.0/24 kan delas i två /25: 192.168.10.0–127 (/25) och 192.168.10.128–255 (/25). Varje /25 ger 126 användbara värdar.*

1.3.6 NAT och PAT (IPv4)

NAT (Network Address Translation) översätter privata adresser till publika. **PAT (Port Address Translation)** låter många interna klienter dela **en** publik adress genom att särskilja dem via **portar** (kallas ofta “NAT overload”).

Port forwarding (DNAT) öppnar en publik port och vidarebefordrar den till en specifik intern adress/tjänst (t.ex. publik:443 → 192.168.10.20:443).

Exempel: *Din OPNsense har publik IP 203.0.113.10. Du vidarebefordrar TCP 443 till 192.168.10.20 för en webbserver i DMZ.*

1.3.7 IPv6: grunder och skillnader

IPv6 har 128 bitar och skrivs i **hex** (t.ex. 2001:db8:abcd:12::15). Regler för förkortning:

- Ledande nollor i varje fält kan tas bort.

26

- En sekvens av nollfält kan ersättas **en gång** med `::`.

Adress-typer:

- **Global Unicast** (2000::/3) – motsvarar publika adresser.
- **Unique Local (ULA)** (fc00::/7, vanligt: fdxx::/8) – för interna nät.
- **Link-local** (fe80::/64) – finns alltid på ett gränssnitt; används för grannupptäckt.
- **Multicast** (ff00::/8) – ersätter broadcast-konceptet.

Centrala protokoll/koncept:

- **ICMPv6** (obligatoriskt): eco-request/-reply, fel, och **ND (Neighbor Discovery)**—ersätter ARP.
- **SLAAC** (stateless autoconf) och **DHCPv6** för adressättning.
- **NAT behövs normalt inte** (och undviks); segmentering och brandväggsregler står för säkerhet.

Exempel: *Ett subnät kan vara 2001:db8:100:10::/64. En server kan ha 2001:db8:100:10::20, gateway 2001:db8:100:10::1.*

Dual-stack betyder att köra IPv4 och IPv6 parallellt; vanligt i övergångsperioder.

1.3.8 Gateways och grundläggande routing

Default gateway krävs för att nå andra nät. Finns ingen väg (route) till ett nät, kommer trafiken att skickas mot **defaulten**. På routrar kan du skapa **statiska rutter** (manuellt) eller använda **dynamiska protokoll** (t.ex. RIP, OSPF) som lär sig vägar automatiskt.

I små miljöer räcker ofta statisk routing; i större nät blir dynamisk routing en nödvändighet för skalbarhet och redundans.

1.3.9 Vanliga portar och tjänster

- **20/21** FTP (filöverföring – legacy, ersätt ofta med SFTP/FTPS)
- **22** SSH (säker fjärranslutning)
- **23** Telnet (osäkert, lämnas av historiska skäl)
- **25/587/465** SMTP (e-post ut)
- **53** DNS (UDP/TCP)
- **67/68** DHCP (server/klient)
- **80** HTTP (webb)
- **123** NTP (tid)
- **139/445** SMB (fildelning i Windows)

27

- **389/636** LDAP/LDAPS (katalogtjänster)
- **443** HTTPS (säker webb)
- **3389** RDP (fjärrskrivbord)

*I brandväggsdesign öppnas bara nödvändiga portar **inifrån → ut** och **utifrån → in**; allt annat blockeras som standard.*

28

📌 Kontrollfrågor

1. Varför är protokoll uppdelade i lager i en stack?
2. Nämn två skillnader mellan TCP och UDP.
3. Vilken funktion har ARP i ett LAN?
4. Vad gör DNS och varför är det kritiskt för användare?
5. Förklara relationen mellan IP-adress, subnätmask/prefix och gateway.
6. Vilka är de tre privata IPv4-områdena enligt RFC1918?
7. Vad är fördelarna med PAT jämfört med “ren” NAT?
8. Ange tre centrala skillnader mellan IPv4 och IPv6.
9. Vad är en default gateway och när används den?
10. Ge tre exempel på vanliga portar och tillhörande tjänster.

29

🔗 Fördjupningslänkar

1. https://www.cloudflare.com/learning/network-layer/what-is-arp/
2. https://www.ripe.net/about-us/press-centre/understanding-ipv6/
3. https://www.cisco.com/c/en/us/solutions/enterprise-networks/ipv6/index.html
4. https://www.rfc-editor.org/rfc/rfc1918
5. https://www.rfc-editor.org/rfc/rfc8200

30

✍️ Egna anteckningar

31

1.4 Kommunikation i nätverk – Ethernet och WLAN

1.4.1 Ethernet på datalänk- och fysiskt lager

Förtydligande (PoE-klasser – översikt): 802.3af (upp till 15,4 W), 802.3at (upp till 30 W), 802.3bt (upp till 60–90 W). Kontrollera alltid switchens PoE-budget.
Ethernet beskriver hur data delas upp i **ramar** (frames) och skickas över ett fysiskt medium. En Ethernet-ram innehåller bl.a. **destination MAC-adress**, **källa MAC-adress**, **EtherType/Length**, **payload** (nyttodata) och **FCS/CRC** för felkontroll. MAC-adresser används för att hitta rätt nätverkskort på **lager 2** i OSI-modellen. På **lager 1** (fysiskt) skickas signaler över koppar eller fiber.

Broadcast skickas till MAC ff:ff:ff:ff:ff:ff (alla). Multicast skickas till en adressgrupp. Unicast går till en enda mottagare.

***Exempel:** En klient vill prata med 192.168.10.20 men känner inte mottagarens MAC. Den skickar en ARP-förfrågan som broadcast på Ethernet. Servern svarar med sin MAC, och efter det kan klienten skicka unicast-ramar.*

Standard-MTU för Ethernet är 1500 byte. “Jumbo frames” (t.ex. 9000) används ibland i datacenter och lagringsnät, men kräver att hela vägen stödjer samma MTU.

Switchar lär sig dynamiskt vilka MAC-adresser som finns bakom varje port (MAC-tabell). Okänd destination → **flood** (skicka ut på alla portar i samma VLAN). Länkar i ring kan skapa **loopar**; därför används protokoll som **STP/RSTP** för att blockera redundanta vägar tills de behövs.

PoE (802.3af/at/bt) kan strömförsörja t.ex. accesspunkter och IP-telefoner via nätverkskabeln.

1.4.2 Mediedelning, duplex och kollisioner

Förr delade flera enheter samma kabelsegment och konkurrerade om “taletid” (**CSMA/CD**) – kollisioner uppstod och hanterades i hårdvara. I moderna nät (switch + full duplex) uppstår i praktiken **inga kollisioner**; varje länk har sin egen dedikerade kapacitet.

Autonegotiation sätter hastighet/duplex automatiskt. Felmatchning (t.ex. ena änden forced 100/Full, andra auto) ger kraftig paketförlust och “mystiska” prestandaproblem.

***Exempel:** En skrivare känns “långsam” efter byte av switch. Det visar sig att skrivaren är låst till 100/half medan switchporten är auto (100/full) → duplex-mismatch. När båda sätts till auto försvinner problemet.*

1.4.3 VLAN och trunkar i korthet

Detta avsnitt är en översikt. Detaljer som native VLAN, DTP med mera behandlas i avsnitt 2.2.
Korshänvisning: se avsnitt 2.2 VLAN och segmentering för fördjupning.
VLAN (802.1Q) skapar logiska, separata nät på samma fysiska switch. En **accessport** tillhör ett VLAN. En **trunk** bär flera VLAN mellan switchar/routrar med hjälp av 802.1Q-taggar. VLAN-ID 1–4094 (praktiskt: undvik “native VLAN” exponerat på trunkar).

32

*För trafik mellan VLAN krävs **routing** (t.ex. L3-switch eller "router-on-a-stick"). Segmentering minskar broadcastdomäner och höjer säkerheten.*

1.4.4 WLAN – begrepp och arkitektur

WLAN (IEEE 802.11) ger trådlös åtkomst till LAN. En **accesspunkt (AP)** sänder ett **SSID** (nätverksnamn). Varje AP-radio har en **BSSID** (MAC-adress). En **BSS** är cellen runt en AP; flera BSS som använder samma SSID bildar ett **ESS** (samma logiska nät över större yta).

Band och kanaler

- **2,4 Ghz (802.11/b, 802.11/g, 802.11/n)**: Lång räckvidd, färre icke-överlappande kanaler, mer störningskänsligt.
- **5 GHz(802.11/ac)**: Fler kanaler, högre kapacitet, kortare räckvidd; vissa kanaler kräver DFS.
- **6 GHz (802.11ax i 6 GHz-bandet)**: Många rena kanaler (där det är tillåtet), kortare räckvidd.

I 2,4 GHz används i praktiken 1/6/11 som icke-överlappande kanalval (i vissa länder finns kanal 13, men överlapp måste beaktas).

Kanalbredd (20/40/80/160 MHz) påverkar kapacitet och räckvidd: bredare kanal ⇒ högre topphastighet men större risk för störningar och sämre frekvensåteranvändning.

***Exempel:** I en skola med många klassrum ger 5 GHz på 20 eller 40 MHz ofta bäst total kapacitet (mer frekvensåteranvändning) jämfört med några få breda 80/160-kanaler.*

***Viktigt**: 802.11ax finns även i 2,4/5 GHz-bandet; här avses specifikt varianten i 6 GHz.*

1.4.5 Medietillgång och prestanda i WLAN

WLAN använder **CSMA/CA** (collision avoidance): klienter lyssnar innan de sänder. Mekanismer som **RTS/CTS** kan användas för att minska problem med "hidden nodes" (klienter som inte hör varandra).

Moderna WLAN använder **OFDM/OFDMA** och **MIMO/MU-MIMO** för att utnyttja luften effektivt och serva flera klienter samtidigt. Verklig kapacitet styrs av **SNR**, **interferens**, kanalbredd, antal **spatial streams** och hur många klienter som konkurrerar om **airtime**.

Placering av AP, sändareffekt och dämpning (väggar, metall, glas) avgör täckning. För stark effekt kan försämra roaming och skapa mer ko-kanalsstörning.

1.4.6 Säkerhet i WLAN

Använd **WPA2/WPA3**. Undvik gamla metoder som **WEP** och **WPA/TKIP** (osäkra).

- **WPA2-PSK / WPA3-SAE (Personal)**: Lösenordsbaserat. Enkelt men kräver god lösenordshygien.

33

- **WPA2-Enterprise / WPA3-Enterprise (802.1X/EAP)**: Autentisering via **RADIUS**. Rekommenderas i organisationer (unika identiteter, bättre spårbarhet).
- **Gästnät**: Separat SSID + VLAN-isolering + brandväggspolicyer.

“Dold SSID” ger inte verklig säkerhet. Bygg istället på stark kryptering, separering (VLAN/brandvägg) och uppdaterade klienter/AP-firmware.

Exempel: *Elev-SSID placeras i VLAN 20 med internetåtkomst men blockerat mot server-VLAN. Personal-SSID i VLAN 10 med åtkomst till AD/filresurser. Gäst-SSID i VLAN 30 med strikt egress-policy.*

1.4.7 Roaming och upplevelse

När en klient rör sig mellan celler bör bytet gå snabbt. Stöd för **802.11r (Fast Transition)**, **802.11k** och **802.11v** förbättrar roaming. **Band steering** kan putta dual-band-klienter mot 5 GHz. **Minimum RSSI** gör att klienter släpper svaga AP:er snabbare.

För många SSID försämrar airtime. Håll antalet lågt (ofta 2–4 väl valda) och separera med VLAN och policyer.

1.4.8 Ethernet vs WLAN – när väljer man vad?

- **Stabilitet och förutsägbarhet**: Ethernet vinner (dedikerad länk, låg jitter).
- **Rörlighet och skalbarhet**: WLAN vinner (ingen kabel till varje klient).
- **Toppkapacitet**: Fiber/Ethernet i ryggrad och servrar; WLAN för klientaccess.
- **Säker segmentering**: VLAN/brandvägg på båda, men för WLAN krävs extra noggrann kanal- och SSID-design.

En vanlig design är fast Ethernet till stationär utrustning/servrar och välplanerat WLAN för klienter. Båda binds ihop med VLAN och centrala policyer.

1.4.9 802.11-standarder

Nedan är de vanligaste standarderna som nämns i specifikationer för klienter och accesspunkter. (Vi använder **IEEE-namnen** konsekvent.)

Standard	Band	Teknik i korthet	Typiska kanalbredder	Max teoretisk länkhastighet*	Notering
802.11b	2,4 GHz	DSSS/CCK	20 MHz	11 Mbit/s	Legacy, undvik i nya miljöer
802.11g	2,4 GHz	OFDM	20 MHz	54 Mbit/s	Bakåtkompatibel med b
802.11n	2,4 & 5 GHz	OFDM, **MIMO**	20/40 MHz	upp till 600 Mbit/s (4×4)	Vanlig, men äldre; stöd för både band
802.11ac	5 GHz	**MU-MIMO** (DL), högre	20/40/80/160 MHz	>1 Gbit/s (beroende på streams/bredd)	Endast 5 GHz

34

Standard	Band	Teknik i korthet	Typiska kanalbredder	Max teoretisk länkhastighet*	Notering
802.11ax	2,4/5/6 GHz	QAM **OFDMA**, MU-MIMO (UL/DL), TWT	20/40/80/160 MHz	flera Gbit/s (beroende på streams/bredd)	Effektivare airtime, stöd i 6 GHz där tillåtet

*Max “länk-rate” är teoretisk och förutsätter optimala förhållanden; **verklig throughput är lägre**.

Snabba köptips för elever

- Satsa på **802.11ax** för framtidssäkerhet (gärna stöd för både 2,4 och 5 GHz; 6 GHz där det är tillåtet).
- Välj minst **2×2 MIMO** på klienter; fler streams på AP ger bättre kapacitet för många klienter.
- Kontrollera stöd för **WPA3** och regelbundna firmware-uppdateringar.
- I skolmiljö: AP med **PoE-stöd**, **DFS** (för fler 5 GHz-kanaler) och möjlig **controller/central hantering**.
- Titta på **kanalbredd** (20/40 MHz räcker ofta bättre i miljöer med många celler än 80/160 MHz).

Exempel: *En elev köper en laptop med 802.11ax (2×2) och får stabil prestanda i 5 GHz-bandet i skolan, även när många är uppkopplade samtidigt.*

35

📌 Kontrollfrågor

1. Vilka fält är centrala i en Ethernet-ram och vad används FCS/CRC till?
2. Varför förekommer inte kollisioner i moderna switchade nät?
3. Vad är skillnaden mellan accessport och trunk i 802.1Q?
4. Förklara SSID, BSSID, BSS och ESS.
5. När är 20 MHz kanalbredd att föredra framför 80 MHz i WLAN?
6. Vad gör CSMA/CA och när används RTS/CTS?
7. Varför bör WEP och WPA/TKIP undvikas?
8. Ange två centrala nyheter i 802.11ax jämfört med 802.11ac.
9. Vad innebär MU-MIMO och varför hjälper det i miljöer med många klienter?
10. Nämn tre saker du bör kontrollera när du köper WLAN-utrustning.

36

🔗 Fördjupningslänkar

1. https://standards.ieee.org/standard/802_3/
2. https://standards.ieee.org/standard/802_11/
3. https://www.cisco.com/c/en/us/solutions/enterprise-networks/what-is-wlan.html
4. https://www.wlanpros.com/resources/
5. https://datatracker.ietf.org/doc/html/rfc5416

37

✍️ Egna anteckningar

…

38

1.5 IP-adresser och subnetting

1.5.1 Vad är en IP-adress och varför använder vi dem?

En IP-adress identifierar ett nätverksinterface så att paket kan levereras rätt, ungefär som gata + postnummer för post. Det finns två versioner i bruk: IPv4 och IPv6. ***IPv4*** *har* ***2^{32} = 4 294 967 296*** *möjliga adresser; i praktiken färre för vanliga värdar eftersom många intervall är reserverade (t.ex. privatnät, loopback, multicast). Med fler enheter per person, servrar och sensorer tog IPv4-utrymmet i praktiken* ***slut****, vilket ledde till tekniker som* ***CIDR*** *(effektivare uppdelning),* ***NAT*** *(många interna delar på en publik adress) och hårdare återanvändning.* ***IPv6*** *löser bristen med* ***$2^{128} \approx 3{,}4\times10^{38}$*** *adresser och en hierarkisk modell som skalar – vanligtvis* ***/64 per VLAN*** *i lokala nät.*

Vi skiljer också på **publika** och **privata** adresser (vilka får "synas" på internet). Publika adresser är globalt routbara och tilldelas via internetleverantörer. Privata IPv4-intervall (**10/8**, **172.16/12**, **192.168/16**) används **inom** organisationer och översätts via **NAT** när de ska ut på internet. I IPv6 motsvaras detta av **Global Unicast** (publikt, `2000::/3`) och **ULA** (privat, `fc00::/7`) som inte ska routas på internet; i IPv6 ersätts NAT normalt av tydliga brandväggspolicys. På lagnivå finns dessutom **link-local**-adresser för lokal kommunikation: **169.254.0.0/16** i IPv4 (APIPA) och **fe80::/10** i IPv6.

Exempel: *En privat IP-adress är som ett* ***smeknamn*** *– bara de närmaste "i huset" (ditt lokala nät) använder det. Mamma säger kanske "lilla snuttegubben" hemma, men det säger man inte utanför väggarna. En* ***publik*** *adress är däremot som ditt* ***officiella namn/adress*** *som fungerar var du än är och som alla postkontor (routrar på internet) förstår.*

För att lyckas i resten av boken behöver du tre saker:

1. Se att en adress tillhör ett **interface**, inte "hela datorn".
2. Kunna läsa **prefixlängden** (t.ex. /24) och därmed skilja **Net ID** och **Host ID**.
3. Förstå **publik vs privat** och när **NAT** respektive **brandväggspolicy** används.

Exempel: *Skolan använder privata IPv4-adresser (t.ex. 10.20.0.0/22) på klienterna; brandväggen gör* ***NAT*** *mot en* ***publik*** *adress när trafiken går ut mot internet. I IPv6 får klienterna* ***globala*** *adresser (2001:db8:…) men skyddas av* ***brandväggsregler*** *– ingen NAT krävs.*

1.5.2 Net ID och Host ID (grunden först)

Varje IPv4-adress delas logiskt i **Net ID** (vilket nät/vilken broadcastdomän) och **Host ID** (vilken värd i nätet). **Nätadressen** (alla hostbitar = 0) och **broadcastadressen** (alla hostbitar = 1) reserveras och kan inte tilldelas vanliga hostar (undantag: /31 p2p). Att kunna peka ut dessa snabbt gör resten av subnettingen enkel.

Förtydligande:

- Net ID identifierar **nätet**.

39

- Host ID identifierar **värden i nätet**.
- Antal **användbara hostar** i ett subnet (IPv4) = `2^(hostbitar) − 2`.

Exempel: *`192.168.1.0/24`: Net ID = 192.168.1.0, broadcast = 192.168.1.255, hostspann = .1–.254.*

1.5.3 Subnätmasken – uppbyggnad (binärt → decimal)

Subnätmasken markerar hur många bitar som tillhör Net ID. I binär form är masken en följd av **1:or** (nät) följt av **0:or** (host). Dessa 8-bitars block (oktetter) skrivs i **decimal**:
`11111111` = 255, `11100000` = 224, `11000000` = 192, `10000000` = 128 …
Masken **/27** betyder 27 ettor: `255.255.255.224`. Blockstorleken i den "brytande" oktetten styr **intervallsteget** mellan subnäten.

Förtydligande:

- En oktett är 8 bitar: **128,64,32,16,8,4,2,1**.
- **/X** ↔ mask i decimal (t.ex. /26 ↔ 255.255.255.192).
- **Blockstorlek** = 256 − (oktettens maskvärde), t.ex. 256−224 = **32**.

Exempel: *`192.168.10.37/27` → mask 255.255.255.224 (block 32). Subnetten är .0–31, .32–63, .64–95 … 37 hamnar i **.32–63**, nät = .32, broadcast = .63, hostspann = .33–.62.*

1.5.4 Klass A/B/C – vad de var och deras "defaultmasker"

Historiskt delades IPv4 i klasser: **A** (1.0.0.0–126.0.0.0), **B** (128.0.0.0–191.255.0.0), **C** (192.0.0.0–223.255.255.0). Defaultmaskerna var **/8**, **/16** och **/24**. Klassfullt tänk används inte på internet längre, men är nyttigt pedagogiskt (och för vissa standardexempel). **127.0.0.0/8** reserveras för loopback.

Förtydligande:

- Klass A: default **255.255.255.0?** Nej — **255.0.0.0 (/8)**.
- Klass B: **255.255.0.0 (/16)**.
- Klass C: **255.255.255.0 (/24)**.

Exempel: *Ett "klass C-nät" 192.168.1.0 med defaultmask /24 har **256 adresser** totalt, **254** användbara hostar.*

1.5.5 Hur masken styr antal hostar och subnät

När du **lånar hostbitar** och gör dem till nätbitar skapas fler subnät, men varje subnät får färre hostar. Antal hostar = `2^(hostbitar) − 2`. Antal subnät (inom ett givet ursprungsprefix) = `2^(antal lånade bitar)`.

Förtydligande:

- **/25** (lånar 1 bit från /24): 2 subnät × 126 hostar.

40

- **/26**: 4 subnät × 62 hostar.
- **/27**: 8 subnät × 30 hostar.
- **/28**: 16 subnät × 14 hostar … (se lathund nedan).

Exempel: *Behov: ca 50 klienter/klassrum ur ett 192.168.1.0/24 → välj **/26** (62 hostar) eller **/27** (30 hostar) beroende på krav.*

1.5.6 Räkna ut rätt mask – steg för steg (subnettingmetodik)

1. **Utgå från behov:** antal **subnät** eller antal **hostar** per subnät.
2. **Bestäm antal bitar:** subnät → `2^b ≥ antal subnät`; hostar → `2^h − 2 ≥ antal hostar`.
3. **Härled masken:** lägg b lånade bitar till nätet (öka /X), eller håll h hostbitar kvar.
4. **Rita intervall:** blockstorlek i relevant oktett = 256 − maskoktett.
5. **Skriv ut tabellen:** nät, första host, sista host, broadcast.

Exempel: *Du har **6 segment** á ~20 datorer ur 192.168.1.0/24. Låna **3 bitar** (ger **8 subnät**), kvar blir **5 hostbitar → 30 hostar**/subnät. Mask: **255.255.255.224 (/27)**. Subnät: .0–31, .32–63, .64–95, .96–127, .128–159, .160–191, .192–223, .224–255.*

Exempel: *Samma /27 listat: Subnet 1: 192.168.1.33–62, Subnet 2: 65–94, … Subnet 6: 193–222.*

1.5.7 CIDR-lathund (klass C-utgångsläge /24)

Snabbreferens när du delar ett **/24** (vanligast i övningar):

- **/25:** 2 subnät, 126 hostar vardera, mask **255.255.255.128**
- **/26:** 4 subnät, 62 hostar, **255.255.255.192**
- **/27:** 8 subnät, 30 hostar, **255.255.255.224**
- **/28:** 16 subnät, 14 hostar, **255.255.255.240**
- **/29:** 32 subnät, 6 hostar, **255.255.255.248**
- **/30:** 64 subnät, 2 hostar (p2p), **255.255.255.252**
- **/31–/32:** särskilda/labb (inga vanliga hostar)
  (Se också din PPT-tabell “Kombinationer av subnät”.)

1.5.8 IPv6 – struktur (hex) och Net ID/Interface ID

IPv6 är 128 bitar och skrivs i **hex**: åtta block separerade med `:`. Ledande nollor kan utelämnas per block; **en** följd av nollblock kan komprimeras till `::`. En IPv6-adress delas oftast som **prefix** (Net ID) + **interface identifier** (Host-del), där **/64** är standard för LAN/VLAN (krav för SLAAC och mycket annat).

41

Förtydligande:

- **Global unicast:** `2000::/3` (publikt). **ULA:** `fc00::/7` (internt).
- **Link-local:** `fe80::/10` (automatiskt på varje interface).
- **Loopback:** `::1`.
- **/64** är tumregel för vanliga LAN: 2^{64} adresser per VLAN.

Exempel: *Prefix `2001:db8:1200::/48` delas per hus till /56 (256 st /64 vardera). VLAN i hus A: `2001:db8:1200:00A0::/64` för elev, …`:00A1::/64` för personal.*

1.5.9 IPv6 – “mask”, prefix och lathund (nibbel-tänk)

I IPv6 talar vi om **prefixlängd**, inte mask. Tänk i **nibblar** (4 bitar = en hexsiffra) för enkel huvudräkning. Vanliga snitt:

- **/48:** site-prefix (ger **4096 st /64**).
- **/52:** 16 block à /56 (varje /56 = **256 st /64**).
- **/56:** per byggnad/zon (**256 st /64**).
- **/60:** per våningsplan (**16 st /64**).
- **/64:** per VLAN (standard).

Exempel: *Från `2001:db8:1200::/48` tilldelas B-huset …`:1200:0100::/56`. Där kan du skapa t.ex. …`:0100:0010::/64` (elev) och …`:0100:0020::/64` (personal).*

1.5.10 Vanliga misstag och snabba kontroller

- **Fel mask/gateway** → klienten blir “ensam ö”.
- Räkna **alltid** nät/broadcast i IPv4 (utom /31 p2p).
- Håll **/64** i IPv6-LAN, annars kan funktioner sluta fungera.
- Dokumentera **VLAN ↔ prefix ↔ gateway ↔ DHCP/DNS**.
- Testa IP först, **namn sen** (isolerar DNS-problem).

Exempel: *Du planerar 70 klienter i ett klassrum men väljer /26 (62 hostar) → räcker inte. Välj /25 (126 hostar) eller dela på två /26.*

42

📌 Kontrollfrågor

1. Förklara Net ID och Host ID och ge ett snabbt sätt att hitta nät- och broadcastadress i ett /27-nät.
2. Varför är **/27 = 255.255.255.224** och vad betyder “blockstorlek 32”?
3. Räkna ut nät, första/sista host och broadcast för **192.168.50.77/26**.
4. Du behöver **6** subnät med minst **20** hostar ur **192.168.1.0/24** – vilken mask väljer du och varför?
5. Vad innebär klass A/B/C historiskt och vilka defaultmasker förknippas med dem?
6. När är `/31` praktiskt och varför fungerar det utan broadcast?
7. Skriv **2001:0db8:0000:0000:0000:0000:0042:0001** i kortaste korrekta form.
8. Varför rekommenderas **/64** för IPv6-LAN och vad händer om du använder /80?
9. Hur många /64 ryms i ett /56-prefix? Beskriv hur du skulle fördela dem per VLAN.
10. Vad är den snabbaste vägen att avgöra om ett IPv4-subnet räcker för X hostar?

43

🔗 Fördjupningslänkar

- RFC 791 – Internet Protocol (IPv4): https://www.rfc-editor.org/rfc/rfc791
- RFC 4632 – Classless Inter-Domain Routing (CIDR): https://www.rfc-editor.org/rfc/rfc4632
- RFC 1918 – Private IPv4 Addressing: https://www.rfc-editor.org/rfc/rfc1918
- RFC 3021 – Using 31-Bit Prefixes on IPv4 P2P Links: https://www.rfc-editor.org/rfc/rfc3021
- RFC 8200 – Internet Protocol, Version 6 (IPv6): https://www.rfc-editor.org/rfc/rfc8200
- RFC 5952 – IPv6 Text Representation: https://www.rfc-editor.org/rfc/rfc5952
- RFC 4193 – Unique Local IPv6 Unicast Addresses (ULA): https://www.rfc-editor.org/rfc/rfc4193
- RFC 4862 – IPv6 Stateless Address Autoconfiguration (SLAAC): https://www.rfc-editor.org/rfc/rfc4862

44

✍️ Egna anteckningar

...

45

2.1 Planering och design av nätverk

2.1.1 Mål och krav

Ett bra nätverk börjar alltid med att vi förstår **varför** det ska byggas och **vilka behov** det ska uppfylla. Det handlar om att kartlägga vad användarna faktiskt gör (applikationer, antal samtidiga klienter, från vilka platser), vilken nivå av tillgänglighet som krävs och vilka säkerhetskrav som finns. Denna kravbild blir styrdokumentet för alla kommande beslut: topologi, adressering, segmentering, brandväggspolicys, samt hur vi designar WLAN för kapacitet och täckning. När kravspecifikationen är tydlig minskar risken för dyra omtag senare, och det blir enklare att verifiera att lösningen verkligen motsvarar förväntningarna.

Kom ihåg (förtydligande):

- *Funktionella krav:* antal användare/klienter, applikationer (t.ex. VoIP, video), tjänster (DNS/DHCP), åtkomst (lokal/fjärr), segmentering (VLAN).
- *Icke-funktionella krav:* prestanda (bandbredd/latens), tillgänglighet, säkerhet, skalbarhet, budget.
- *Gränssnitt:* internetkapacitet, publika IP, site-to-site-behov, leverantörsvillkor.

Exempel: *“Gymnasieskola: 600 samtidiga klienter, elev-/personal-/gäst-segment separerade, latency <10 ms inom campus, centralt loggkrav, 99,5 % tillgänglighet skoltid.”*

2.1.2 Arkitektur och topologi

Nästa steg är att välja en arkitektur som är **begriplig, robust och lätt att felsöka**. Den klassiska hierarkiska modellen—**Core → Distribution → Access**—fungerar utmärkt i skolor och campus: core transporterar trafik mellan byggnader och zoner, distribution hanterar routing mellan VLAN och policys, och access kopplar in klienter och accesspunkter. Logiskt använder vi oftast stjärna i accesslagret, med **redundanta uppkopplingar** uppåt. Redundans kräver samtidigt **loopskydd** (STP/RSTP/MSTP). Placeringen av routing avgörs av policys och skalbarhet: antingen i distributionslagret (SVI på L3-switchar) eller centralt i brandvägg/routers.

Nyckelval (förtydligande):

- *Hierarki:* core/distribution/access för enkel felsökning och skalbarhet.
- *Redundans:* dubbla länkar/enheter där det ger värde—och loopskydd på plats.
- *Routing:* L3 i distribution eller centralt bakom brandvägg beroende på policys.
- *Undvik:* långa “daisy chains” av accesswitchar—samla upp till distribution i stället.

Exempel: *“Två distributionsswitchar per hus (stack/vPC), accesswitchar per våningsplan med dubbla uplinks till båda distributionerna.”*

46

2.1.3 IP-plan, VLAN och namnstandard

En konsekvent **IP- och VLAN-plan** är avgörande för drift och felsökning. Börja med att reservera en privat adressrymd (t.ex. 10.0.0.0/8) och dela upp i **byggnads- och funktionsbaserade prefix**. Knyt sedan varje VLAN till ett subnät, definiera **gateway-adress**, **DHCP-scope** och **DNS/NTP-optioner**. Använd en **namnstandard** som kodar plats och roll i enhetsnamnet—det gör dokumentation och felsökning snabbare.

Rekommendationer (förtydligande):

- *VLAN-konvention:* t.ex. VLAN 10 Personal, 20 Elev, 30 Gäst, 40 Servrar, 50 WLAN-AP.
- *Subnettering:* dimensionera för samtidiga klienter + tillväxt.
- *DHCP/DNS:* scopes, reservationer, relevanta optioner (router, DNS, NTP).
- *Namn:* kort, tydligt, inkl. plats/roll (t.ex. *HUS1-SW-A3*).

Exempel: *"VLAN 20 (Elev) = 10.20.0.0/22, gateway 10.20.0.1, DHCP 10.20.0.50–10.20.3.200, DNS 10.40.0.10/11 (AD)."*

2.1.4 Fysisk infrastruktur

Den fysiska miljön är grunden för stabil drift. **Racks och patchpaneler** ska vara märkta och organiserade så att service går snabbt. **Kablage** väljs efter sträckor och kapacitet (Cat6/Cat6a i access; multimode eller singlemode fiber i backbone). Planera **el och UPS**, särskilt om du driver mycket **PoE-last** (accesspunkter, kameror, telefoner). Tänk även på **kylning, låsbarhet och kabelvägar**, så att utrustningen är skyddad och lätt att jobba med.

Checklista (förtydligande):

- *Patchning:* märkning och ordning; separera koppar/fiber.
- *Kablage:* Cat6/Cat6a i access, fiber i backbone.
- *PoE-budget:* summera AP:er, kameror, telefoner—dimensionera switchar rätt.
- *Miljö:* kylning, lås, kabelstegar, 10–30 % expansionsmarginal.

Det är enklare (och billigare) att lägga extra fibrer och några tomma panelplatser vid installation än att göra om senare.

2.1.5 WLAN-design (kanal, täckning, kapacitet)

Ett bra WLAN byggs för **kapacitet och upplevelse**, inte bara täckning. Börja med en **site survey**: kartlägg lokaler, väggmaterial, störkällor och var användare befinner sig. Planera **kanaler och band**: 2,4 GHz (802.11/b/g/n) har lång räckvidd men få icke-överlappande kanaler och mer störningar; 5 GHz (802.11/ac) ger fler kanaler och högre kapacitet men kortare räckvidd (vissa kanaler kräver DFS); 6 GHz (802.11ax i 6 GHz-bandet) erbjuder många rena kanaler där det är tillåtet, men har kortast räckvidd. Håll antalet **SSID** lågt och mappa dem till VLAN. Dimensionera för **samtidiga klienter per cell**, inte bara "en AP per X kvadratmeter".

47

Riktlinjer (förtydligande):

- *Kanalbredd:* 20/40 MHz i miljöer med många celler (bättre återanvändning).
- *SSID-strategi:* 2–4 SSID räcker ofta (t.ex. Personal, Elev, Gäst, ev. BYOD).
- *Roaming:* tillräcklig cellöverlappning; min-RSSI och band steering vid behov.
- *Säkerhet:* WPA2/WPA3 (undvik WEP/TKIP); separera gäst- och personal-trafik via VLAN/brandvägg.

Exempel: *"Personal-SSID (VLAN 10), Elev-SSID (VLAN 20), Gäst-SSID (VLAN 30). 5 GHz prioriteras; 20/40 MHz; min-RSSI för att tvinga klienter att släppa svaga AP:er."*

2.1.6 Säkerhetsdesign och zoner

Säkerhet bör byggas in från början. Dela upp nätet i **zoner** (LAN, Servrar, DMZ, Gäst, Management) och definiera **brandväggspolicys** med **minsta möjliga åtkomst**. Använd **802.1X** för port- och WLAN-autentisering med RADIUS där det är rimligt. Separera **management-trafik** från användartrafik, och logga mot central syslog/SIEM. Tänk också på **multi-factor-autentisering** till kritiska admin-gränssnitt.

Zoner och policys (förtydligande):

- *Gäst → Internet:* tillåtet; blockerat mot LAN/Servrar.
- *Elev ↔ Personal:* styrt efter behov; inga generella "any-any".
- *LAN ↔ Servrar:* tillåt endast nödvändiga portar/tjänster.
- *Management:* åtkomst från admin-subnät/VPN, inte från klient-VLAN.

Segmentering minskar spridning vid incidenter och gör felsökning tydligare.

2.1.7 Drift, övervakning och livscykel

Designa för **drift från dag ett**. Standardisera grundkonfigurationer (NTP, syslog, AAA/SSH-policy, banners). Sätt upp **övervakning** (SNMP/telemetri, larmgränser), och skapa rutiner för **konfig-backup**, **firmware-uppgraderingar** och **ändringshantering**. Håll en enkel assetlista (CMDB light) med plats, serie-nr, IP, VLAN och supportstatus.

Praktiskt (förtydligande):

- *Övervaka:* länkar, PoE-budget, CPU/RAM, fläkt/temp, uplinks, WLAN-airtime.
- *Backuper:* nattliga konfig-dump och versionshantering.
- *Change:* ändringslogg, fönster och tydlig rollback-plan.
- *Dokumentation:* håll topologi/IP/VLAN/ACL-matris uppdaterad.

Ett nät som är lätt att övervaka är också lätt att drifta och felsöka.

48

2.1.8 Test, validering och acceptans

Acceptanstest — kort mall: Länk och VLAN-reachability, DHCP/DNS, inter-VLAN-policyer, WLAN-roaming/kapacitet, redundans (länkbryt/VRRP/HSRP), loggning/övervakning, backup och rollback dokumenterade.
Avsluta med att **bevisa** att nätet uppfyller kraven. Kör funktionstester (DHCP/DNS, inter-VLAN-routing, internetåtkomst), **prestandamätningar** (throughput, latens/jitter), **redundansträning** (bryt länkar, se failover) och en **WLAN-lasttest** under realistisk belastning (lektionstid). Dokumentera utfallet och spara som acceptansprotokoll.

Testpunkter (förtydligande):

- *Funktion:* adressättning, namnupplösning, policyer (tillåt/blockera).
- *Prestanda:* uppmätt throughput mot krav, latens/jitter i real drift.
- *Redundans:* STP-konvergens, uplink-failover, VRRP/HSRP (om använt).
- *WLAN:* roaming, kanalinterferens, verklig airtime vid hög belastning.

Exempel: *“Simulerat fiberbrott mellan distribution A–core: trafiken tog sekundär väg, applikationer obrutna, logg visar failover <1 s.”*

49

📌 Kontrollfrågor

1. Varför bör funktionella och icke-funktionella krav dokumenteras innan design?
2. Beskriv kort core–distribution–access och fördelen med denna modell.
3. Varför kräver redundanta länkar loopskydd som STP/RSTP?
4. Ge exempel på en konsekvent VLAN- och namnstandard.
5. När bör fiber väljas i backbone i stället för TP-kabel?
6. Varför ger 20/40 MHz ofta bättre resultat än 80/160 MHz i tät WLAN-miljö?
7. Ange tre typiska brandväggspolicys mellan vanliga zoner.
8. Varför är 802.1X relevant för både kabel och WLAN?
9. Nämn tre saker du alltid bör övervaka i produktion.
10. Vad bör ingå i ett acceptanstest innan driftsättning?

50

🔗 Fördjupningslänkar

1. https://www.cisco.com/c/en/us/solutions/enterprise/design-zone-campus.html
2. https://www.meraki.com/blog/2017/09/vlan-segmentation-best-practices/
3. https://www.arista.com/en/solutions/campus-design
4. https://wlanprofessionals.com/resources/
5. https://www.rfc-editor.org/rfc/rfc3580

51

✍️ Egna anteckningar

…

52

2.2 VLAN och segmentering

2.2.1 Varför segmentera nät?

Segmentering delar upp nätverket i **mindre, separata domäner** så att broadcasttrafik, fel och attacker inte sprids fritt. Det gör nätet **effektivare, säkrare och lättare att felsöka**. I praktiken görs segmentering oftast med **VLAN** (Virtual LAN), och kompletteras med brandväggspolicys mellan segmenten.

I ett ofragmenterat nät hamnar alla klienter i samma broadcastdomän – en felkonfigurerad klient eller infekterad enhet kan då påverka alla andra.

Förtydligande:

- *Prestanda:* mindre broadcastdomäner minskar onödig trafik.
- *Säkerhet:* begränsar lateral förflyttning; policys mellan VLAN.
- *Felsökning:* tydligare gränser gör orsaker lättare att ringa in.

Exempel: *Separera "Personal", "Elev", "Gäst", "Servrar" och "Management" i egna VLAN. Tillåt endast definierade flöden mellan dem.*

2.2.2 VLAN-grunder: 802.1Q, access och trunk

Ett VLAN är en **logisk** indelning av ett fysiskt nät. På switchportar skiljer vi mellan **accessportar** (tillhör ett enda VLAN) och **trunkar** (bär flera VLAN samtidigt). På trunkar märks ramar med **802.1Q-taggar** som anger VLAN-ID (1–4094).

I vissa fabrikat finns ett förhandlingsprotokoll för trunkning, **Dynamic Trunking Protocol (DTP)**. DTP försöker automatiskt komma överens med motänden om att göra en port till trunk. I en säker och förutsägbar design **stänger man av DTP** och sätter portläget **explicit** (access *eller* trunk), för att undvika oavsiktlig trunkning.

Förtydligande:

- *Accessport:* o-taggad trafik; en klient/port hör till exakt ett VLAN.
- *Trunk:* taggad trafik för flera VLAN mellan switchar/routrar/AP.
- *Native VLAN:* o-taggad trafik på en trunk; håll det **oanvänt/avskilt** för att minska risk.

ISL är legacy och används inte i modern design – 802.1Q är praxis.

Exempel: *SW1–SW2 är ihopkopplade med en trunk som bär VLAN 10/20/30/40. En skrivare på port 12 är access i VLAN 20.*

2.2.3 Inter-VLAN-routing: L3-switch eller “router-on-a-stick”

Eftersom VLAN är separata behövs **routing** för att de ska kunna nå varandra. Det löses antingen i en **L3-switch** via SVI (Switch Virtual Interface) eller med en router/brandvägg via en **trunk** (“router-on-a-stick”).

Förtydligande:

- *SVI:* ett virtuellt interface per VLAN (gateway-adress), hög prestanda i switch.
- *Router/brandvägg:* en subinterface per VLAN; bra när policys ska centreras.
- *Placering:* välj där du enklast kan **tillämpa policy** och **skala**.

Exempel: *SVI VLAN 20 = 10.20.0.1 fungerar som gateway för elever. Brandväggen har policy som tillåter VLAN 20 → internet, men blockerar VLAN 20 → Servrar (VLAN 40).*

2.2.4 Designmönster: vanliga VLAN och betalager

I skolor och campus återkommer några mönster. Separera funktioner och riskprofiler – och lägg rätt **brandväggspolicys** mellan dem.

Vanliga segment (förtydligande):

- *Personal (VLAN 10):* åtkomst till AD, fil och interna system.
- *Elev (VLAN 20):* begränsad åtkomst; inga administrativa resurser.
- *Gäst (VLAN 30):* endast internet; isolerade klienter.
- *Servrar (VLAN 40):* AD/DNS/fil/DHCP, skyddat.
- *WLAN-AP (VLAN 50):* hanterings- och tunnlingtrafik; separera från klient-VLAN.
- *Voice (VLAN 60):* IP-telefoni med QoS (DSCP/802.1p).
- *Management (VLAN 99 t.ex.):* endast för admin via VPN/arbetsstationer.

Voice VLAN kan prioriteras i switch (trust boundary nära telefonen). Gästnät bör ha hårda egress-regler och klientisolering (AP/brandvägg).

2.2.5 Lager-2-skydd: “hårda portar” och kontroll av giftig trafik

För att segmenteringen ska hålla behöver accesslagret förstärkas med **lager-2-skydd**. Här spelar två byggstenar stor roll:
Bridge Protocol Data Unit (BPDU) och **Spanning Tree Protocol (STP)**. En **BPDU** är ett kontrollmeddelande som switchar skickar till varandra för att dela topologi-information. **STP** använder BPDUs för att upptäcka loopar och bestämma vilka länkar som ska vara aktiva respektive blockerade.

Med detta i ryggen låser vi portlägen, validerar kontrolltrafik och stoppar vanliga attacker.

Skydd (förtydligande):

54

- *Stäng DTP/auto trunk:* sätt accessportar **fixed access**; tillåt trunk endast där du behöver den.
- *Port Security:* begränsa antal MAC/adresser per port; lär in/”sticky” vid behov.
- *BPDU Guard/Root Guard:* stäng port som tar emot oväntade BPDUs (BPDU Guard) och hindra oönskad root-förskjutning (Root Guard).
- *DHCP Snooping:* lita endast på legitima DHCP-servrar.
- *Dynamic ARP Inspection (DAI):* stoppa ARP-spoofing (kräver DHCP Snooping).
- *IP Source Guard:* blockera falska käll-IP på accessportar.
- *Storm Control:* stryp broadcast/multicast vid fel.

Exempel: *En elev kopplar in en liten “smart-switch” som börjar skicka BPDU – med BPDU Guard slår porten ifrån och resten av nätet påverkas inte.*

2.2.6 VLAN i WLAN: SSID → VLAN och gästisolering

I trådlösa nät mappas **SSID** till **VLAN**. Det gör att samma säkerhets- och segmenteringsmodell gäller oavsett medium. Gäster får eget SSID och **eget VLAN**, ofta med klient-till-klient-blockering på AP och strikta brandväggsregler.

Förtydligande:

- *Få SSID:* 2–4 väl valda, mappa till rätt VLAN.
- *Personliga/känsliga nät:* 802.1X/Enterprise; unika identiteter.
- *Gäst:* Captive portal om behov; internet-only; rate limit och egress-filter.

Exempel: *Elev-SSID → VLAN 20, Personal-SSID → VLAN 10, Gäst-SSID → VLAN 30. Gäst isoleras klient-vis och får bara TCP/UDP ut mot internet.*

2.2.7 STP, RSTP och MSTP i redundanta VLAN-miljöer

Spanning Tree Protocol (STP) är ett loopskydd som automatiskt blockerar överflödiga vägar i lager-2-nät så att en **trädstruktur** skapas. **Rapid STP (RSTP)** är en förbättrad variant med **snabbare konvergens** vid fel (typiskt millisekunder–sekunder snarare än många sekunder). När nätet innehåller **många VLAN** är **Multiple STP (MSTP)** användbart: du grupperar VLAN i **instanser** och kan låta olika instanser ha olika “root” och länkval, vilket både balanserar last och förenklar drift.

I alla fall är det viktigt att tydligt **placera root-bridge** (oftast i distribution/core), **trimma vilka VLAN** som alls får gå på en trunk (pruning) och se till att **trunkarnas allowed-listor** matchar i båda ändar. Native VLAN ska vara konsekvent konfigurerat och helst **inte användas** för produktionstrafik.

Förtydligande:

- *STP:* grundläggande loopskydd.

55

- *RSTP:* snabbare återhämtning.
- *MSTP:* grupperar VLAN i instanser för lastbalans och enklare hantering.
- *Root-kontroll:* lås root till avsedd plats (distribution/core).

2.2.8 Dokumentation och felsökning

Bra dokumentation är din snabbaste felsökningsresurs. För varje VLAN: **ID, namn, subnät/prefix, gateway (SVI), DHCP-scope, DNS-policy, brandväggspolicys, trunk-rutter och tillåtna portar**. Vid felkontroll: börja i access (länk, port-mode, rätt VLAN) och rör dig uppåt.

Check vid fel (förtydligande):

- *Portläge:* access/trunk korrekt? rätt VLAN?
- *Trunk:* allowed VLANs lika i båda ändar? native lika?
- *SVI:* up/up? rätt IP/mask?
- *DHCP/DNS:* får klient leas? svarar DNS?
- *STP:* blockad länk där du väntar dig? root där du planerat?
- *Tabeller:* MAC/ARP i rätt segment?

Exempel: *Klient i Personal når inte filservern: porten var access i VLAN 20 (Elev) i stället för VLAN 10 – ändra och problemet försvinner.*

56

📌 Kontrollfrågor

1. Vilka tre huvudsakliga skäl finns för att segmentera nätet med VLAN?
2. Förklara skillnaden mellan accessport och trunk samt vad 802.1Q gör.
3. Vad är DTP och varför bör det stängas av i en säker design?
4. Vad är en BPDU och hur använder STP den?
5. När väljer du SVI/L3-switch jämfört med “router-on-a-stick”?
6. Ge exempel på fem vanliga VLAN i en skolmiljö och deras syfte.
7. Vad är “native VLAN” och hur bör det hanteras?
8. Nämn fyra lager-2-skydd du bör aktivera i accesslagret och deras syfte.
9. Beskriv skillnaden mellan STP, RSTP och MSTP och när MSTP hjälper.
10. Lista fem saker du alltid kontrollerar när en klient “hamnat i fel nät”.

57

🔗 Fördjupningslänkar

1. https://www.cisco.com/c/en/us/support/docs/lan-switching/virtual-lans-vlans/10060-3.html
2. https://www.meraki.com/blog/2017/09/vlan-segmentation-best-practices/
3. https://www.hpe.com/us/en/networking/tech-vlan.html
4. https://www.juniper.net/documentation/en_US/junos/topics/concept/vlan-overview.html
5. https://wlanprofessionals.com/resources/ (SSID ↔ VLAN och gästisolering – översikter)

58

✍️ Egna anteckningar

...

59

2.3 Routing

2.3.1 Grunder: hur routrar väljer väg

Routing handlar om att välja **bästa vägen** för ett paket från källa till mål via en eller flera hopp. Varje router underhåller en **routingtabell** med nät (prefix), **nästa hopp** och utgående gränssnitt. När flera rutter matchar samma mål används **längsta-prefix-match** – den mest specifika (t.ex. /27) vinner över en mer generell (t.ex. /24). Om flera **källor** kan lägga in rutter (statisk, RIP, OSPF) avgör **administrativ distans (AD)** vilken källa som anses mest pålitlig, medan **metric** inom ett protokoll jämför alternativa vägar. Vid lika kostnad kan **ECMP** (Equal-Cost Multi-Path) dela lasten över flera lika bra vägar.

Metric är protokollens "kostnadsskala" (hop-count i RIP, bandbreddsbaserad kostnad i OSPF).

Exempel: *Ett paket till 10.20.1.15 matchar både 0.0.0.0/0 och 10.20.0.0/22; längsta-prefix-match gör att /22 väljs.*

2.3.2 Statisk routing – enkelhet och kontroll

Statisk routing innebär att administratören **manuellt** skriver in rutter. Det passar när topologin är liten eller stabil och när man vill ha **full determinism** i vägvalet. En klassiker är att lägga en **standardrutt** (0.0.0.0/0) mot internetleverantören, medan interna nät pekas **explicit** mot rätt nästa hopp. Du kan även använda **"floating static"** (statisk rutt med högre AD) som **backup** till en dynamisk rutt – den aktiveras bara när den dynamiska försvinner. Tänk på att statisk routing måste finnas **åt båda håll** (returväg), annars uppstår asymmetri och svarspaket hittar inte tillbaka.

Fördelar:

- *Enkelt och förutsägbart* – lätt att förstå och felsöka.
- *Ingen protokollchatt* – noll kontrolltrafik och minimal CPU-belastning.
- *Fin kontroll* – du bestämmer exakt vägval.
- *Bra som backup* – "floating static" tar över först vid fel.

Nackdelar:

- *Ingen självläkning* – topologiändringar kräver manuell uppdatering.
- *Skalar dåligt* – många nät = många rader att underhålla.
- *Risk för asymmetri/blackholes* om returvägar missas.

Användningsområden:

- *Små nät / labbar / edge mot internet* (default route).
- *Enkla hub-and-spoke* (spoke pekar mot hub med statik).
- *Backup-vägar* (floating static bakom dynamik).

60

Exempel: *Brandväggen har* `0.0.0.0/0` *mot ISP. Filialsubnet 10.50.0.0/16 pekas statiskt mot VPN-routern. Returväg från hub till filial summeras och annonseras tillbaka.*

2.3.3 Dynamisk routing – när nätet ska anpassa sig själv

Dynamiska routingprotokoll låter routrar skapa **grannrelationer** (adjacencies), skicka **hellos**/uppdateringar och **konvergera** till en **gemensam** bild av nätet. De reagerar när en länk går ner, när ett nytt nät tillkommer eller när en kostnad ändras, och väljer då om väg automatiskt. Det minskar manuellt arbete, särskilt när antalet prefix och länkar växer, men introducerar också **protokollbeteenden** (timers, metrics, filtrering) som man behöver förstå. Grovt finns två familjer: **distance-vector** (t.ex. RIP) som vidareförmedlar information om "avstånd" till nät, och **link-state** (t.ex. OSPF) som delar topologi-information så att varje router själv räknar bästa vägar.

Fördelar (generellt):

- *Självläkning* – anpassar sig vid fel utan manuell insats.
- *Skalbarhet* – enklare när nät och sajter växer.
- *Mindre manuellt pyssel* – färre statiska rader att underhålla.

Nackdelar (generellt):

- *Ökad komplexitet* – grannskap, timers, filtrering, autentisering.
- *Protokolltrafik* – viss overhead och potentiella angreppsytor.
- *Felkonfiguration kostar* – felaktiga parametrar kan ge sämre vägar.

2.3.4 RIP v2 – enkelt och pedagogiskt

Observera: RIP v2 rekommenderas sällan i produktion p.g.a. långsam konvergens, hop-begränsning, enkel metric och loopsäkerhet — bra pedagogiskt, men skala/snabb återhämtning kräver ofta OSPF eller liknande.
RIP v2 är ett **distance-vector**-protokoll som använder **hop-count** som metric (max 15 hopp). Routrar skickar **periodiska uppdateringar** (typiskt var 30:e sekund) till sina grannar. För att minska loopar används tekniker som **split horizon**, **poison reverse** och **hold-down timers**, men vid större eller mer dynamiska nät blir konvergensen ändå **långsam**. Fördelen är att det är **enkelt att konfigurera** och mycket **pedagogiskt** för att förstå routinggrunder.

Fördelar:

- *Mycket enkelt* – snabbt att få igång i labb/mindre nät.
- *Stöd för VLSM/CIDR* (till skillnad från v1).
- *Bred kompatibilitet* – finns på många plattformar.

Nackdelar:

- *Skalar dåligt* – 15 hopp-begränsning.
- *Långsam konvergens* – periodiska uppdateringar; "count-to-infinity"-problem.

61

- *Grovt metric* – tar inte hänsyn till bandbredd/kvalitet.

Användningsområden:

- *Utbildning och små labbnät* (visa principer).
- *Mycket små, statiska miljöer* där enkelhet > prestanda.
- *Temporära proof-of-concept* där snabb uppstart är viktig.

Exempel: *Tre routrar i triangel kör RIP v2. En länk bryts → efter en stund räknar hop-count upp och trafiken hittar alternativ väg, men med märkbar fördröjning.*

2.3.5 OSPF – snabbt och skalbart

Miniprofil: Area 0 som ryggrad; kostnad baserad på bandbredd; DR/BDR i delade L2-segment; summering/filtrering vid area-gränser; autentisering rekommenderas.
OSPF är ett **link-state**-protokoll där varje router annonserar **länkstatus** via LSAs (Link-State Advertisements). Alla i samma område bygger en **topologikarta** och räknar bästa vägar med **Dijkstras algoritm**. OSPF **konvergerar snabbt**, och genom att dela upp nätet i **områden** (med **Area 0** som ryggrad) kan man **skala** till många prefix utan att varje enhet behöver hela världens detaljer. I Ethernet-segment väljs **DR/BDR** (Designated/Backup Designated Router) för att minska antalet grannrelationer. OSPF stödjer **autentisering**, **summering**, **filtrering** och olika **area-typer** (t.ex. stub/NSSA) för att minska brus.

Fördelar:

- *Snabb konvergens* – reagerar snabbt på fel.
- *Skalbar arkitektur* – areas minskar databas och uppdateringar.
- *Finare metric* – bandbreddsbaserad kostnad ger bättre vägval.
- *Många verktyg* – summering, filtrering, autentisering.

Nackdelar:

- *Mer komplext* – kräver förståelse för areas, LSAs, timers och adjacency-tillstånd.
- *Kräver korrekt design* – fel DR/BDR-placering eller area-gränser kan skapa problem.
- *Lite mer overhead* än statik/RIP.

Användningsområden:

- *Campus/enterprise* (medel–stora till stora nät).
- *Miljöer med redundans* där snabb återhämtning behövs.
- *Nät med många VLAN/subnät* där summering/areas ger ordning.

Exempel: *Distributionen i två hus kör OSPF Area 0; våningsplanen är egna areas. Vid fiberbrott floods en LSA, alla räknar om, och trafiken går sekundär väg på under en sekund.*

62

2.3.6 Första-hopp-redundans – VRRP/HSRP

Klienter skickar sin trafik till en **default gateway** i det egna VLAN:et. För att slippa avbrott när en gateway faller använder man **första-hopp-redundansprotokoll**: **VRRP** (öppen standard) eller **HSRP** (Cisco). Två (eller fler) routrar delar en **virtuell IP** och **virtuell MAC**; en är **aktiv/master**, och en **standby** som tar över när den aktiva faller. Övertagandet annonseras ofta via **gratuitous ARP**, så klienterna fortsätter prata med samma adress. Med **preemption** kan den "bästa" enheten ta tillbaka rollen när den är frisk, och med **tracking** kan man sänka prioritet om en kritisk länk bakom gatewayn går ner.

Fördelar:

- *Transparent för klienter* – samma gateway-IP lever vidare vid fel.
- *Enkelt koncept* – lätt att förstå, testa och dokumentera.
- *Flexibelt* – preemption/track ger snabb och rätt failover.

Nackdelar:

- *Ytterligare protokoll att hantera* – prioritet, timers, tracking måste hållas rätt.
- *Ingen automatisk lastdelning* (GLBP kan, men är ovanligt/leverantörsbundet).
- *Löser bara gateway-resiliens* – inte bakomliggande routing- eller brandväggsproblem.

Användningsområden:

- *VLAN-gateways i distribution/core* där hög tillgänglighet krävs.
- *Migrering/underhåll* – byt hårdvara utan klientstopp.

Exempel: *VLAN 10 har gateway 10.10.0.1 (virtuell). Dist-SW-A är aktiv, Dist-SW-B standby. Vid service tar B över – eleverna märker inget.*

2.3.7 Summering och PBR – kontroll och städning

Prefix-summering innebär att annonsera **ett större sammanhållet prefix** i stället för många små. Det "städar" routingtabeller, minskar CPU-belastning och dämpar uppdateringar över länkar mellan sajter. För att påverka vägval för **specifik trafik** kan **Policy-Based Routing (PBR)** användas – t.ex. att viss källadress eller applikation styrs via en sekundär internetlänk eller en proxy. PBR kräver noggrann **dokumentation** och ofta **övervakad next-hop** (t.ex. SLA/track) så att policyn inte skickar trafik mot en död länk.

Fördelar:

- *Summering:* mindre tabeller och mindre brus mellan sajter.
- *PBR:* granular kontroll för särskilda flöden (compliance/prestanda).

Nackdelar:

- *Summering:* kräver planerad adressering; risk att "gömma" fel under en bred annons.

63

- *PBR:* ökar komplexitet; kan bryta “bästa-väg”-logiken och försvåra felsökning.

Användningsområden:

- *Större nät med OSPF* – summera per site/area-gräns.
- *Speciella trafikfall* – dirigera vissa källor/appar via annan länk/proxy.

2.3.8 Felsökning och validering

Felsök routing **lager för lager**: verifiera länkstatus, ARP/ND, gateway, routingtabell och slutligen applikationen. Kontrollera sedan protokollstatus (grannar, timers, tabeller) och **brandväggspolicys** längs vägen. Dokumentera förändringar och mät före/efter när du justerar metric, AD eller summeringar.

Förtydligande:

- *Grundverktyg:* `ping`, `traceroute/tracert`, `arp -a`, `ip neigh`, `ip route`/`route print`.
- *På nätutrustning:* `show ip route`, `show ip ospf neighbor`, `show ip protocols`, `show interfaces`.
- *Vanliga fel:* saknad default, fel nästa hopp, överlappande prefix, asymmetrisk routing, blockerad port i brandvägg.

Exempel: *Klient når internet men inte filialserver: traceroute stannar efter brandvägg; inter-VLAN-regeln saknar tillåtelse mot 10.50.0.0/16 — lägg regel, testa igen.*

64

📌 Kontrollfrågor

1. Vad innebär längsta-prefix-match och varför är den viktig?
2. När passar statiska rutter bättre än dynamiska protokoll?
3. Förklara skillnaden mellan administrativ distans och metric.
4. Nämn två begränsningar i RIP v2 och varför de spelar roll.
5. Hur skiljer sig OSPF:s arbetssätt från RIP:s?
6. Vad är syftet med OSPF-områden och vilken roll har Area 0?
7. Vad gör DR/BDR i OSPF och i vilken typ av nät väljs de?
8. Hur fungerar VRRP/HSRP och varför behövs de?
9. Vad är prefix-summering och vilket problem löser den?
10. Lista fyra kommandon du använder först när routing “inte funkar”.

65

🔗 Fördjupningslänkar

1. https://www.rfc-editor.org/rfc/rfc2453 (RIP v2)
2. https://www.rfc-editor.org/rfc/rfc2328 (OSPFv2)
3. https://www.rfc-editor.org/rfc/rfc5798 (VRRPv3)
4. https://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/7039-1.html
5. https://www.juniper.net/documentation/us/en/software/junos/routing-policy/topics/concept/policy-basics.html

66

✍️ Egna anteckningar

…

67

2.4 DHCP, DNS och NAT

2.4.1 DHCP – grunder

DHCP automatiserar utdelningen av nätverksparametrar till klienter: **IP-adress**, **subnätmask/prefix**, **default gateway** och **DNS-servrar**. När en klient ansluter skickar den en broadcast (**DISCOVER**) för att hitta en server. Servern svarar med ett **OFFER**, klienten accepterar med **REQUEST**, och servern bekräftar med **ACK** – den klassiska **DORA-processen**. Den tilldelade adressen behålls under en **lease** (tidsperiod) som klienten förnyar innan den går ut. Utöver grundvärden kan DHCP skicka **options** (t.ex. router = option 3, DNS = option 6, domänsuffix = option 15, NTP = option 42).

Exempel: Exempel: *En elevdator i VLAN 20 startas och ropar efter adress. Gatewayn vidarebefordrar (relay) DHCP-förfrågan till skolans centrala server. Klienten får 10.20.1.57/22, gateway 10.20.0.1 och DNS 10.40.0.10–11.*

Förtydligande (sammanfattning):

- *DORA:* Discover → Offer → Request → Ack.
- *Lease:* giltighetstid; förnyas automatiskt.
- *Options:* extra parametrar (t.ex. 3, 6, 15, 42).
- *Reservation:* fast IP till specifik MAC (skrivare/servrar).

2.4.2 DHCP-design: scopes, relay och robusthet

I segmenterade nät finns ofta **en central DHCP-server** som servar flera VLAN via **DHCP relay** (ibland “IP helper”) på gatewayn. Varje VLAN får ett **scope** som matchar dess subnät; intervall för **statisk adressering** (gateway, switchar, brandvägg) exkluderas från utdelningen. **Lease time** anpassas efter målgruppen: kortare för gästnät med hög omsättning, längre för personal. För robusthet kör man två servrar i **failover** (load share eller hot-standby). Säkerhet i accesslagret stärks med **DHCP Snooping** som hindrar falska DHCP-servrar.

Exempel: Exempel: *VLAN 30 (Gäst) har lease time 8 h och rate-limits mot internet. VLAN 10 (Personal) har 3–7 dagar. DHCP Snooping är aktivt; endast brandväggens gränssnitt är “trusted” för DHCP.*

Förtydligande (sammanfattning):

- *Relay:* central server når många VLAN.
- *Lease time:* kort i gäst, längre i personal.
- *Failover:* två servrar för hög tillgänglighet.
- *Säkerhet:* DHCP Snooping i accesslagret.

68

2.4.3 DNS – grunder

DNS översätter **namn → IP** (och **PTR** för IP → namn). Systemet är **hierarkiskt** (rot → TLD → zoner) och bygger på att klienter frågar en **rekursiv resolver** som i sin tur hämtar svar från **autoritativa namnservrar**. **Caching** med **TTL** minskar svarstider och belastning. Förutom A/AAAA-poster används **CNAME** (alias), **MX** (e-post), **TXT** (t.ex. SPF/DKIM), **SRV** (tjänster som LDAP/Kerberos i AD) och **NS** (pekar ut namnservrar). I en AD-miljö är **SRV-posterna** avgörande för att hitta domänkontrollanter.

Exempel: Exempel: *Klienten vill till* `files.skolan.local`*. Den frågar skolans resolver, som har zonen autoritativt (AD-integrerad) och svarar direkt: A = 10.40.0.50. Webbläsaren kontaktar 10.40.0.50 utan att behöva känna till IP från början.*

Förtydligande (sammanfattning):

- *A/AAAA, CNAME, MX, TXT, SRV, NS* – vanligaste posterna.
- *Rekursiv resolver* – frågar vidare och cache:ar.
- *TTL* – styr cache-längd.
- *AD-integrerad DNS* – nödvändig för SRV och intern namnlösning.

2.4.4 DNS-design: split-horizon, forwarders och redundans

I skolor används ofta **split-horizon DNS**: interna namn (t.ex. `*.skolan.local`) besvaras internt, medan externa namn (`*.skolan.se`) slås upp mot internet. Rekursiva resolvers kan använda **forwarders** (t.ex. operatörens eller en filtrerande tjänst) för att snabba upp svar och tillämpa policy. Ha **minst två** resolvers för redundans och separera dem gärna i olika VLAN/zoner. Sätt **lämpliga TTL**-värden (kortare för föränderliga poster, längre för stabila). Loggning och **svarstidsövervakning** hjälper både säkerhet och brukarkvalitet.

Exempel: Exempel: *Två AD-integrerade DNS:er (10.40.0.10–11) hanterar* `skolan.local`*. För externa namn använder de forwarders hos ISP. En tredje, läsande resolver i DMZ loggar och alarmerar onormala mönster.*

Förtydligande (sammanfattning):

- *Split-horizon:* interna vs externa zoner.
- *Forwarders:* snabbar upp, kan filtrera.
- *Redundans:* minst två resolvers.
- *TTL:* kort för rörligt, långt för stabilt.

2.4.5 NAT och PAT – grunder och begrepp

NAT44 översätter privata adresser (RFC1918) till publika vid utgång mot internet. Den vanligaste varianten är **PAT** (Port Address Translation, “NAT overload”) där många interna klienter delar **en** publik adress; portnummer skiljer flödena. **SNAT** (käll-NAT) används vid utgående trafik; **DNAT**

69

(destination-NAT, “port forwarding”) används när externa ska nå en intern tjänst. NAT ger **adressöversättning**, men ska inte förväxlas med **säkerhet** – **brandväggsregler** behövs fortfarande. I **IPv6** ersätts NAT i normalfallet av **brandväggspolicy**; man använder globala unicast-adresser och segmentering i stället (NPTv6 kan förekomma i specialfall).

Exempel: Exempel: *Skolan har publik IP 203.0.113.10 på brandväggen. Utgående trafik från VLAN 20 (10.20.0.0/22) SNAT:as till 203.0.113.10 (PAT). För en intern webbserver i DMZ vidarebefordras TCP/443 externt → 192.0.2.20:443 (DNAT/port forwarding).*

Förtydligande (sammanfattning):

- *SNAT/PAT:* många interna → en publik (utgående).
- *DNAT/Port forwarding:* publik port → intern tjänst.
- *NAT ≠ säkerhet:* komplettera alltid med brandväggspolicy.
- *IPv6:* normalt **ingen** NAT; använd brandvägg + segmentering.

2.4.6 Vanliga tillämpningar och fallgropar

NAT och DNS interagerar ofta med andra funktioner. **Hairpin NAT** behövs ibland om interna klienter ska nå en intern tjänst via dess **publika** namn/IP. **Double NAT** (t.ex. operatörs-router + egen brandvägg) komplicerar portöppningar och felsökning; **CGNAT** hos operatören kan omöjliggöra inkommande portar. Vissa protokoll är känsliga för NAT (t.ex. **SIP**, äldre **FTP aktivt läge**), och kräver antingen applikations-inspektion eller modernare lägen. I DHCP är **fel VLAN**, **saknad relay** eller **fel scopes** typiska orsaker till “ingen IP”. I DNS ger fel **forwarders**, **split-horizon**-missar eller **för lång TTL** mystiska fel.

Exempel: Exempel: *Elever kan nå* `intranet.skolan.se` *från internet men inte från klassrummet. Orsak: hairpin NAT saknas – DNAT gäller bara utifrån. Lösning: skapa intern policy och en “NAT loopback”-regel, eller lägg in intern A-post som pekar direkt på DMZ-IP.*

Förtydligande (sammanfattning):

- *Hairpin NAT:* krävs för internt→internt via publik IP.
- *Double/CGNAT:* krånglar portöppning och VPN.
- *Protokoll:* SIP/FTP m.fl. kan behöva särskild hantering.
- *DHCP/DNS:* fel VLAN/relay/TTL/forwarders = vanliga orsaker.

2.4.7 Felsökning: metodik och verktyg

Börja närmast klienten och gå lager för lager. För DHCP: kolla länklampa, rätt VLAN på porten, **DHCP Snooping**-loggar, och på klienten `ipconfig /release /renew` (Windows) eller `dhclient -r; dhclient` (Linux). För DNS: testa namnlösning med **`nslookup`**/**`dig`**, kontrollera vilket namn som efterfrågas (CNAME-kedjor), och rensa cache (`ipconfig /flushdns`). För NAT: verifiera översättningar med brandväggens **NAT-tabell**, gör **paketfångst**

70

på in- och utgränssnitt och kontrollera brandväggsregelns riktning. Vid märkliga fel – testa **direkt med IP** för att skilja DNS- från anslutningsproblem.

Exempel: Exempel: *Klient når IP 10.40.0.50 men inte* `files.skolan.local`*.* `nslookup` *visar att klienten frågar 8.8.8.8 direkt (policy-avvikelse). Åtgärd: tvinga klienter att använda skolans resolvers via DHCP option 6 och brandväggsregler.*

Förtydligande (sammanfattning):

- *DHCP:* rätt VLAN/relay, lease/status på servern.
- *DNS:* `nslookup/dig`, TTL/cache, rätt resolver.
- *NAT:* kontrollera översättningstabell, riktning, hairpin.
- *Avgränsa:* testa med IP vs namn för att isolera felet.

71

📌 Kontrollfrågor

1. Beskriv DORA-processen och vad en lease innebär.
2. Varför används DHCP relay och vilka säkerhetsåtgärder bör aktiveras i accesslagret?
3. Vilka DNS-poster är vanligast och vad används de till?
4. Vad är TTL och hur påverkar den både prestanda och felrättning?
5. Förklara skillnaden mellan SNAT/PAT och DNAT/port forwarding.
6. Varför ska NAT inte ses som en säkerhetsmekanism i sig?
7. När behövs hairpin NAT och vilket alternativ finns i DNS?
8. Ge exempel på hur split-horizon DNS används i en skolmiljö.
9. Nämn tre vanliga orsaker till “ingen IP” i ett segmenterat nät.
10. Vilka steg tar du för att skilja på DNS- och anslutningsproblem?

72

🔗 Fördjupningslänkar

- RFC 2131 – Dynamic Host Configuration Protocol: https://www.rfc-editor.org/rfc/rfc2131
- RFC 1034 – Domain Names: Concepts and Facilities: https://www.rfc-editor.org/rfc/rfc1034
- RFC 1035 – Domain Names: Implementation and Specification: https://www.rfc-editor.org/rfc/rfc1035
- RFC 1918 – Address Allocation for Private Internets: https://www.rfc-editor.org/rfc/rfc1918
- RFC 4787 – NAT Behavioral Requirements for Unicast UDP: https://www.rfc-editor.org/rfc/rfc4787
- RFC 6296 – IPv6-to-IPv6 Network Prefix Translation (NPTv6): https://www.rfc-editor.org/rfc/rfc6296
- RFC 6146 – Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers: https://www.rfc-editor.org/rfc/rfc6146
- RFC 6888 – Common Requirements for Carrier-Grade NATs (CGN): https://www.rfc-editor.org/rfc/rfc6888

73

✍️ Egna anteckningar

...

74

3.1 Grunder i nätverksadministration

3.1.1 Syfte, roller och arbetssätt

Nätverksadministration handlar om att **planera, drifta, övervaka och förbättra** nätet så att användare, servrar och tjänster får den **tillgänglighet, prestanda och säkerhet** som krävs. Arbetet spänner från fysisk infrastruktur (kablar, switchar, brandväggar) till logiska lagret (VLAN, routing, DNS/DHCP, autentisering) och vidare till **processer** (dokumentation, ändringshantering, incidentrutiner). Även i en liten miljö underlättar det att tänka i **roller** – den som designar, den som driver, den som godkänner förändringar – även om samma person ibland gör allt.

Förtydligande (sammanfattning):

- *Mål:* tillgänglighet, prestanda, säkerhet och spårbarhet.
- *Omfång:* fysisk infrastruktur + logiska tjänster + processer.
- *Roller:* design/arkitektur, drift/övervakning, change/incident.

Exempel: *En elevdator kan inte nå intranätet. Administratören följer en standardmetodik (se 3.1.5), dokumenterar observationer och återkopplar enligt skolans incidentrutin.*

3.1.2 Basverktyg och kommandon

Jämförelse (snabb): Cisco ”show ip interface brief” ↔ Linux ”ip -brief addr”; Cisco ”show interfaces status” ↔ Linux ”ip link show”; Cisco ”ping 8.8.8.8 source VlanX” ↔ Linux ”ping -I eth0 8.8.8.8”.
En administratör behöver behärska ett **kärnbibliotek av verktyg** för kontroll, mätning och felsökning – både på klient/servrar och på nätutrustning (CLI).

Klient/Server – vanliga verktyg:

- **Adress/kort:** `ipconfig` (Win), `ip addr` (Linux), `ip link`, `iw` (WLAN).
- **Namn/DNS:** `nslookup`, `dig`.
- **Reachability:** `ping`, `tracert/traceroute`.
- **Rutter/tabeller:** `route print` (Win), `ip route` (Linux).
- **Portar/flöden:** `netstat` (Win), `ss` (Linux).
- **Paketfångst: Wireshark** (GUI), `tcpdump` (CLI).
- **Fjärr: SSH** (säker fjärrterminal), **RDP** (fjärrskrivbord).

Nätutrustning – typiska kommandon:

- **Status:** `show interfaces`, `show mac address-table`, `show arp`.
- **VLAN/routing:** `show vlan`, `show ip route`, `show ip ospf neighbor`.

75

- **Trådlöst:** controller/AP-översikter (kanaler, SNR, klienter).
- **Loggning:** `show logging`, kontroll av syslog/Event Viewer.

Lär dig känna igen "normalvärden" (baslinje) – då ser du snabbare när något avviker.

Exempel: *Du kör* `ping 10.40.0.50` *(OK) men* `nslookup files.skolan.local` *misslyckas. Slutsats: IP-anslutning finns, problemet är DNS (3.1.4).*

3.1.3 Standardkonfigurationer och baslinjer

Baslinje-checklista: NTP mot gemensam källa; central syslog/event; endast SSH (stäng Telnet); unika adminkonton och MFA där möjligt; separat management-VLAN och ACL; stäng DTP; aktivera storm control/Port Security; konfig-backup och versionshantering.
Starka **grundinställningar** gör nätet **förutsägbart och säkert**. Börja varje switch/brandvägg/router med en **baslinje**: tidskällor (**NTP**), loggning (**syslog/Event Viewer**), **AAA** (autentisering/auktorisering), **SSH** (i stället för Telnet), banners, **management-VLAN**, samt versionshantering av konfig.

Förtydligande (sammanfattning):

- *Tid/logg:* NTP till samma klocka; skicka loggar centralt.
- *Åtkomst:* SSH, unika adminkonton, MFA där det går.
- *Management:* separat VLAN/subnät + brandväggsregler.
- *Konfig:* "golden config", backup och diff/historik.

Exempel: *Alla switchar har NTP = 10.40.0.12, syslog till 10.40.0.14, SSH-only, management på VLAN 99 med ACL som endast släpper in admin-verktyg.*

3.1.4 DNS, DHCP och adressering i drift

I vardagen är **DNS och DHCP** navet för klientupplevelsen: kan du få adress? kan du slå upp namn? Administratören sköter **scopes**, reservationer och **relay** i DHCP, samt interna zoner, **split-horizon** och **forwarders** i DNS. Dokumentera **vilket VLAN** som hör till **vilket scope** och **vilka resolvers** klienterna ska använda (option 6).

Förtydligande (sammanfattning):

- *DHCP:* rätt relay, rimlig lease-tid (gäst kort, personal längre).
- *DNS:* AD-integrerad intern zon, två resolvers, vettig TTL.
- *Adressplan:* tabell över VLAN ↔ prefix ↔ gateway ↔ scope.

Exempel: *Gäster klagar på "internet segt". Mätning visar att de går mot publika resolvers direkt. Policy uppdateras: tvinga DNS till skolans resolvers och förbättra svarstid via interna forwarders.*

76

3.1.5 Felsökningsmetodik (lager för lager)

En konsekvent metodik sparar timmar. Arbeta **uppifrån och ner** eller **nerifrån och upp** – men **ändra inte flera saker samtidigt**. Säkra fakta med mätningar/loggar.

Stegvis metod (förtydligande):

1. *Symptom & scope:* vad fungerar/inte? en klient eller många?
2. *Länk och låg-nivå:* lampa, duplex/hastighet, rätt VLAN på porten.
3. *Adress & gateway:* `ip addr/ipconfig`, `ping gateway`.
4. *Namn & rutter:* `nslookup/dig`, `traceroute`, tabeller.
5. *Brandvägg/policy:* loggar och reglers riktning.
6. *Applikation:* port, certifikat, autentisering.

Exempel: *Elev-SSID funkar i A-huset men inte i B. Kontroll visar att B-husets trunk inte bär VLAN 20 – lägg till på allowed-listan, verifiera med klient.*

3.1.6 Övervakning och loggning

Övervakning talar om hur nätet mår; **loggning** talar om vad som hände. Använd SNMP/telemetri för status/larm (CPU, minne, länk, PoE, temperatur) och centrala loggar för brandväggsbeslut, inloggningar och systemhändelser. Visualisering (dashboards) och **larmtrösklar** gör att du kan agera proaktivt.

Förtydligande (sammanfattning):

- *SNMP/telemetri:* länkar, PoE, resursutnyttjande, WLAN-airtime.
- *Syslog/Event:* brandvägg, autentisering, kritiska fel.
- *Dashboards:* nyckeltal (upptid, latens, incidents).

Exempel: *Plötslig airtime-spik i ett våningsplan varje lunch – loggar visar gäst-SSID överbelastat. Åtgärd: extra AP + band steering + min-RSSI.*

3.1.7 Patchning, ändringshantering och backup

Enkla mallar: Ärende (namn, syfte, påverkan, risk/rollback), Ändringslogg (datum, version, ansvarig, sammanfattning), Portetikettering (switch, port, VLAN, beskrivning, plats). Minimikrav för dokument: titel, datum/version, ansvarig, godkännande, revisionshistorik och referenser. Regelbundna **firmware-/säkerhetsuppdateringar** minskar risker. Kör **ändringshantering**: planera fönster, dokumentera, ha **rollback-plan** och verifiera efteråt. Säkerställ **automatiska konfig-backuper** från all nätutrustning och versionera dem.

Förtydligande (sammanfattning):

- *Patchpolicy:* kritiska säkerhetsfixar prioriteras, testa i liten skala först.
- *Change:* vem gör vad, när, hur mäter vi effekt, hur backar vi.

77

- *Backup:* nattliga konfigdump, off-site-kopia, återläsningstester.

Exempel: *Uppgradering av brandvägg kl 18:00. Före/efter-mätning av latens, tydlig rollback. Konfig tas backup automatiskt och diffas mot golden.*

3.1.8 Säker administration och åtkomst

Börja säkert: lokalt adminkonto, nyckelautentisering (SSH), undvik lösenord i klartext, och ta konfigurationsbackup före och efter ändringar.
Tillämpa **least privilege** och separera **admin-åtkomst** från användartrafik. Ha **unika adminkonton**, **MFA** där det går, och en **jump host/bastion** (***jump host/bastion*** *är en hårt härdad mellanserver i ett separat, säkert nät; du loggar in där först och därifrån når du management-IP på utrustningen — ingen direktadministration från det vanliga användarnätet*) för fjärradministration. Logga alla förändringar. Lås accessportar i lager 2 (Port Security, DHCP Snooping, DAI) och skydda STP med **BPDU Guard** (se 2.2).

Förtydligande (sammanfattning):

- *Adminväg:* VPN/jump host, management-VLAN, ACL/brandvägg.
- *Identitet:* unika konton, MFA, tidsbegränsade rättigheter.
- *L2-skydd:* hårda portar, snooping/inspection, storm control.

Exempel: *Admin kopplar via VPN till bastion i management-VLAN. Därifrån SSH till switchar/brandvägg. Endast bastion får nå management-IP.*

3.1.9 Fjärråtkomst och drift på distans

Säker **fjärrdrift** kräver rätt verktyg och vägar. **SSH** används för nät/Unix, **RDP** för Windows. För större miljöer: central hantering av nycklar/certifikat, sessionsloggning och **privileged access**-rutiner. Vid internetavbrott: planera **out-of-band** (t.ex. LTE-modem) för kritiska platser.

Förtydligande (sammanfattning):

- *Protokoll:* SSH, RDP – alltid krypterat.
- *Kanaler:* VPN med stark auth; helst inga öppna admin-portar från internet.
- *OOB:* alternativ väg vid primär-länkned.

Exempel: *Primär fiber är nere – OOB-LTE i serverrummet ger åtkomst till brandväggen för felsökning och omläggning.*

3.1.10 Automatisering och skriptning

Även små miljöer vinner på **automatisering**: återkommande kommandon blir skript; uppgifter standardiseras. **PowerShell** (Windows) och **Bash** (Linux) är utmärkta för inventering, logginsamlning, massändringar och rapporter. Använd **mallar** (templates) för switchkonfigar och lagra dem i versionskontroll.

Förtydligande (sammanfattning):

- *Verktyg:* PowerShell, Bash; på sikt konfighantering (t.ex. mallar).
- *Vinster:* färre manuella fel, repeterbarhet, snabbare återställning.
- *Spårbarhet:* versionshistorik, kodgranskning för kritiska skript.

Exempel: *Ett PowerShell-skript läser domänens katalogstruktur och skapar en CSV över datorobjekt per OU (**OU** = Organizational Unit, en organisatorisk enhet — en grundläggande byggsten i Windows Servers domänhantering där du grupperar datorer och användare för enklare delegering och policy.); samma körning schemaläggs veckovis och postar status i loggkanalen.*

79

📌 Kontrollfrågor

1. Vilka mål styr nätverksadministrationens prioriteringar?
2. Nämn fem verktyg/kommandon du använder i första felsökningssteget och vad de visar.
3. Vad bör ingå i en standard “baslinje” på switchar/routrar/brandväggar?
4. Hur hänger DNS och DHCP ihop med användarupplevelse i vardagen?
5. Beskriv en lager-för-lager-metod för felsökning och ge ett exempel.
6. Vilka nyckeltal och loggar vill du alltid övervaka?
7. Vad innebär ändringshantering, och varför behöver du en rollback-plan?
8. Hur separerar du administrativ åtkomst från användartrafik på ett säkert sätt?
9. När och varför används out-of-band-åtkomst?
10. Ge två praktiska uppgifter som lämpar sig för automatisering med PowerShell/Bash.

80

🔗 Fördjupningslänkar

- Microsoft Learn – PowerShell-grunder: https://learn.microsoft.com/powershell/
- Wireshark – User Guide: https://www.wireshark.org/docs/wsug_html_chunked/
- RPKI/NTP (tid & säker loggning) – NTP Project: https://www.ntp.org/documentation/
- Cisco – Network Troubleshooting Best Practices (översikter): https://www.cisco.com/c/en/us/support/docs.html
- Debian/Ubuntu – iproute2 (ip, ss, tc): https://man7.org/linux/man-pages/man8/ip.8.html

81

✍️ Egna anteckningar

...

82

3.2 Verktyg: ping, traceroute, ipconfig, netstat etc.

3.2.1 Varför dessa verktyg?

När något “inte funkar” gäller det att snabbt avgöra **var** felet finns: nätlänk, adressering, namnupplösning, routing eller själva tjänsten/porten. De klassiska verktygen – **ping**, **traceroute/tracert**, **ipconfig** (Windows) / **ip** (Linux), **netstat/ss**, samt **nslookup/dig**, **arp/route** och vid behov **Wireshark/tcpdump** – låter dig testa kedjan steg för steg. Tanken är att **bekräfta det som fungerar** och ringa in det första lagret där det **slutar** fungera.

Förtydligande (sammanfattning):

- *Länk & IP:* ipconfig/ip, ping gateway.
- *Namn:* nslookup/dig.
- *Vägval:* tracert/traceroute.
- *Tjänst/port:* netstat/ss lokalt, `test-netconnection` (Win) eller `nc` (Linux).
- *Djupanalys:* Wireshark/tcpdump.

Exempel: *En elev säger “internet är nere”. Du pingar skolans gateway (OK), pingar 8.8.8.8 (OK), men* `nslookup google.com` *faller. Slutsats: problemet är DNS, inte “internet”.*

3.2.2 ping – når vi fram överhuvudtaget?

ping använder ICMP (echo request/reply) för att mäta **nårbarhet**, **latens** och **förlust**. Börja nära: ping **egen IP**, **gateway**, sedan **extern IP**. Tänk på att brandväggar kan blockera ICMP trots att applikationer fungerar – ping är alltså ingen garanti för att en **tjänst** är uppe, bara att vägen är öppen för ICMP.

Förtydligande (sammanfattning):

- *Windows:* `ping 1.1.1.1`, `ping -4/-6`, `ping -t` (kontinuerlig).
- *Linux:* `ping -c 10 1.1.1.1`, `ping -I eth0 10.0.0.1`.
- *Tolkning:* hög latens/jitter och paketförlust → nätproblem; 100 % förlust kan vara blockerat ICMP.

Exempel: `ping 10.20.0.1` *(gateway) fungerar men* `ping 8.8.8.8` *tappar paket – felet ligger troligen “utåt”, mellan gateway och internet.*

3.2.3 traceroute / tracert – var tar trafiken vägen?

traceroute (Linux) / **tracert** (Windows) visar **varje hopp** på vägen till ett mål genom att successivt öka TTL och läsa svaren. Används för att se **var** trafiken stannar eller går omvägar. Stjärnor `*` betyder att ett hopp inte svarade (kan bero på brandvägg eller policys). Windows använder som standard ICMP, Linux ofta UDP – växla vid behov (`traceroute -I` för ICMP).

83

Förtydligande (sammanfattning):

- *Windows:* `tracert -d 8.8.8.8` (`-d` hoppar över DNS-upplösning).
- *Linux:* `traceroute -I 8.8.8.8`, eller **mtr** för livevy.
- *Läsning:* privata IP i mitten = NAT/operatörsled; stopp mellan hop N och N+1 = trolig blockering/fel där.

Exempel: *`tracert intranet.skolan.se` stannar vid brandväggen → kontrollera NAT/brandväggsregler och intern rutt mot DMZ.*

3.2.4 ipconfig (Windows) / ip (Linux) – grunder och snabbrengöring

Här ser du **adressering, gateway, DNS, suffix, DHCP-status** och länkläge. I Windows använder du **ipconfig**, i Linux moderna **ip**-verktyget (ersätter ifconfig).

Förtydligande (sammanfattning):

- *Windows:* `ipconfig /all` (allt), `/release` → `/renew` (DHCP), `/flushdns` (töm cache), `/displaydns` (visa cache).
- *Linux:* `ip addr`, `ip link`, `ip route`, `resolvectl status` (systemd-baserat).
- *Snabbkoll:* Har interfacet IP? Rätt VLAN? Rätt gateway/DNS?

Exempel: *Klienten har 169.254.x.x (APIPA) → DHCP når inte fram. Kontrollera DHCP relay och VLAN-tillhörighet på porten.*

3.2.5 netstat / ss – lyssnar tjänsten, och vem pratar den med?

netstat (Windows) och **ss** (Linux) visar **lyssnande portar** och **aktiva anslutningar**. Perfekt för att verifiera att en server **verkligen lyssnar** på rätt port, eller för att hitta vilket program som håller en port.

Förtydligande (sammanfattning):

- *Windows:* `netstat -ano` (alla + PID), `netstat -anob` (kräver admin). Matcha PID med `tasklist /FI "PID eq 1234"`.
- *Linux:* `ss -tulpen` (TCP/UDP, lyssnar, process, port, nät).
- *Tolkning:* "LISTEN" finns lokalt men klienten når inte → sannolikt brandvägg/routing mellan.

Exempel: *Webbservern svarar inte externt. `netstat -ano` visar `LISTENING :443` på servern → felet ligger troligen i brandväggen eller DNAT.*

3.2.6 nslookup / dig – namnen som allt hänger på

DNS är beroende för nästan alla applikationer. **nslookup** (Windows/Linux) och **dig** (Linux) låter dig fråga *exakt vilken resolver* du vill, och vilken **posttyp** du behöver (A/AAAA/MX/SRV).

84

Förtydligande (sammanfattning):

- *Snabbtest:* `nslookup files.skolan.local 10.40.0.10` eller `dig A example.com @1.1.1.1`.
- *Fördjupning:* `nslookup -type=SRV _ldap._tcp.skolan.local` eller `dig +trace example.com`.
- *Symptom:* Namn funkar internt men inte externt → split-horizon/forwarder-fråga.

Exempel: *`nslookup intranet.skolan.se` svarar med publik IP, men intern åtkomst kräver DMZ-IP → lös via intern A-post eller hairpin NAT (se 2.4).*

3.2.7 arp / route – lager-2-cache och standardvägar

arp visar **MAC ↔ IP-cache** för IPv4 (Linux: `ip neigh`; för IPv6 används ND). **route print** (Windows) / `ip route` (Linux) visar **routingtabellen** – kolla särskilt **default gateway** och eventuella specifika statiska rutter som skär av trafiken.

Förtydligande (sammanfattning):

- *ARP/ND:* saknas uppslag till gatewayens MAC → ingen L2-väg (VLAN/port?).
- *Default:* dubbla default-rutter kan skapa slumpmässig eller asymmetrisk trafik.
- *Specifika rutter:* en felaktig /32 kan “stjäla” ett mål.

Exempel: *`route print` visar två default gateways (WLAN + Ethernet). Klienten skickar ibland via fel länk → stäng av ena interfacet eller prioritera rätt.*

3.2.8 Wireshark / tcpdump – när du behöver se paketen

När verktygen ovan inte räcker fångar du **paket**. **Wireshark** (GUI) är lätt att filtrera i, **tcpdump** (CLI) är perfekt via SSH eller på server/brandvägg. Fånga **så nära felet som möjligt** och filtrera snävt (host/port/protokoll) för att undvika brus, och tänk på **sekretess** när du fångar riktig trafik.

Förtydligande (sammanfattning):

- *tcpdump:* `tcpdump -i eth0 host 10.40.0.50 and port 443`.
- *Wireshark:* filter “ip.addr==10.40.0.50 && tcp.port==443”.
- *Tolkning:* SYN utan SYN/ACK → block/ingen lyssnare; trevägshandskak komplett men app felar → applikationslager.

Exempel: *Klient når serverns 443 men inloggning snurrar. Fångst visar TLS-fel (för gammal cipher). Åtgärd: uppdatera serverns TLS-policy/klient.*

3.2.9 Ett snabbt felsökningsrecept (kedjan från klient till tjänst)

Snabb affisch (länk → app): IP → Gateway → Extern IP → DNS → Rutt → Port/tjänst → Applikation.

85

Börja lokalt och rör dig utåt. Säkerställ länk → IP → gateway → namn → väg → port → applikation. Bygg vana att **dokumentera varje steg** (kommando + resultat), så blir återkopplingen till användaren och kollegor rak.

Förtydligande (sammanfattning):

1. *ipconfig/ip* (har vi IP, rätt VLAN/gateway/DNS?).
2. *ping gateway* (lokal nåbarhet).
3. *ping extern IP* (internet utan DNS).
4. *nslookup/dig* (namnupplösning).
5. *tracert/traceroute* (var stannar trafiken?).
6. *netstat/ss* (lyssnar servern? portar?).
7. *Wireshark/tcpdump* (vid behov).

Exempel: *Elev kan inte nå* `files.skolan.local`*. 1) IP OK. 2) Ping gateway OK. 3) Ping 10.40.0.50 OK. 4)* `nslookup files.skolan.local` *ger fel IP. 5) Uppdatera intern DNS → problemet löst utan att "röra nätet".*

86

📌 Kontrollfrågor

1. Vilken kedja av tester kör du för att skilja på DNS-fel och “internet ligger nere”?
2. Vad säger `ping` dig om vägen, och vad säger det **inte**?
3. När använder du `tracert/traceroute -d` och varför kan `*` visas i utskriften?
4. Vilken information letar du efter i `ipconfig /all` respektive `ip route`?
5. Vad betyder en 169.254.x.x-adress och vilka två saker kontrollerar du först?
6. Hur använder du `netstat -ano` eller `ss -tulpen` för att verifiera att en tjänst verkligen lyssnar?
7. Hur testar du namnupplösning mot en **specifik** resolver med `nslookup`/`dig`?
8. När väljer du paketfångst (Wireshark/tcpdump), och vilket minimifilter sätter du först?
9. Vilket problem kan dubbla default-gateways orsaka och hur ser du det?
10. Ge ett exempel där `traceroute` visar rätt väg men appen ändå inte fungerar – vad undersöker du då?

87

🔗 Fördjupningslänkar

- Microsoft Learn – Windows-kommandon (översikt): https://learn.microsoft.com/windows-server/administration/windows-commands/windows-commands
- ping (Windows): https://learn.microsoft.com/windows-server/administration/windows-commands/ping
- tracert (Windows): https://learn.microsoft.com/windows-server/administration/windows-commands/tracert
- ipconfig (Windows): https://learn.microsoft.com/windows-server/administration/windows-commands/ipconfig
- netstat (Windows): https://learn.microsoft.com/windows-server/administration/windows-commands/netstat
- Test-NetConnection (PowerShell): https://learn.microsoft.com/powershell/module/nettcpip/test-netconnection
- `ip` (Linux, man-sida): https://man7.org/linux/man-pages/man8/ip.8.html
- `ss` (Linux, man-sida): https://man7.org/linux/man-pages/man8/ss.8.html
- traceroute (Linux, man-sida): https://man7.org/linux/man-pages/man8/traceroute.8.html
- Wireshark – User Guide: https://www.wireshark.org/docs/wsug_html_chunked/
- tcpdump – man-sida: https://www.tcpdump.org/manpages/tcpdump.1.html
- `dig` – BIND9-dokumentation: https://bind9.readthedocs.io/en/v9.18.0/dig.html

88

✍️ Egna anteckningar

…

89

3.3 Dokumentation, etikett, felsökning

3.3.1 Varför dokumentation?

Dokumentation gör nätet **förutsägbart, överlämningsbart och felsökningsbart**. Utan uppdaterade kartor, IP-/VLAN-planer, portlistor och ändringsloggar blir varje fel en gissningslek och varje uppgradering en chansning. Bra dokumentation sänker MTTR (Mean Time To Repair), minskar dubbelarbete och gör att nya i teamet snabbt blir produktiva. Den skapar också **spårbarhet** – du kan visa vad som ändrades, när och varför, vilket är ovärderligt vid incidenter och revisioner.

Exempel: *En skrivare slutar fungera efter ett switchbyte. Tack vare aktuell portmatris ser du direkt att skrivaren satt i VLAN 20 men nya porten är default VLAN 1. En rad ändring, klart.*

3.3.2 Vad ska dokumenteras?

Minsta gemensamma nämnare är: **topologidiagram**, **IP-/VLAN-plan**, **enhetsinventarie**, **konfigbackup**, **portmatriser**, **brandväggspolicys** och **ändringslogg**. Lägg till **runbooks** (steg-för-steg för vanliga åtgärder) och **driftjournal** (vad som hände, när, varför). Tänk “**en sanning**”: en central plats där allt hålls uppdaterat, med tydlig ägare per dokument och versionsmärkning så att historiken går att följa.

Förtydligande (sammanfattning):

- *Topologi:* core/distribution/access, uplinks, trunkar, redundans.
- *IP/VLAN:* prefix, gateways, DHCP-scopes, DNS/NTP.
- *Enheter:* modell, serienummer, plats, management-IP, version.
- *Portmatris:* vägguttag ↔ patchpanel ↔ switchport ↔ VLAN.
- *Policys:* översikt över tillåtna flöden mellan zoner.
- *Ändringslogg:* datum, syfte, risk, utförare, rollback, resultat.

Exempel: *En enkel runbook “Lägga till nytt VLAN” innehåller: skapa VLAN, SVI + DHCP-scope, tillåta på relevanta trunkar, uppdatera brandvägg, dokumentera i IP-plan och portmatris, validera med testklient.*

3.3.3 Namnstandard, märkning och port-etikett

Konsekventa namn gör att du “läser” nätet som en karta. Namnge enheter med **plats–roll–löpnummer** (t.ex. *HUS1-DIST-A*, *HUS1-ACC-03*). Märk **vägguttag**, **patchpanel**, **switchport** och **fiber** på båda sidor. Etikett i nätet handlar också om hur du lämnar efter dig: **återställ patchning**, **ta bort tillfälliga labbgrejer**, och **spara konfig**. En tydlig namngivning sparar tid i alla led – från beställning och felsökning till städning efter projekt.

Förtydligande (sammanfattning):

- *Namn:* kort, meningsfullt, konsekvent.

90

- *Portdisciplin:* dokumentera innan du drar om; inga “dolda” hopkopplingar.
- *Märkning:* tydliga etiketter + matchande i dokumentationen.

Exempel: *Port A3 i HUS2-ACC-04 märks “Klassrum 214 – skrivare” och loggas i portmatrisen. Vid fel ser du portens VLAN på två sekunder.*

3.3.4 Ärenden och kommunikation – etikett i drift

Felsökning sker inte i vakuum. Ha en **ärendeprocess**: ta emot felanmälan, bekräfta mottagning, sätt prioritet, återkoppla vid milstolpar och stäng ärendet med **kort RCA** (root cause analysis). Var **transparent**: skriv vad du mätt och ändrat. Var **blameless** i analys – fokusera på system och process, inte personer. God kommunikation reducerar frustration, skapar förtroende och gör att alla vet vad som händer – även när lösningen dröjer.

Förtydligande (sammanfattning):

- *Prioritet:* påverkan + brådska → P1–P4.
- *Status:* uppdatera när något ändras – även “vi undersöker” hjälper.
- *RCA light:* orsak, åtgärd, hur vi undviker återkomst.

Exempel: *“P2: Elevnät i B-huset påverkat. Åtgärd: trunk bar inte VLAN 20. Fix: tillåta VLAN 20 på uplink. Förebygg: mall för trunkar + checklista vid switchbyte.”*

3.3.5 Change management – planera, testa, backa

Ändringar ska vara **förutsägbara**. Använd **underhållsfönster**, skriv en kort **risk- och effektbedömning**, och ha en **rollback-plan**. Testa i labb eller på en **pilot** först. Efter ändringen: mät och dokumentera utfallet. En enkel checklista minskar risken för “små missar” som ger stora effekter, och gör att samma kvalitet kan upprepas gång på gång.

Förtydligande (sammanfattning):

- *Plan:* syfte, omfattning, risk, testplan, rollback.
- *Tid:* skoltid vs icke-skoltid – minimera påverkan.
- *Efterkontroll:* mät latens/throughput, kontrollera loggar/alarms.

Exempel: *Firmware på distributionsswitchar: pilot i HUS3, sedan rullas samma version ut övriga hus med samma checklista och rollback.*

3.3.6 Felsökningsmetodik – från symptom till rotorsak

Börja med **scope** (en enhet eller många? ett VLAN eller alla?), definiera **symptom** och **förväntat beteende**, och jobba **lager för lager** (se 3.1 och 3.2). Verifiera med **mätbara tester** i varje steg och **ändra inte flera saker samtidigt**. Skriv ned *vad du testade och vad resultatet blev* – det blir dokumentation i realtid och hjälper nästa person som tar över ärendet.

Förtydligande (sammanfattning):

91

- *Scope → Hypotes → Test → Resultat → Nästa steg.*
- *Isolera:* byt variabel (kabel, port, VLAN) en i taget.
- *Bevis:* spara kommandon/utdata i ärendet.

Exempel: *“Ingen IP i sal 215.” Test:* `show int` *OK, port i VLAN 20 OK, men klienter får APIPA.* `show ip dhcp snooping` *visar block – uplink ej trusted. Åtgärd: trust uplink, verifiera* `ipconfig /renew`*.*

3.3.7 Säkerhetsetikett i vardagen

På delade nät (skola!) gäller “**gör ingen skada**”. Starta inte egna **DHCP/DNS** på klientnät, scanna inte nät blint, och kör inte osignerade verktyg på produktionsservrar. Paketfångst kan innehålla känsliga data – **minimera, filtrera, radera**. Admin-gränssnitt nås via **jump host/bastion** i **management-VLAN**, inte direkt från klientnät. *(En **jump host/bastion** är en hårt härdad mellanserver i ett separat, säkert nät; du loggar in där först och därifrån når du management-IP — ingen direktadministration från användarnätet.)* Etiketten handlar också om att lämna spår: skriv vad du gjorde och varför, så att nästa person kan förstå förändringen.

Förtydligande (sammanfattning):

- *Rogue-tjänster:* förbjudet på elev-/personalnät.
- *Fångst:* minst möjligt, lagra säkert, rensa efteråt.
- *Åtkomst:* VPN/jump host, MFA, minst behörighet.

Exempel: *Ett “mystiskt” DHCP-scope dök upp. Med DHCP Snooping spårar du källporten och hittar en elevs hemmarouter – porten stängs och incidenten loggas.*

3.3.8 Mallar och checklistor

Standardisera vanliga jobb med **mallar** (VLAN-skapande, SSID-sättning, ny switch i access, nytt subnät, ny server-VM). Lägg **checklistor** i samma dokumentkällor som konfigmallarna så att de hålls uppdaterade samtidigt. När checklistan samtidigt är en **valideringslista** (mätpunkt per steg) blir kvaliteten jämn och lätt att revidera.

Förtydligande (sammanfattning):

- *En källa:* wiki/git för mallar + checklistor.
- *Versioner:* märk med datum/version och ägare.
- *“Definition of done”*: vad ska vara sant när du är klar (mät/validera).

Exempel: *Checklistan “Ny accesswitch” inkluderar: namn, mgmt-IP, NTP/syslog, SSH/AAA, portprofiler, trunk-VLAN, STP-root-prioritet, dokumentation, validering.*

92

3.3.9 Mätetal för driftkvalitet

Välj enkla **SLO:er** (service-nivåmål) som går att mäta: **uptime** per zon, **latens** mellan hus, **WLAN-airtime** vid toppar, **DHCP-svarstid**, **DNS-hit-rate**, antal **P1-incidenter** per termin och genomsnittlig **MTTR**. Visualisera i en dashboard så att trender upptäcks tidigt och att förbättringar kan följas upp med fakta – inte magkänsla.

Förtydligande (sammanfattning):

- *Få, relevanta tal:* hellre 5 som används än 30 som ignoreras.
- *Visualisera:* dashboard – grönt/gult/rött.
- *Lärdomar:* varje P1 → en förbättring i design, process eller dokumentation.

Exempel: *Efter återkommande P2 p.g.a. “saknar IP” införs krav: DHCP-relay + Snooping-kontroll i checklista och auto-test i övervakning.*

3.3.10 Ett mini-bibliotek (vad och var)

Ha en **enkelsida** som länkar till allt: topologi, IP/VLAN, portmatriser, policies, runbooks, mallar, ändringslogg och kontaktlista. Då vet alla **var** man börjar – även när det brinner. En “start-här”-yta minskar startsträcka för nya i teamet och sparar tid i akuta lägen.

En bra startsida är värd sin vikt i guld under incidenter.

93

📌 Kontrollfrågor

1. Nämn tre konkreta sätt som dokumentation sänker MTTR.
2. Vad ska minst ingå i en uppdaterad IP-/VLAN-plan och varför?
3. Ge ett exempel på en bra namngivningskonvention för nätverksenheter.
4. Hur definierar du prioritet (P1–P4) i ärendehantering och varför är det viktigt?
5. Vad bör en change-plan innehålla utöver själva åtgärden?
6. Beskriv en lager-för-lager-metod för att hitta rotorsaken till “ingen IP i sal 215”.
7. Vilka etiska regler bör gälla för paketfångst i skolmiljö och varför?
8. Varför bör admin-åtkomst gå via jump host/bastion i management-VLAN?
9. Ge tre mätetal (SLO:er) som ger tidig varning om kvalitetsproblem i nätet.
10. Vad ska finnas på en “start-här”-sida för nätverksdrift?

94

🔗 Fördjupningslänkar

- Google SRE – Incident Management & Postmortems: https://sre.google/sre-book/table-of-contents/
- NIST SP 800-61 r2 – Computer Security Incident Handling Guide: https://csrc.nist.gov/publications/detail/sp/800-61/rev-2/final
- Cisco – Network Troubleshooting Best Practices (översiktssidor): https://www.cisco.com/c/en/us/support/docs.html
- NetBox (IPAM/DCIM) – dokumentation & inventarie: https://docs.netbox.dev/
- diagrams.net (draw.io) – topologidiagram: https://www.diagrams.net/
- Grafana – dashboards & larm: https://grafana.com/docs/
- Wireshark – User Guide: https://www.wireshark.org/docs/wsug_html_chunked/
- The Practice of System and Network Administration (bokresurser): https://www.sage.org/ (resurser/vidare läsning)

✍️ Egna anteckningar

…

95

3.4 Introduktion till CLI (inkl. Cisco-kommandon)

3.4.1 Vad är CLI och varför använda det?

CLI (Command Line Interface) är textbaserad administration där du skriver kommandon istället för att klicka i ett GUI. För nätverk ger CLI **precision**, **hastighet** och **reproducerbarhet**: du ser exakt vad som konfigureras, kan scripta uppgifter, och får samma resultat varje gång. I felsökning är CLI överlägset eftersom du kommer åt detaljer (tabeller, räkneverk, debug) som ofta saknas i ett GUI.

CLI är också "språket" som förenar olika fabrikat: koncepten liknar varandra även om kommandona skiljer sig.

***Exempel:** Du hittar en duplex-mismatch genom att köra `show interfaces` och läsa felräknare; i GUI:n syntes bara "länk up".*

3.4.2 Åtkomst till nätutrustning: console, SSH och sessionshygien

Du når switchar/routrar via **console** (seriell kabel lokalt) eller **SSH** (krypterad fjärranslutning). **Telnet** undviks (okrypterat). I skolmiljö används ofta en terminalklient (t.ex. Windows Terminal, PuTTY) och säkra **jump host/bastion** för att nå management-nätet först.

Förtydligande:

- *Console:* räddar dig när nätet ligger nere; kräver fysisk närvaro.
- *SSH v2:* standard för fjärr; stäng av telnet (`transport input ssh`).
- *Sessionshygien:* sätt `exec-timeout`, `logging synchronous`, och jobba från management-VLAN.

***Exempel:** En felkonfigurerad trunk släpper inte in dig via nätet. Du kopplar konsolkabel, rättar VLAN-listan och återfår fjärråtkomst via SSH.*

3.4.3 Cisco IOS – lägen och hjälp

Cisco-CLI har tydliga **lägen**:

- **user EXEC** (`>`): läs-kommandon.
- **privileged EXEC** (`#`): `enable` för fler kommandon.
- **global config** (`(config)#`): `configure terminal`.
- **under-lägen:** t.ex. `interface`, `line`, `router`, `vlan`.

Navigering och hjälp:

- `?` visar tillgängliga kommandon i kontext.
- `tab` auto-kompletterar; delord + `?` visar alternativ.

96

- Prefixet **no** negaterar (tar bort) en inställning.
- `do <show-kommando>` funkar inne i config-läge.

Exempel: *Du skriver* `spanning-tree bpdug?` *och ser* `bpduguard` *som val – snabb påminnelse utan att lämna läget.*

3.4.4 Visa, spara och jämföra konfiguration

Det viktigaste "visa-kommandot" är `show running-config`. Spara ändringar med `copy running-config startup-config` (eller `write memory`). Använd filter för att hitta rätt rader snabbt.

Förtydligande:

- *Visa:* `show run | section interface`, `| include`, `| exclude`, `| begin`.
- *Spara:* `copy run start`.
- *Jämför/återställ:* `show archive` (om aktiverat), `reload in 5` som "fallskärm" inför riskfyllda ändringar.

Exempel: *Du sätter* `reload in 10` *innan du ändrar en fjärrtrunk. Om du tappar sessionen rullar enheten automatiskt tillbaka om du inte* `reload cancel`*:ar.*

3.4.5 "Show" du använder varje dag (snabbdiagnos)

De här kommandona ringar in fel på minuter:

Förtydligande:

- *Länk & portar:* `show interfaces status`, `show interfaces` (felräknare, duplex/hastighet), `show power inline` (PoE).
- *Lager 2:* `show vlan brief`, `show mac address-table`, `show spanning-tree`, `show storm-control`.
- *Grannar:* `show cdp neighbors detail` och/eller `show lldp neighbors detail`.
- *L3 & ARP:* `show ip interface brief`, `show ip route`, `show arp`.
- *WLAN/AP via controller:* kanaler, SNR, klienter (vendor-specifikt).

Exempel: *Klienten når inte DHCP.* `show vlan brief` *visar att porten ligger i VLAN 1 istället för VLAN 20. Justera portprofilen och testa igen.*

3.4.6 Grundläggande lager-2-konfiguration (switch)

Det här är kärnan för access- och uplink-portar.

Förtydligande:

97

- *Beskriv & profil:*
```
interface gi1/0/12
description Klassrum214-Skrivare
switchport mode access
switchport access vlan 20
spanning-tree portfast
spanning-tree bpduguard enable
```
- *Trunk:*
```
interface gi1/0/48
switchport mode trunk
switchport trunk allowed vlan 10,20,30,40
switchport trunk native vlan 999
```
(oanvänt)
- *VLAN:* `vlan 20` → `name ELEV`

Exempel: *En elev kopplar in en "smart-switch" som skickar BPDUs. Med* `bpduguard enable` *err-disablas porten direkt och nätet skyddas.*

3.4.7 Grundläggande lager-3-konfiguration (SVI och routing)

På L3-switchar skapar du SVI:n som gateways och slår på routing. På routrar konfigurerar du adresser per gränssnitt.

Förtydligande:

- *SVI & routing:*
```
interface vlan 20
ip address 10.20.0.1 255.255.252.0
no shutdown
ip routing
```
- *Statik:* `ip route 0.0.0.0 0.0.0.0 10.0.0.1`
- *OSPF exempel:*
```
router ospf 1
network 10.20.0.0 0.0.3.255 area 0
```

Exempel: *Efter att ha lagt till SVI för VLAN 40 kan elever nå servrar först när* `ip routing` *aktiveras. Utan den stannar trafiken på L2.*

3.4.8 Härda åtkomst och management

Säkra accessen innan du släpper lös nätet.

Förtydligande:

- *SSH v2 & användare:*
```
ip domain-name skolan.local
```


98

```
crypto key generate rsa modulus 2048
username admin secret <långt-lösenord>
line vty 0 4 → transport input ssh → login local → exec-timeout 10
```

- *Banners:* `banner login` för juridisk varning.
- *Syslog/NTP:* `service timestamps log datetime msec` + skicka loggar; ställ NTP.
- *AAA/RADIUS (översikt):* central auth (beskrivs mer i senare kapitel).

Exempel: *Efter att ha stängt Telnet med* `transport input ssh` *och lagt* `login local` *syns inloggningar i syslog med korrekta tidsstämplar.*

3.4.9 Effektivitet i CLI (små knep som sparar timmar)

Små vanor gör stor skillnad.

Förtydligande:

- *Historik & redigering:* piltangenter, `ctrl-a/e` (början/slut), `ctrl-u` (rensa).
- *Rörledning:* `show run | section interface` → kopiera bara det du behöver.
- *Interface-range:* `interface range gi1/0/1-24` för massändringar.
- *Alias:* `alias exec svi show ip interface brief | include Vlan` (snabbvy).
- *"Do show" i config:* slipp hoppa mellan lägen.

Exempel: *Du behöver slå på* `portfast` *på 24 portar. Med* `interface range` *gör du det på 10 sekunder istället för 10 minuter.*

3.4.10 Vanliga fallgropar och hur du undviker dem

Många incidenter beror på små missar.

Förtydligande:

- *Glömd "write"*: ändringar överlever inte om du inte sparar.
- *Fel VLAN på accessport:* ser "länk up" men klienten hamnar i fel nät.
- *Ofullständig trunk:* VLAN saknas i `allowed vlan`.
- *Looprisk:* portfast saknas på klientportar → lång konvergens.
- *Telnet kvar:* okrypterade lösenord på nätet.

Exempel: *"Allt funkar – tills omstart." Klassikern:* `copy run start` *glömdes. Inför rutin: spara och verifiera varje change.*

99

100

📌 Kontrollfrågor

1. När är Console nödvändigt och när räcker SSH?
2. Vilka lägen finns i Cisco-CLI och vad gör `no`-prefixet?
3. Hur använder du `?`, `tab` och `do` för att arbeta snabbare och säkrare?
4. Vad gör `show interfaces` respektive `show vlan brief` och när använder du dem?
5. Ge tre steg för att felsöka en port som “ser upp” men där klienten inte får IP.
6. Skriv de fyra raderna som gör en port till säker accessport för klient: VLAN, portfast, bpduguard, description.
7. Hur aktiverar du L3-routing på en switch och konfigurerar en SVI-gateway?
8. Vilka inställningar krävs för att säkra fjärråtkomst (SSH v2, login, timeout, timestamps)?
9. När använder du `reload in` och varför är det smart vid fjärrändringar?
10. Nämn tre vanliga fallgropar i trunk-konfigurationer och hur du upptäcker dem.

101

🔗 Fördjupningslänkar

- Cisco – Introduction to the Cisco IOS CLI (guide): https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/fundamentals/configuration/15-s/fund-15-s-book/cf_cli.html
- Cisco – Basic Switch Configuration: https://www.cisco.com/c/en/us/support/docs/lan-switching/catalyst-2960-x-series-switches/118763-config-2960x-00.html
- Cisco – Using the `show` and filtering pipelines: https://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-operating-system-ios/7133-logging.html
- OpenSSH – Secure Shell best practices: https://www.openssh.com/manual.html
- Microsoft – Windows Terminal & OpenSSH Client: https://learn.microsoft.com/windows/terminal/
- man7 – Linux `ip` och `ss` referens (iproute2): https://man7.org/linux/man-pages/man8/ip.8.html
- Wireshark – User Guide (för djup felsökning): https://www.wireshark.org/docs/wsug_html_chunked/

102

✍️ Egna anteckningar

…

103

3.5 Windows Server – Domäner och AD

3.5.1 Överblick: vad är en domän och Active Directory (AD DS)?

En **domän** är en överenskommelse om identiteter och regler: vem du är, vad du får, och hur det bevisas på ett sätt som alla datorer i domänen litar på. **Active Directory Domain Services (AD DS)** är katalogen som håller ihop detta – ett "telefonregister med hjärna" som lagrar konton, grupper, datorer, skrivare och relationer mellan dem. När en användare loggar in händer tre saker i snabb följd: klienten **hittar** en domänkontrollant (via DNS), **bevisar** vem användaren är (Kerberos/NTLM), och **hämtar** regler och resurser som hör till identiteten (GPO, gruppmedlemskap, rättigheter). I större miljöer delas katalogen in i flera **domäner** under samma **skog** (forest), där skogens gemensamma språk – **schemat** – gör att alla förstår samma objekttyper. För en skola betyder detta att inloggning och datorupplevelse kan vara lika i alla salar, samtidigt som administration kan **delegeras** lokalt utan att ge bort "superkrafter".

Förtydligande (sammanfattning):

- *Domän:* en gemensam säkerhetsgräns och policyyta.
- *Skog:* en eller flera domäner som delar schema och kan lita på varandra.
- *Domänkontrollant (DC):* server som autentiserar och replikerar katalogen.

Exempel: *En elev får samma skrivare och samma hemkatalog oavsett dator. Det är inte "turen" som avgör, utan domänens identitets- och policylogik.*

3.5.2 Grundkomponenter: DC, FSMO-roller, tid och platser (Sites)

En domän behöver **flera domänkontrollanter** av samma skäl som en stad behöver fler än ett sjukhus: driftstopp ska inte bli en kris. Vissa uppgifter är "ensamrättsroller" – **FSMO** – för att undvika kollisioner (t.ex. schemaändringar, tilldelning av SID-delar). Den mest synliga i vardagen är **PDC Emulatorn**, som fungerar som tidsreferens och där många lösenordsrelaterade uppgifter prioriteras. **Tid** låter trivialt, men i Kerberos är den en säkerhetsparameter: ligger klockorna fel upphör biljetterna att gälla. I ett nät som täcker flera byggnader hjälper **Sites and Services** AD att förstå geografin: klienter i Hus A ska i första hand använda "sin" DC, och replikering mellan hus kan ske med rimliga intervall.

Förtydligande (sammanfattning):

- *Minst två DC:* för robusthet och underhåll.
- *FSMO:* särskilda ansvar som inte mår bra av att vara "flera samtidigt".
- *Tidssynk:* en kedja från PDC Emulator → övriga DC → klienter.
- *Sites:* kopplar subnät till "närmaste" DC och styr replikering.

Exempel: *När en ny elev byter lösenord på morgonen ska samma lösenord gälla i alla hus efter minuter, inte timmar. Det är replikeringstopologin som gör det möjligt.*

104

3.5.3 DNS-integrering – AD:s nervsystem

AD lutar sig mot **DNS** i nästan varje steg. När en klient vill logga in måste den hitta en DC genom att slå upp särskilda **SRV-poster** i den interna zonen. Därför lagras zonen ofta **AD-integrerat**: den följer med katalogens replikering och accepterar **säkra, dynamiska** uppdateringar från domänanslutna maskiner. Det gör att en dator automatiskt kan registrera sig med rätt namn och att klienter spontant hittar rätt väg även när adresser ändras. Fel i den här kopplingen maskerar sig gärna som "AD-problem", men roten är ofta att någon dator råkar fråga **fel DNS** (t.ex. en publik resolver som inte känner till interna SRV-poster).

Förtydligande (sammanfattning):

- *SRV-poster:* "här finns LDAP/Kerberos för den här domänen".
- *AD-integrerad zon:* säker dynamik + samma replikering som katalogen.
- *Klientupplevelse:* rätt DNS på klienten är en förutsättning för "AD funkar".

***Exempel:** En dator kan surfa men "hittar inte domänen". Förklaringen är ofta att DNS pekar på internet istället för skolans egna resolvers – då syns aldrig AD:s SRV-poster.*

3.5.4 OU-design och delegation

OU:er (Organizational Units) är katalogens ordning och reda. De fungerar både som **mappar** för att gruppera liknande objekt och som **gränser** där man kan fästa policyer och ansvar. En bra design speglar **driften**: datorer för elever, personal och servrar i skilda grenar; underindelning per hus eller sal när det hjälper arbetet; och tydliga platser där **delegering** sker utan att släppa in någon på fel nivå. Principen **LSDOU** (Local → Site → Domain → OU) berättar i vilken ordning policyer tolkas; den sista som passar vinner – med undantag för några specialflaggor. Poängen är att kunna uttrycka "hur ska det vara här?" utan att behöva uppfinna regler på varje enskild dator.

Förtydligande (sammanfattning):

- *Struktur efter drift:* vem äger vad, vem behöver ändra vad, var gäller vilka regler?
- *Placera objekt rätt:* lämna standardcontainrar, lägg i OU.
- *Delegation:* ge "lagom" makt nära där jobbet sker.

***Exempel:** En datasal kan få hårdare skrivbordspolicyer än lärarrummet, även om samma elev loggar in på båda. Det avgörs av datorns OU och hur policyerna är kopplade.*

3.5.5 GPO – hur policyn "blir verklighet"

Group Policy är verktygslådan där **datorinställningar** och **användarinställningar** bor. När en användare loggar in "läggs" deras egna regler ovanpå datorns regler enligt **LSDOU**. Ibland vill man tvärtom: att **datorns plats** bestämmer även användarupplevelsen – då används **loopback processing** (t.ex. i datasalar). **Security Filtering** och ibland **WMI-filter** gör att en policy träffar rätt grupp eller rätt sorts dator, och **SYSVOL** bär runt själva innehållet så alla DC:er kan dela på jobbet.

105

Allt detta är osynligt när det fungerar, men tydligt när det inte gör det: GPO är den tysta kraften bakom “samma upplevelse varje dag”.

Förtydligande (sammanfattning):

- *Två sidor:* dator- vs användarpolicys.
- *Ordning:* Local → Site → Domain → OU (senast relevant “vinner”).
- *Riktning:* filtrera på grupp eller egenskap; loopback när rummet ska styra.

***Exempel:** Om alla i datasalen ska se samma startmeny oavsett vem de är, låter man datorernas OU tala högst via loopback. Då följer upplevelsen rummet, inte personen.*

3.5.6 Konton, grupper och rättigheter (AGDLP, UPN, lösenpolicy, gMSA)

Identitet i AD blir begriplig först när **konton** och **grupper** hålls isär i tanken. Kontot beskriver *en person eller en tjänst*; gruppen beskriver *en roll*. Med **UPN**-formatet (namn@domän) känns inloggning igen från andra tjänster. **AGDLP** är bara ett sätt att säga att vi **först** samlar människor i passande **globala grupper**, **sedan** kopplar rollerna till **resurser** via **Domain Local-grupper** som bär behörigheten. För tjänster är **gMSA** en tryggare variant än lokala konton med “hemliga” lösenord – domänen sköter rotationen. Tillsammans ger detta ett språk där förändringar i verkligheten (en elev byter klass, en lärare byter roll) speglas av **gruppmedlemskap**, inte av specialfall på resurserna.

Förtydligande (sammanfattning):

- *Kontot = individ/tjänst, gruppen = roll.*
- *AGDLP:* skär isär “vem” och “var” så att behörigheter blir underhållbara.
- *gMSA:* minskar riskerna med tjänstkonton och stillastående lösenord.

***Exempel:** När en elev byter program byts inte “filservern” – bara gruppen hen tillhör. Därför följer rättigheter och skrivare med på köpet.*

3.5.7 Användarrättigheter i OS vs rättigheter till filer och mappar

Det är lätt att blanda ihop två olika sorters rättigheter: **User Rights Assignment** i operativsystemet handlar om *vad ett konto får göra mot datorn* (logga in, köra som tjänst, ta ägarskap), medan **NTFS- och share-rättigheter** handlar om *vad kontot får göra mot data*. Den första påverkar inloggning och tjänster, den andra påverkar mappar och filer. I praktiken leder tydlig **gruppbaserad** hantering till färre överraskningar: datorn vet vilka som får logga in; filservern vet vilka som får läsa och skriva – och de två världarna korsar bara varandra där de måste.

Förtydligande (sammanfattning):

- *User Rights:* OS-beteenden (t.ex. “Log on as a service”).
- *NTFS/Share:* dataåtkomst; effektiv rätt = snittet av båda.
- *Gruppbaserat:* stabilare än individuella undantag.

106

Exempel: *En tjänst vägrar starta “trots admin”. Felet är ofta att kontot saknar själva* ***rätten*** *att köras som tjänst, vilket inte är samma sak som filrättigheter.*

3.5.8 DHCP i domänmiljö (autorisering, redundans, namn)

DHCP i en domän är inte bara “adressutdelning”, utan också **tillit** och **namn**. Servern behöver vara **auktoriserad** i katalogen för att få dela ut adresser – ett skydd mot oväntade källor. Två servrar kan arbeta tillsammans så att utdelningen fortsätter även under patchar och fel. Och eftersom namnet betyder mycket i en AD-miljö är det vanligt att låta DHCP hjälpa till med **dynamiska DNS-uppdateringar**, så att “vem bor var?”-frågan alltid har ett rimligt svar. Allt detta är små detaljer var för sig, men tillsammans gör de att klassrum “bara fungerar” när många maskiner kommer och går varje dag.

Förtydligande (sammanfattning):

- *Autorisering:* domänen godkänner DHCP som “talar för den”.
- *Redundans:* två servrar delar ansvar.
- *DNS-uppdatering:* adresser får namn och namn får adresser utan handpåläggning.

Exempel: *I elevnätet byts adresser ofta. När DHCP och DNS pratar med varandra slipper du spökinlägg – namnen pekar dit datorerna faktiskt finns just nu.*

3.5.9 Fildelning – struktur, NTFS vs share, ABE, VSS och GPP

En filserver är inte bara en hårddisk på nätet; den är en **scenografi** för hur människor arbetar. Genom att skilja på **share-rättigheter** (grov grind vid ingången) och **NTFS-rättigheter** (finmaskig styrning inne i huset) går det att uttrycka “alla som kommer in i den här korridoren får läsa – men bara vissa får öppna klassrumsdörrarna”. **Access-Based Enumeration (ABE)** döljer dörrar du ändå inte får öppna, vilket gör upplevelsen lugnare och säkrare. **Skuggkopior (VSS)** låter användare backa tiden själva när någonting råkade bli fel, och **Group Policy Preferences** gör att rätt diskar dyker upp där de behövs, kopplade till roller istället för datorbilder. Med den här kompositionen blir filservern begriplig: du ser det du ska se, och det du gör går att ångra.

Förtydligande (sammanfattning):

- *Share = grov nivå, NTFS = fin nivå.*
- *ABE minskar “kan inte öppna”-support.*
- *VSS och GPP ger mindre friktion i vardagen.*

Exempel: *En elev ser bara sin klassmapp och sin hemkatalog. Läraren ser dessutom ämnesmappar. Blir något fel i gårdagens dokument finns “Tidigare versioner” ett högerklick bort.*

3.5.10 Standardmönster för behörighet: AGDLP i praktiken

Det är viktigt att tänka på att **AGDLP** används konsekvent eftersom det separerar *vem* som behöver åtkomst från *var* åtkomsten faktiskt ges. I praktiken innebär det att individuella **konton** samlas i

107

globala grupper (per roll, klass eller funktion) för att visa *vilka* personer som hör ihop. Dessa globala grupper läggs i sin tur in i **Domain Local**-grupper som är knutna till *resurserna* (t.ex. en fildelning eller en applikation) och som **bär själva behörigheten**. Poängen är att förändringar i personal, klasser eller roller kan göras genom att bara flytta medlemmar mellan **globala grupper** – utan att behöva röra de känsliga behörighetslistorna på resurserna. På det sättet minskar risken för misstag, och helpdesk kan lösa de flesta åtkomstärenden snabbt och spårbart.

Namnstandarden blir också en del av läsbarheten. Om **Domain Local**-gruppen berättar *vilken resurs och nivå* den representerar (t.ex. `DL_Share_Personal_Modify`) och den **globala** gruppen berättar *vilken roll* den samlar (t.ex. `G_Larare` eller `G_23TE`) blir behörighetstabellen nästan självdokumenterande. Att dessutom ange en **ägare** för varje DL-grupp gör att det alltid är tydligt vem som får besluta om medlemskap – något som är viktigt i en skolmiljö där klasser byts och personal tillkommer över tid. Om miljön växer till flera domäner i samma skog kan **Universal**-grupper användas som ett mellansteg, men grundidén är densamma: separera människorna från resurserna och låt grupperna vara "kopplingsdosor" mellan dem.

Exempel: *När en ny lärare börjar läggs kontot i* `G_Larare`*. Den gruppen är redan medlem i* `DL_Share_Personal_Modify`*, som i sin tur har NTFS-rättigheten* ***Modify*** *på* `\\srv-fil-01\Personal`*. Läraren får åtkomst utan att någon behöver öppna NTFS-dialogen — och när läraren slutar räcker det att ta bort kontot ur* `G_Larare`*.*

3.5.11 Felsökning: var börjar jag, och varför just där?

När något "AD-relaterat" strular är **namn** och **tid** de två första frågorna. Utan rätt DNS hittar klienten inte sin DC, och utan rimlig tid accepterar Kerberos inte biljetten. Därifrån följer två spår: **replikering** (delar DC:erna samma bild av världen?) och **policytillämpning** (träffar policyn rätt objekt, på rätt plats?). Genom att hålla sig till den ordningen blir felsökningen mindre magi och mer metod: vi slår fast "vad fungerar", ringar in "var slutar det fungera", och låter detaljerna leda oss vidare.

Förtydligande (sammanfattning):

- *DNS först, tid sen:* de är fundamenten.
- *Replikering:* en sanning i katalogen, inte många halvsanningar.
- *Policy:* rätt plats, rätt riktning, rätt villkor – annars händer "inget".

Exempel: *En dator i datasalen får inte "datasalsupplevelsen". Felet visar sig vara att den flyttats till fel OU – policyn missar helt enkelt målet.*

3.5.12 Rekommenderade bas-GPO:er (varför just de?)

Det är bra att etablera ett litet **grundgolv av policyer** som alltid finns, eftersom det ger förutsägbarhet i undervisningen och minskar antalet incidenter. **Windows Update** med en enkel "ring"-modell (test → pilot → brett) hjälper till att få ut säkerhetsfixar utan att riskera att en hel datasal tappar funktion samma dag; några maskiner får först, resten när det visat sig stabilt.

108

Windows-brandväggen med korrekta profiler (domän/privat/offentlig) säkerställer att samma dator beter sig olika i skolnät och hemma — öppet där det måste vara öppet, stängt i övrigt. **Tid/NTP** styrs centralt för att Kerberos ska fungera; en liten tidsdrift räcker för att inloggningar och biljetter ska börja fallera.

Windows LAPS motiveras av att lokala administratörslösenord inte ska återanvändas mellan datorer; varje maskin får ett unikt, roterande lösenord som går att läsa vid behov, vilket minskar risken för lateral förflyttning. Att **stänga Telnet och andra legacy-protokoll** handlar om att ta bort onödiga angreppsytor – moderna, krypterade protokoll räcker. **RDP** bör vara tillåtet **endast** från management-nät eller via bastion, för att inte exponera administrativa gränssnitt i elev- och personalnät. **Audit-policyer** (inloggningar, policyändringar, åtkomstförsök) gör det möjligt att efterhand förstå "vad som hände" när något går fel eller när en incident måste följas upp.

Till det kommer **Group Policy Preferences** för skrivare och enhetskopplingar (P:, S: osv.). Poängen är att kopplingar då följer **grupper** snarare än enskilda datorbilder – när en elev byter klass följer skrivare och mappar med utan manuell handpåläggning. En **standardiserad startmeny** i datasalar gör att lektioner börjar likadant oavsett vilken dator man sätter sig vid; både elever och lärare sparar tid när viktiga program ligger på samma plats. Slutligen är det klokt att **versionera** policyer och **dokumentera** var de är länkade, för att korta vägen från "något ändrades" till "nu vet vi vad".

Exempel: *En ny datasal får samma upplevelse som de andra eftersom "Datasal-Golv" redan innehåller brandväggsprofiler, LAPS, RDP-begränsning, audit och standardmeny. Skrivare och nätverksenheter kopplas in via GPP baserat på grupper, så lärarna slipper jaga genvägar.*

109

📌 Kontrollfrågor

1. Vad betyder det att en domän är en *säkerhetsgräns*, och hur märks det i vardagen?
2. Varför är **tid** en säkerhetsfråga i en AD-miljö, och vilken roll spelar PDC Emulatorn?
3. Hur hänger **DNS** ihop med inloggningen, och vilka symptom får du när klienten frågar “fel” resolver?
4. På vilket sätt hjälper **OU:er** både policys och delegation, och varför bör standardcontainrar undvikas?
5. Förklara **LSDOU** och ge ett exempel där **loopback** är nyckeln till rätt upplevelse.
6. Vad vinner du på att tänka **konton = individer/tjänster** och **grupper = roller**?
7. Varför är **AGDLP** lättare att underhålla än individuella rättigheter direkt på resurser?
8. Vad skiljer **User Rights** i OS från **NTFS-/share-rättigheter**, och varför är den skillnaden viktig?
9. Hur bidrar **DHCP-autorisation** och **dynamiska DNS-uppdateringar** till att domäninloggning “bara funkar”?
10. Vad gör **ABE**, **VSS** och **GPP** för skillnad i en filservermiljö för skola?

110

🔗 Fördjupningslänkar

- Microsoft Learn – Active Directory Domain Services (översikt): https://learn.microsoft.com/windows-server/identity/ad-ds/get-started/active-directory-domain-services-overview
- DNS i AD (SRV, AD-integrerade zoner, dynamiska uppdateringar): https://learn.microsoft.com/windows-server/identity/ad-ds/plan/active-directory-integrated-dns
- Group Policy – grunder, loopback, filtrering: https://learn.microsoft.com/windows/security/threat-protection/security-policy-settings/
- DHCP Server – auktorisering, failover, DDNS: https://learn.microsoft.com/windows-server/networking/technologies/dhcp/dhcp-top
- File and Storage Services – NTFS, SMB, ABE, FSRM, VSS: https://learn.microsoft.com/windows-server/storage/file-server/file-server
- Windows LAPS: https://learn.microsoft.com/windows-server/identity/laps/laps-overview
- AD-replikering och DC-hälsa (repadmin, dcdiag): https://learn.microsoft.com/troubleshoot/windows-server/identity/active-directory-replication-troubleshooting

111

✍️ Egna anteckningar

...

112

3.6 Ubuntu Server och Linux i nätverk

3.6.1 Grundläggande nätverkskonfiguration i Linux

På Linux finns nätverket i flera “lager” samtidigt: **kärnan** håller reda på interface, adresser och tabeller; **användarverktyg** (som `ip`) pratar med kärnan i realtid; **tjänster** (som DNS-resolvern och tidssynk) påverkar hur namn och klockor upplevs; och **beständig konfiguration** (på Ubuntu via **Netplan**) beskriver hur allt ska se ut efter en omstart. Det låter abstrakt, men poängen är enkel: man skiljer på *tillstånd just nu* och *tillstånd över tid*. I undervisning och drift blir det därför naturligt att “läsa” nätet med verktyg som visar nuläget, samtidigt som man förstår var den beständiga sanningen faktiskt bor.

På **Ubuntu Server** beskriver **Netplan** nätet i YAML-filer (vanligen `/etc/netplan/*.yaml`) och låter sedan en “motor” – ofta **systemd-networkd** på servrar – göra jobbet vid boot. Det gör att sådant som **statisk IP**, **gateway**, **namnservrar**, **VLAN**, **bonding** och **bryggor** (bridges) lever i samma modell, och att alla delar kan uttryckas lika tydligt för IPv4 och IPv6. Nuläget speglas av moderna kommandon: `ip address` och `ip route` visar adresser och rutter, **`resolvectl`** visar hur namn faktiskt slås upp, och **`networkctl`** ger en översikt över kända länkar och deras status. **VLAN** på Linux är inte magiskt: ett taggat underinterface (t.ex. `enp3s0.20`) *är* bara samma fysiska port med en tydlig etikett för nätet; **bonding** binder flera fysiska portar till en logisk (för redundans/kapacitet), och **bryggor** kopplar virtuella gäster (container/VM) till “riktiga” nät. I praktiken liknar det hur du tänker i en switch: *namn* på saker betyder mindre än *hur de kopplas*.

DNS-funktionen på moderna Ubuntu sköts ofta av **systemd-resolved**, som både cache:ar och vet vilken namnserver som gäller på vilket interface. Det är här **split-horizon** och “rätt resolver på rätt plats” blir konkret: interna namn ska fråga interna resolvers, och externa namn får gå utåt. Samma logik gäller **tidssynk** – **systemd-timesyncd** eller NTP-klient – där en liten avvikelse får stora följder om servern ingår i en domän som använder **Kerberos**. I servermiljö är det dessutom vanligt att separera **management-trafik** (administration) från **tjänstetrafik** (det som servrarna levererar) till olika VLAN och att styra åtkomst med brandvägg (Ubuntu har **UFW** som ett vänligt gränssnitt till underliggande paketfilter). Det gör inte servern “paranoid”; det gör den **förutsägbar**.

Förtydligande (sammanfattning):

- *Två perspektiv:* **nuläge** (kärna + verktyg) och **beständighet** (Netplan).
- *Byggstenar:* VLAN för etiketter, **bond** för redundans/kapacitet, **bridge** för att koppla gäster till rätt nät.
- *Tjänster:* **resolved** för namn, **timesyncd/NTP** för tid – båda styr “allt annat”.
- *Säker drift:* separera **management** från **tjänstetrafik**, låt brandvägg och riktade regler göra sitt.

Exempel: *En Ubuntu-filserver svarar långsamt på domäninloggningar men “internet” är snabbt. Förklaringen är sällan CPU — det är oftare att servern råkar fråga* ***fel DNS*** *(extern resolver utan*

*SRV-poster) eller har en liten **tidsdrift**. När namn och tid blir rätt, blir allt annat "magiskt" rätt också.*

3.6.2 SFTP, SMB, SSH, SCP

I Linuxvärlden gör man en poäng av att **välja rätt verktyg för rätt typ av överföring och åtkomst**. **SSH** är grunden: ett säkert skal som ger fjärrterminal, **tunnel** (port forward) och **subsystem** som **SFTP**. Det viktiga för elever att förstå är att **SFTP inte är "FTP med S"**, utan en logisk filkanal *inuti* SSH – vilket betyder att *exakt samma identitet och kryptering* som skyddar terminalen också skyddar filen. **SCP** är historiskt det snabbaste sättet att "bara få över en fil", men i dag föredrar många **SFTP** (för robusthet och interoperabilitet) eller **rsync över SSH** (för diff/återupptagning och rättigheter). Poängen är inte att minnas alla flaggor, utan att kunna resonera: *är detta en engångskopia, en synk som ska köras nattligen, eller ett incheckat arbetsflöde?*

SMB (Samba) adresserar en annan situation: **samarbete med Windowsvärlden**. När en Linuxserver ska dela mappar till Windows-klienter är Samba språktolken som gör att **NTFS-liknande rättigheter** (inklusive **ACL**) och **Windows-inloggning** känns hemma. I en skolmiljö är det vanligt att låta Samba **läsa identiteter** från en domän – oavsett om servern formellt "går med i" domänen eller bara konsumerar gruppinformation. Det som ser enkelt ut ("mappar till elever och lärare") blir stabilt först när identiteter, grupper och rättigheter **hänger ihop** över protokollgränsen. Därför är det bra att elever kopplar **SMB-upplevelsen** till **gruppbaserad behörighet** (se 3.5): på Linuxsidan **ägare/grupper/ACL**, på Windows-sidan **roller** – och Samba översätter.

Tillbaka till **SSH**: utöver terminal och SFTP är SSH också ett **transportlager** för andra behov. Med en **local forward** (t.ex. `localhost:8443` → "serverns 443") går det att administrera webbtjänster utan att öppna dem externt, och med **ProxyJump** blir en **jump host/bastion** första halt på varje resa – vilket ger färre öppna portar och ett tydligt spår av vem som gjorde vad. Nyckelbaserad inloggning är inte bara "bekvämt": den *byta ut* det som annars vore ett lösenord på nätet mot **kryptografiska bevis**. I skolmiljö försvinner mycket friktion när man *förklarar varför* det är värt att hålla ordning på **nycklar** (och deras passfraser), snarare än att lägga energi på att komma ihåg "det senaste superlösenordet".

Förtydligande (sammanfattning):

- *SSH:* fjärrskal, tunnlar, **SFTP-subsystem**; **nycklar** ger både säkerhet och smidighet.
- *SFTP/SCP/rsync:* tänk **fall** – engång vs synk vs rättighetsbevarande.
- *SMB (Samba):* bro mellan Linuxfilsystem och Windowsklienter; identitet + ACL gör att allt "känns likadant" för användaren.
- *Bastion:* en obligatorisk mellanlandning gör adminspår tydligt och ytan mindre.

***Exempel:** En ämnesansvarig lärare vill lägga upp stora kursfiler på filservern från hemmet. Med **rsync över SSH** kopieras bara det som ändrats, rättigheterna bevaras, och överföringen kan återupptas. När läraren är på skolan monteras samma mapp som **SMB** på lärardatorn – två vägar, samma källa, ingen dubbeladministration.*

114

3.6.3 Användarskapande och rättigheter via CLI

Linux håller identitet enkelt och synligt: **användare** har **UID**, **grupper** har **GID**, och filsystemet bryr sig om **ägare**, **grupp** och "alla andra" när det avgör vad som är tillåtet. I praktiken betyder det att **ägarskap** och **gruppmedlemskap** är den vardagliga valutan – och att **rättigheter** (r, w, x) läses lite olika för filer och kataloger. Elever som snabbt lär sig "**x på en katalog betyder 'få gå igenom'**" får *plötsligt* superkrafter i felsökning: de förstår varför en mapp "som alla har rätt till" ändå inte går att nå (för att en föräldramapp saknar x), och de ser varför en **sticky bit** på en gemensam "inlämningskorg" gör att alla kan lägga till, men bara ägaren kan ta bort.

I en skolmiljö är det dessutom värt att se **grupp** som "rollen" i vardagen. En **lärargrupp** och en **klassgrupp** går att använda både i **Linux-ACL** (för finmaskiga rättigheter och **default-ACL** i träd) och i **Samba** (för att matcha Windowsvärldens förväntningar). Samtidigt håller **sudo** ordning på *vem* som får eskalera till administratör – inte för att försvåra, utan för att ge ett **spårbart ja** till en begränsad uppgift i stället för ett tyst "allt". När eleverna ser hur **ägarskap → grupp → ACL** hänger ihop blir filservern *begriplig* och uppgiften "ge *bara* lärare rätt att ändra i ämnesmappen" slutar vara ett mysterium.

Förtydligande (sammanfattning):

- *Identitet:* **UID/GID** och **primär/sekundär grupp** driver mycket av vardagen.
- *Rättigheter:* **r/w/x** tolkas olika för fil vs katalog; **sticky**, **setgid** på kataloger ger förutsägbarhet i delade ytor.
- *ACL:* kompletterar klassiska rättigheter; **default-ACL** är "ärva framåt" i nyckelkataloger.
- *Säker admin:* **sudo** ger smala, spårbara rättigheter; **grupper** representerar roller och mappar snyggt till Samba/SMB.

Exempel: *En gemensam "inlämna-hit"-mapp får* ***skrivrätt*** *för elever och* ***sticky bit*** *så att ingen kan ta bort någon annans filer. Ämnesmappen under samma träd har* ***setgid*** *på katalogen så att nya filer alltid får gruppen "Larare", och en* ***default-ACL*** *som ger lärargruppen* ***write*** *utan att någon behöver "komma ihåg" att ändra.*

115

📌 Kontrollfrågor

1. Varför skiljer Linux mellan **nuläge** (kärnans bild) och **beständig konfiguration** (Netplan), och hur märks det i felsökning?
2. Hur relaterar **VLAN**, **bonding** och **bridges** till vardagsproblemet “koppla rätt gäster till rätt nät”?
3. Varför påverkar **DNS-resolvern** och **tidssynk** ofta “allt annat”, och vilka symptom brukar du se när de är fel?
4. Förklara varför **SFTP** inte är “FTP + kryptering” utan ett **SSH-subsystem**, och vad det betyder för identitet och säkerhet.
5. När är **SCP** tillräckligt och när bör du hellre välja **rsync över SSH**?
6. Vilken roll spelar **Samba** i en blandad miljö, och hur kopplar den samman **Linux-rättigheter** med **Windows-roller**?
7. Vad betyder **x på katalog** jämfört med **x på fil**, och varför stoppar avsaknad av x ibland åtkomst trots “read”?
8. Hur hjälper **setgid** på en katalog och **default-ACL** till att hålla en delad ämnesmapp konsekvent?
9. Varför är **sudo** att föredra framför “logga in som root”, och vad vinner du på gruppbaserad delegering?
10. Beskriv ett scenario där en server upplevs “seg mot domänen” fast internet är snabbt — vilka två saker kontrollerar du först, och varför?

116

🔗 Fördjupningslänkar

- **Ubuntu Server Guide** (nätverk, lagring, tjänster): https://ubuntu.com/server/docs
- **Netplan** – översikt och exempel: https://netplan.io/
- **systemd-networkd / networkctl** – nät på systemd: https://www.freedesktop.org/software/systemd/man/latest/systemd.network.html
- **systemd-resolved / resolvectl** – DNS i systemd: https://www.freedesktop.org/software/systemd/man/latest/systemd-resolved.service.html
- **OpenSSH** – klient/server och nycklar: https://www.openssh.com/manual.html
- **Samba** – dokumentation och wiki: https://www.samba.org/samba/docs/
- **rsync** – manual och best practices: https://download.samba.org/pub/rsync/rsync.html
- **POSIX ACL** – `getfacl`/`setfacl`: https://man7.org/linux/man-pages/man5/acl.5.html
- **Filrättigheter och `chmod`**: https://man7.org/linux/man-pages/man1/chmod.1.html
- **Användare och grupper (`useradd`, `groupadd`)**: https://man7.org/linux/man-pages/man8/useradd.8.html

117

✍️ Egna anteckningar

...

118

3.7 MDM och klienthantering

3.7.1 Vad är MDM – och varför spelar det roll?

MDM (Mobile/Modern Device Management) är idén att styra **klienternas beteende och säkerhet centralt**, oavsett var de befinner sig. Där **AD + GPO** utgår från att datorn befinner sig "på plats" i domänen, utgår MDM från att enheten kan stå **var som helst** – hemma, i klassrummet, på ett café – och ändå få rätt policy, rätt appar och rätt nätprofiler över internet. Tekniskt sker detta via en **push-kanal** (Apple/Google/Microsofts notistjänster) som väcker klienten och säger "hämta din nästa policy", medan en **MDM-agent** eller inbyggt stöd i OS ser till att inställningarna faktiskt sätts.

Det viktiga för administratörer att förstå är **skillnaden i filosofi**: MDM modellerar datorn som ett **hanterat objekt** med mätbara tillstånd (kryptering? brandvägg? låsskärm?) och **efterlevnad** (compliance). Uppfyller enheten kraven får den tillgång (t.ex. till e-post eller lärarportaler); annars blockeras eller begränsas den tills kraven är uppfyllda. På så sätt blir **säkerhet en egenskap av klientens tillstånd**, inte bara av vilket nät man sitter på.

Förtydligande (sammanfattning):

- *MDM → internetförst:* funkar oavsett plats och VLAN.
- *Policy som tillstånd:* "är BitLocker/FileVault på?", "är OS uppdaterat?".
- *Efterlevnad → åtkomst:* klarar du kraven får du tjänsten (annars inte).

***Exempel:** En lärardator som är helkrypterad, har skärmlås och aktuell patchnivå får automatiskt full åtkomst till skolans e-post och molnlagring. En dator som halkat efter får bara läsa kursplaner tills den har uppdaterats.*

3.7.2 Ekosystemet: Intune, Jamf – och öppna alternativ

I Windowsvärlden är **Microsoft Intune** den vanligaste MDM-plattformen. Den talar alla de språk Windows behöver (Windows-MDM-CSP, Autopilot, Entra-ID-integration) och hanterar även **iOS/iPadOS/macOS** och **Android** i samma verktyg, vilket är praktiskt i skola. På Apple-sidan är **Jamf** branschstandard för djup styrning av **macOS** och iOS/iPadOS – särskilt när man vill in i Apples "alla detaljer" (Configuration Profiles, appdistribution, "supervised" läge, kontrollerad återställning). För Android är **Android Enterprise** själva modellen: "Work profile" för BYOD och "Fully managed" för skolägda enheter, ofta orkestrerat via Intune eller en Android-fokuserad MDM.

Det finns **öppna och öppet licensierade** alternativ, men de är oftast mer **plattformsspecifika** och kräver mer **hands-on**: t.ex. **MicroMDM** för macOS (ren Apple-MDM), eller Android-fokuserade community-varianter. De kan vara utmärkta i labb eller nischade behov, men ersätter sällan en **heterogen** skolmiljös krav på rapporter, efterlevnad och integrationer (identitet, appar, certifikat).

Förtydligande (sammanfattning):

- *Intune:* bred täckning (Win/macOS/iOS/Android), bra på Windows-livscykel.

119

- *Jamf:* djup Apple-styrning och Apple-flöden (ADE, VPP).
- *Android Enterprise:* Work profile (BYOD), Fully managed (skolägda).
- *Öppna projekt:* lärorika och vassa, men oftare smalare.

Exempel: *En kommun kör Intune för Windows-parken och låter Jamf sköta Mac-datorer i estetiska programmet. Båda plattformarna rapporterar efterlevnad till samma identitetstjänst, så åtkomstregler blir konsekventa.*

3.7.3 Centralt styrda policyer och säkerhetsinställningar

MDM uttrycker säkerhet som **mätbara regler**. Istället för "har vi rullat ut brandväggen?" frågar du "**är brandväggen på och blockerar den inkommande trafik?**". Därifrån följer ett antal byggstenar som återkommer oavsett OS:

- **Enhetskonfiguration:** skärmlås, lösenordspolicy, brandvägg, kryptering (BitLocker/FileVault), USB-policy, WLAN-profiler (SSID, 802.1X-certifikat), VPN-profiler, webbfiltret.
- **Efterlevnad (compliance):** minsta OS-version, kryptering på, ingen jailbreak/root, aktiva skydd.
- **Apphantering:** paketera, distribuera, uppdatera och – vid behov – ta bort appar; i Applevärlden via **VPP** (licenser) och i Android via **Managed Google Play**.
- **Villkorad åtkomst (Conditional Access):** låt **identiteten** kontrollera **tjänsten** baserat på **enhetens status**.
- **Certifikatlivscykel:** utfärda, förnya och dra tillbaka klient-certifikat för 802.1X, VPN och appar.

Det här låter stort, men effekten i vardagen är just **enkelhet**: du behöver inte längre fråga "var står datorn?", utan "**uppfyller den kraven?**".

Exempel: *En elevdator ansluter till skolans **WLAN** utan att någon rör en knapp: MDM har redan lagt **WLAN-profilen** med 802.1X-certifikatet, och brandväggen vet att bara skolans resolvers får svara på DNS.*

3.7.4 Provisionering och livscykel: från kartong till återbruk

Det moderna idealet är **zero-touch**: att en ny enhet kan öppnas av användaren och "hitta hem" utan att IT-personal rör den. För Windows kallas detta **Autopilot**, för Apple **Automated Device Enrollment (ADE)** via Apple School/Business Manager, och för Android **Zero-touch** (eller QR/NFC-inrullning). Poängen är inte magi, utan **kedjan**: enheten känner igen sig som skolägd, ansluter till **MDM**, får **policyer, appar, profiler** – och presenterar login på rätt sätt. Lika viktigt är **offboarding**: att kunna **återtaga licenser**, **rensa data** (helt eller selektivt vid BYOD), och **återanvända** enheten till nästa elev.

Förtydligande (sammanfattning):

120

- *Ägda enheter:* full styrning, full wipe; kan låsas till skolans ägarskap.
- *BYOD:* **selektiv rensning** (ta bort skoldata, lämna privatdata).
- *Livscykel:* inrullning → drift/uppdatering → support → återlämning/återbruk.

Exempel: *Årskurs 1 får nya laptops. Autopilot visar skolans välkomstbild, kopplar upp mot MDM, lägger in WLAN-profil och Office-appar – lektionen kan börja direkt utan imaging eller manuell patchning.*

3.7.5 Mobilitet, BYOD och externa enheter

BYOD (“ta med egen enhet”) och **externa minnen** (USB, mobil lagring) rör inte bara teknik, utan **juridik och förtroende**. Bra MDM-upplägg håller isär **personlig integritet** och **skolans data**. På Android innebär det **Work profile**: skolans appar och data bor i en skyddad “bubbla” som IT kan styra, medan privatdelen förblir privat. På iOS/iPadOS finns **User Enrollment** för liknande separation, och på macOS kan policyer riktas utan att ge IT “insyn” i allt. För USB är poängen inte förbud i sig, utan **riskreducering**: vissa roller kanske behöver fullt stöd (t.ex. media-elever), andra bara läsning eller helt avstängt.

Förtydligande (sammanfattning):

- *BYOD:* separera **skoldata** från **privatdata**; var tydlig med vad som loggas.
- *USB och externa enheter:* differentiera per program/roll; logga beslut.
- *Rese-scenarier:* nätprofiler/VPN gör att “skolan” följer med även utanför skolans väggar.

Exempel: *En vikarie loggar in på sin privata iPad och får **bara** access till lärarportalen via en skyddad container. När uppdraget är slut kan skolan rensa container-delen utan att röra privata bilder och appar.*

3.7.6 Varför Windows Server ofta är navet – och var Linux glänser

*I många svenska miljöer inom offentlig sektor och utbildning fungerar **Windows Server med Active Directory (AD DS)** som ett **nav för identitet, policy och infrastruktur**. Det beror både på klientflottans sammansättning (ofta Windows) och på att AD samlar **autentisering (Kerberos/NTLM)**, **grupp- och rättighetsmodeller**, **DNS/DHCP** och **gruppolicy (GPO)** i en sammanhängande helhet. Alltifrån skrivare och filresurser till inloggningsupplevelse kan därför kopplas till samma identitetskälla. Under senare år har många organisationer gått mot **hybrida upplägg** där ett lokalt AD samverkar med **molnidentitet och MDM** (t.ex. Entra ID/Intune), så att traditionella GPO:er och moderna, internetförsta policyer kan samexistera.*

Linuxservrar fyller samtidigt en **stark och bred tjänsteroll**. De används ofta för **filer och utskrifter** (Samba/CUPS), **webb och applikationer** (NGINX/Apache, applikationsramverk), **infrastrukturtjänster** (reverse proxy, cache/proxy, databaser, RADIUS, syslog), **automatisering** (t.ex. Ansible) och **containrar** (Docker/Kubernetes). Linux kan även bära **identitetstjänster** (t.ex. Samba-AD-DC eller FreeIPA/IdM) när arkitekturen kräver det, men där Windows-klienter dominerar är AD ofta fortsatt den **primära identitetskällan**, medan Linux fungerar som **stabila**

121

tjänstenoder runt navet. Valet handlar mindre om “bäst i test” och mer om **arkitektur**: *AD där sammanhållen klientidentitet och policystyrning är centralt; Linux där öppna, skalbara servertjänster och plattformar behövs* – med **interoperabilitet** via LDAP/Kerberos/SMB som gör att helheten sitter ihop.

Exempel: *Ett verksamhetsnät använder AD för konton, grupper, DNS/DHCP och skrivare. Runtom levererar Linux en filresurs för elevmaterial, en webbtjänst för schema och en containerplattform för interna appar. Klienterna hanteras med MDM så att policy och åtkomst är konsekvent även utanför kontoret.*

3.7.7 Vanliga arketyper i organisationer

I praktiken framträder några återkommande mönster för hur klienter och tjänster hanteras. De skiljer sig främst i var identiteten bor, hur policy distribueras och var data/appar levereras.

Klassisk on-prem
Identitet och policy ligger i lokalt **AD** med **GPO** för Windows-klienter; **Configuration Manager** eller likvärdigt används för paket och imaging; **Jamf** används ofta för macOS/iPadOS; fil- och utskriftstjänster körs lokalt. Fördelen är förutsägbarhet och låg latens på plats. Nackdelen är ett starkt beroende av skolans nät och schemalagda uppkopplingar.

Hybrid
Ett lokalt AD samverkar med molnidentitet och **MDM** (t.ex. Entra ID/Intune för Windows, Jamf för Apple). GPO hanterar miljöer som mår bra av platsnära styrning (t.ex. datasalar), medan bärbara enheter och plattor följer **internetförst-policy** via MDM. Filresurser rör sig delvis mot moln, och åtkomst styrs villkorat utifrån enhetens efterlevnad. Fördelen är att både platsnära och rörliga behov får stöd. Nackdelen är ökad komplexitet i gränssnittet mellan system.

Cloud-first
Identitet och policy lever primärt i molnet; **Intune** hanterar Windows, **Jamf/Android Enterprise** används där det passar; lokalt AD minimeras eller avvecklas; applikationer och filer levereras som molntjänster. Fördelen är att hanteringen i hög grad blir oberoende av plats och VLAN. Nackdelen är beroendet av externa tjänster och behovet av väl definierade processer för efterlevnad och dataskydd.

Exempel: *I en kommun med flera skolor drivs salar med platsnära policy via GPO för stabil uppstart och låg latens, medan lärarbärbara och plattor följer MDM-policy över internet. Identiteten är gemensam, men styrningen sker via två kompletterande spår.*

3.7.8 Vanliga fallgropar och hållbara tumregler

De problem som uppstår i MDM-miljöer beror oftare på **otydliga målbilder** än på teknikfel. Om önskat **enhetstillstånd** inte är definierat blir policyer motsägelsefulla. Om **efterlevnadskriterier** är vaga blir åtkomstregler svårförutsägbara. Och om **insyn och loggning** inte kommuniceras tydligt riskerar förtroendet att urholkas. Stabilitet uppnås när grundkraven är mätbara, när skolägda och

122

privata enheter hanteras separat och när nätanslutning görs friktionsfri genom förkonfigurerade profiler.

Förtydligande (sammanfattning):

- *Mätbara grundkrav:* kryptering, patchnivå, skärmlås, brandvägg.
- *Tydlig segmentering:* skolägda enheter vs **BYOD** med olika policy och rensningsrättigheter.
- *Förkonfigurerad anslutning:* **WLAN**-/VPN-profiler och certifikatlivscykel som del av MDM.
- *Transparens:* vad som samlas in/loggas, varför det görs och vad som uttryckligen inte övervakas.

Exempel: *Införandet av BYOD möter initial skepsis. När informationen förtydligar att MDM endast hanterar arbetsprofilens appar, OS-version och krypteringsstatus—och inte privata foton, SMS eller webbhistorik—ökar acceptansen och supportärenden minskar.*

123

📌 Kontrollfrågor

1. Vad är den konceptuella skillnaden mellan **GPO** och **MDM**?
2. Hur omsätter MDM “säkerhet” till **mätbara tillstånd**, och varför är det viktigt för åtkomst?
3. När passar **Intune** bäst, när passar **Jamf** bäst, och vad vinner du på att kombinera dem?
4. Vad innebär **Android Work Profile** jämfört med **Fully Managed**, och hur påverkar det BYOD?
5. Vilka fem saker vill du nästan alltid ha som **MDM-policy** på skolägda datorer?
6. Hur fungerar **zero-touch** i Windows/Apple/Android i stora drag, och varför förenklar det support?
7. Vad menas med **selektiv wipe** och när är det avgörande?
8. Varför är **certifikatlivscykel** central i WLAN/VPN-profiler, och hur hjälper MDM?
9. I vilken roll är **Windows Server/AD** oftast navet, och när lyfter du hellre in **Linuxservrar**?
10. Beskriv en typisk **hybrid**-arkitektur i skola och hur MDM knyter ihop den.

124

🔗 Fördjupningslänkar

- **Microsoft Intune** – översikter och scenarier: https://learn.microsoft.com/mem/intune/
- **Windows Autopilot** – koncept: https://learn.microsoft.com/windows/deployment/windows-autopilot/
- **Conditional Access & device compliance**: https://learn.microsoft.com/entra/identity/conditional-access/
- **Jamf Pro** – dokumentation: https://docs.jamf.com/
- **Apple School/Business Manager & Automated Device Enrollment**: https://support.apple.com/guide/apple-business-manager/welcome/web
- **Managed Apple IDs och User Enrollment**: https://support.apple.com/guide/apple-business-manager/intro-to-user-enrollment/web
- **Android Enterprise** – översikt: https://developers.google.com/android/work
- **Samba (SMB på Linux)** – docs: https://www.samba.org/samba/docs/

125

✍️ Egna anteckningar

...

126

4. Säkerhet och övervakning

4.1 Nätverkssäkerhetens grunder

4.1.1 Grundidéer: lager, rättigheter och synlighet

Nätverkssäkerhet handlar i grunden om att **minska konsekvensen** när något går fel, inte om att garantera att inget någonsin händer. Två idéer ramar in allt annat: **försvar i flera lager (defense in depth)** och **minsta möjliga rättighet (least privilege)**. Lager-tänket gör att ett enskilt fel – en sårbar klient, ett läckt lösenord, en missad uppdatering – inte automatiskt blir ett stort driftstopp. Rättighetsprincipen gör att ett kompromitterat konto eller en felkonfigurerad tjänst inte får mer makt än vad den faktiskt behöver. Allt detta står och faller med **synlighet**: utan loggar, larm och mätningar vet man varken vad som hände eller om åtgärden fungerade.

I praktiken börjar säkerhetsarbetet med tre frågor: **Vad skyddas? Mot vad? Hur märks det?** Tillgångar (elevdata, lärarmaterial, administration), hot (misstag, skadlig kod, obehörig åtkomst) och **telemetri** (loggar, larm, mätvärden). När det är definierat blir prioriteringen enklare: **segmentering** (VLAN/zoner), **uppdateringar**, **stark identitet** (MFA, gruppbaserad åtkomst), **baspolicys** (klient/servrar med brandvägg på) och **loggning som faktiskt läses** ger ofta mest effekt.

Förtydligande (sammanfattning):

- *Lager:* små fel blir inte stora.
- *Rättigheter:* konton/tjänster har bara det de behöver.
- *Synlighet:* utan logg/larm finns ingen säkerhet i praktiken.
- *Prioritering:* segmentering, patchning, stark identitet ger hög “träff per timme”.

Exempel: *Elevklienter och serversystem ligger i olika VLAN. Även om en elevdator infekteras kan skadlig kod inte prata direkt med servern – brandvägg och routingregler står i vägen. Händelsen syns i loggarna och kan hanteras innan den sprider sig.*

4.1.2 CIA-triaden

CIA-triaden beskriver tre mål: **konfidentialitet** (bara rätt mottagare kan läsa), **integritet** (data är oförändrad) och **tillgänglighet** (tjänsten fungerar när den behövs). Varje kontroll bör kunna motiveras med vilket C, I eller A den stöttar. Kryptering och behörighet skyddar **C**, signering och checksummer skyddar **I**, redundans och kapacitetsplanering skyddar **A**.

4.1.3 Brandväggar: klassisk → NGFW (Next-Generation Firewall)

Brandväggar uttrycker vad som **är tillåtet mellan zoner**; allt annat nekas. En **klassisk, stateful brandvägg** arbetar främst på lager 3/4: käll-/mål-IP, portar och protokoll, och håller reda på anslutningars tillstånd (t.ex. TCP SYN/ACK). Det räcker långt för kända flöden (“server X får prata TCP/443 ut”), men blir trubbigt när många applikationer **delar samma portar**.

127

En **NGFW (Next-Generation Firewall)** lägger till **lager-7-förståelse**. Den kan identifiera **applikationer** oavsett portar, koppla regler till **användaridentitet** (via t.ex. AD/Entra), och utföra **innehållsinspektion**. I praktiken innebär det moduler som **IPS (Intrusion Prevention System)**, **URL-/webbfiltrering**, möjlighet till **TLS-inspektion** (dekryptera, kontrollera, kryptera om), **sandboxning** av misstänkta filer samt flöde av **threat intelligence**. Poängen är inte fler "knappar", utan att **policyn kan uttryckas närmare verksamheten**: tillåt rätt **tjänst** för rätt **grupp**, blockera osäkra mönster.

Förtydligande (sammanfattning):

- *Stateful brandvägg:* IP/port/protokoll + tillstånd; enkel och snabb.
- *NGFW:* app-igenkänning (lager 7), identitetsmedvetna regler, IPS, URL-filter, ev. TLS-inspektion, sandboxning, hotflöden.
- *Att tänka på:* prestanda (särskilt TLS-inspektion), integritet/juridik, kompatibilitet (certifikat-pinning) och tydliga undantag.

Exempel: *I en klassisk modell blir policyn "tillåt TCP/443 ut", vilket i praktiken öppnar för "allt som talar HTTPS". Med NGFW kan policyn i stället säga "tillåt Microsoft Update och ett fåtal samarbetsverktyg för Personal, men blockera okända TLS-appar och fildelning för Elever" – även om allt använder 443.*

4.1.4 IDS/IPS (Intrusion Detection/Prevention System) – och samspelet med NGFW

IDS/IPS kompletterar brandväggar genom att fokusera på *beteenden och mönster* i trafiken snarare än bara källor och portar. En **NIDS (Network Intrusion Detection System)** observerar trafik i nätet – ofta via **SPAN (Switched Port Analyzer)** eller en **TAP (Test Access Point)** – och ser flöden **mellan** system ("north-south" mot internet och "east-west" mellan zoner). En **HIDS (Host Intrusion Detection System)** tittar inne i värden (processer, loggar, filändringar) och fångar sådant som inte alltid syns på nätet.

Detekteringsmetoderna är två: **signaturbaserad** detektering jämför mot kända mönster (snabb och exakt för det kända), medan **anomalibas** letar avvikelser från "normal" trafik (bättre chans på okända hot, men kräver inlärning och kan ge fler falsklarm). Körs motorn **inline** blir den **IPS**: samma analys, men med möjlighet att **blockera** eller **reseta** misstänkta förbindelser i realtid; körs den vid sidan om (out-of-band) är rollen att **larma**.

Placering och signal. Vid gränsen mot internet fångas angrepp utifrån, men mycket verksamhetslogik passerar **internt** mellan klient- och serverzoner; där ger en NIDS-sensor på rätt länk hög signal. **Kryptering** (t.ex. TLS) begränsar innehållsinsyn; då blir **metadata** (tidsmönster, anslutningsfrekvens, kända domäner/IP:er) och **HIDS-telemetri** extra värdefullt. Ofta används **flera sensorer** för att skapa **sammanhang**: NIDS ser oväntade anslutningar, HIDS ser onormala processer, och loggkällor (AD, brandvägg, proxy) fyller i tidslinjen.

NGFW och IDS/IPS i samspel. Många NGFW innehåller en **IPS-motor**. Det ger fördelen att stoppa välkända mönster **i brandväggens policyflöde**. Samtidigt behåller **friliggande NIDS/HIDS**

128

värde: de kan placeras på **andra länkar** (t.ex. mellan klient- och serverzon) och fånga **east-west-rörelser** som inte passerar brandväggens kant, eller ge **second opinion** utan att riskera falskpositiv blockering. En vanlig väg är att börja i **detekteringsläge** och **blockera** först när regler har hög tillit – särskilt på systemkritiska länkar.

Förtydligande (sammanfattning):

- *NIDS/HIDS:* nät- respektive värd-perspektiv; kompletterar varandra.
- *Signatur vs anomali:* känt-snabbt kontra okänt-upptäckbart.
- *Inline vs out-of-band:* blockera vs larma.
- *Samspelet:* NGFW-IPS stoppar vid kanten; NIDS/HIDS ger djup och kontext internt.

Exempel: *En serverzon visar plötsliga utgående anslutningar mot ovanliga mål. NIDS (via TAP) larmar för "rare outbound" och HIDS på samma server rapporterar nyskapade processer i ett ovanligt katalogträd. Brandväggen sätter en tillfällig block på utgående trafik från den servern (hög-tillitsregel i IPS-läget), och korrelation med AD-loggar visar ett läckt tjänstkonto som spärras och roteras.*

4.1.5 Honeypots och honeytokens

Honeypots och **deception** skapar medvetna "mål" som ska vara tysta i normal drift. Poängen är inte att "locka angripare för sport", utan att skapa **tidig varning** och **telemetri** i zoner där man i övrigt vill se så lite trafik som möjligt. En enkel variant är en **honeytoken** (falsk uppgift/fil) som larmar när den läses.

Förtydligande (sammanfattning):

- *Syfte:* blixtlampa – inte produktion.
- *Placering:* i zoner där legitim kontakt är sällsynt.
- *Åtgärd:* tydlig ägare och process när det blinkar.

Exempel: *En "tyst" filserver-alias annonseras internt. All aktivitet dit larmar – signal som annars hade drunknat bland normaltrafik.*

4.1.6 Säkerhet i VLAN-design

VLAN är inte bara ordning och adressplan; det är ett **säkerhetsverktyg**. När zoner och flöden uttrycks i VLAN och routing/ACL minskar risken att "tillfälliga lösningar" skapar genvägar.

Återkommande byggstenar:

- **Minimera trunkar** och **pruna** (tillåt) bara relevanta VLAN per trunk; **native VLAN** sätts till ett **oanvänt** (t.ex. 999).
- **BPDU Guard** och **PortFast** på klientportar; **Root Guard** nära roten; **storm-control** för att dämpa toppar.

129

- **DHCP Snooping** (avgör vilka portar som får vara DHCP), **Dynamic ARP Inspection (DAI)**, **IP Source Guard** i accesslagret.
- **ACL på SVI** (gateway): “default deny – öppna minsta möjliga”.
- **Private VLAN** där klienter i samma VLAN inte ska kunna prata direkt.
- **802.1X** i accesslagret för att knyta **identitet till port** (MAB-fallback för skrivare/IoT).
- Separera **managementytor** (SSH/HTTPS/SNMP) i eget VLAN och nå dem via bastion/jump host.

Förtydligande (sammanfattning):

- *Zoner:* klient, server, gäst, management – och uttryckta flöden.
- *Accessskydd:* skydda mot felkopplade switchar och kringutrustning.
- *Förgiftning:* Snooping/DAI/Source Guard hindrar “falska sanningskällor”.

Exempel: *Ett “mystiskt” DHCP-scope dyker upp – klienter får fel adresser. Med* ***DHCP Snooping*** *blockeras den privata routern, och Snooping-tabellen pekar ut exakt switchport. Port Security begränsar MAC-antal och dämpar effekten av medtagen utrustning.*

4.1.7 Protokollanalys med Wireshark

Protokollanalys svarar på frågan **vad som faktiskt hände**. Wireshark blir då ett mikroskop som visar ramar, flaggor och tidslinjer. En bra metod undviker datatsjöar: börja med ett **smalt syfte** (“klienter får inte IP”, “långsam inloggning”), fånga **så nära källan som möjligt** (klient, port-SPAN, server), och **filtrera** både vid fångst och visning. Etik och lag spelar också in: **minimera** (bara det som behövs), **begränsa innehåll** (undvik att spara payload i onödan), och **rensa** material när analysen är klar.

Några vyer ger mycket värde: **tidslinje**, **Follow Stream**, **I/O Graphs** och **Expert Information**. Välkända filter: `dhcp || bootp` för adressdelning, `dns` för namn, `arp` för tabeller, `icmp` för nåbarhet, `tcp.flags.syn==1 && tcp.flags.ack==0` för handskakningar, `tcp.analysis.retransmission` för störningar, `tls.handshake` för att se *att* TLS sker, `ip.addr==X` för att isolera en värd. På **WLAN** krävs ibland **monitor-läge** för att se råa 802.11-ramar – annars syns mest det som passerar via AP/controller.

Förtydligande (sammanfattning):

- *Fråga först:* vad vill tidslinjen bevisa eller motbevisa?
- *Smala fångster:* nära källan, kort tid, rätt filter.
- *Hjälpytor:* Follow Stream, Expert Info, I/O Graphs.
- *Etik:* fånga minsta möjliga, skydda, rensa.

130

Exempel: *Klienter “saknar IP” i ett VLAN. Fångst visar DISCOVER men inga OFFER. På uplinken saknas* ***trusted-flagga*** *i DHCP Snooping, så relätrafiken stoppas. När uplinken sätts som* ***trusted*** *syns OFFER/REQUEST/ACK och klienterna får adresser direkt.*

131

📌 Kontrollfrågor

1. Vad menas med **defense in depth** och **least privilege**, och hur samverkar de?
2. Koppla tre kontroller till **C**, **I** respektive **A** i CIA-triaden.
3. Förklara skillnaden mellan **stateful brandvägg** och **NGFW (Next-Generation Firewall)**.
4. Vad gör en **NIDS (Network Intrusion Detection System)** respektive **HIDS (Host Intrusion Detection System)** – och när behövs båda?
5. Hur skiljer sig **IDS** från **IPS (Intrusion Prevention System)**, och vad innebär **inline** kontra **out-of-band**?
6. Varför kan **NIDS/HIDS** vara värdefulla även när NGFW redan har **IPS**?
7. Nämn fem byggstenar som stärker **VLAN-säkerhet** i accesslagret och motivera dem kort.
8. Vad gör **DHCP Snooping**, **DAI** och **IP Source Guard**, och hur hänger de ihop?
9. Beskriv en metod för att felsöka “långsam inloggning” med Wireshark utan att exponera onödig data.
10. Ge tre visningsfilter i Wireshark som snabbt ringar in problem med DHCP, DNS och TCP-anslutningar.

132

🔗 Fördjupningslänkar

- NIST – Computer Security Incident Handling (översikt): https://csrc.nist.gov/publications/detail/sp/800-61/rev-2/final
- NIST – Zero Trust (ramverkstänk): https://csrc.nist.gov/publications/detail/sp/800-207/final
- RFC 2131 – DHCP: https://www.rfc-editor.org/rfc/rfc2131
- RFC 1034/1035 – DNS grunder: https://www.rfc-editor.org/rfc/rfc1034
- RFC 826 – ARP: https://www.rfc-editor.org/rfc/rfc826
- RFC 8446 – TLS 1.3: https://www.rfc-editor.org/rfc/rfc8446
- Wireshark – User Guide: https://www.wireshark.org/docs/wsug_html_chunked/
- Cisco – Campus LAN Security (designidéer): https://www.cisco.com/c/en/us/solutions/enterprise-networks/index.html

133

✍️ Egna anteckningar

…

134

4.2 Åtkomstkontroll och behörigheter

4.2.1 AAA-modellen och identitet som “valuta”

AAA (Authentication, Authorization, Accounting) är tre perspektiv på åtkomst:

- **Authentication** – *autentisering*: att **bevisa** vem som försöker komma in (lösenord, certifikat, nyckel, passkey).
- **Authorization** – *behörighet*: att **besluta** vad den identifierade får göra (roller, grupper, regler).
- **Accounting** – *spårbarhet/loggföring*: att **registrera** vad som faktiskt hände (vem gjorde vad, när, från var).

I praktiken bärs identiteten runt i en **session** (t.ex. Kerberos-biljett eller webb-token). Behörighet uttrycks ofta som **RBAC (Role-Based Access Control)** – “rollen avgör” – ibland kompletterat med **ABAC (Attribute-Based Access Control)** där attribut som *plats, tid, enhetens status* påverkar beslutet. AAA binder ihop teknik och verksamhet: man bevisar *vem*, beslutar *vad*, och **bevisar i efterhand** via loggar *att* rätt sak hände.

Förtydligande (sammanfattning):

- Autentisering = **bevis**; behörighet = **beslut**; accounting = **bevis i efterhand**.
- RBAC modellerar **roller** (grupper); ABAC lägger på **villkor** (t.ex. “från hanterad enhet”).
- Sessioner bär identiteten; loggar och tidsstämplar gör den **revisionsbar**.

Exempel: *En användare loggar in i domänen (autentisering), hamnar i grupperna* ***G_Larare*** *och* ***G_Mentor*** *(behörighet), och allt deras konto gör i filservern loggas med tid, IP och filstig (accounting).*

4.2.2 Multifaktorautentisering (MFA) och motstånd mot phishing

MFA (Multifaktorautentisering) kombinerar **minst två** av: *något du vet* (lösenord/Pin), *något du har* (telefon, säkerhetsnyckel, certifikat), *något du är* (biometri). Syftet är att **bryta kedjan** där ett läckt lösenord annars räcker. Alla “två steg” är inte lika: **phishing-resistenta** metoder som **FIDO2/WebAuthn** (säkerhetsnyckel/passkey) eller **klientcertifikat** är robustare än SMS-koder och enkla push-notiser. Riskbaserade signaler – plats, ovanlig tid, okänd enhet – kan **förstärka** kravet (ABAC), men ersätter inte MFA.

Förtydligande (sammanfattning):

- Faktorer: **vet/har/är** – blanda minst två.
- Starkast: **FIDO2/WebAuthn** och **certifikat** (motstånd mot phishing).
- Praktik: minimera **push-trötthet**; använd villkor (ABAC) för att be om starkare faktor vid risk.

135

Exempel: *E-post öppnas på en skolhanterad laptop utan extra steg (enheten är känd och compliant). Samma konto på en privat platta kräver* ***passkey*** *innan åtkomst beviljas.*

4.2.3 Gruppolicyer och lösenordspolicyer (Windows-miljö)

I Windowsmiljö uttrycks mycket av **authorization** via **grupper** och **GPO (Group Policy Objects)**. GPO styr både **User Rights Assignment** (vad ett konto *får göra* i OS: logga in lokalt, via RDP, köra som tjänst) och miljöpolicys som påverkar **säker nivå** (låsning, brandvägg, enhetsregler). Lösenord, låsning och relaterade regler definieras i en **domänpolicy** och kan förfinas med **FGPP (Fine-Grained Password Policies)** för särskilda grupper.

Det finns ett skifte i rekommendationer: **långa lösenfraser** och **MFA** ger bättre effekt än täta tvångsbyten och komplexitetsdikt. Tvingad rotation utan indikator på läcka skapar ofta **svagare** lösenord och mer återanvändning. För lokala administratörer ligger säkerheten i **Windows LAPS** (unika, roterade lösenord per klient/server) – inte i "ett gemensamt hemligt lösenord".

Förtydligande (sammanfattning):

- GPO: **User Rights**, **säkerhetslinjer** (brandvägg, lås), **policyuttryck** per OU.
- Lösenord: **längd + MFA** slår "komplexitet + frekvent byte".
- Lokala admin: **Windows LAPS** för unika hemligheter.

Exempel: *RDP tillåts endast för gruppen* ***G_Admin*** *via User Rights Assignment. Lösenordspolicyn kräver minst 15 tecken; gruppen* ***G_HögRisk*** *får dessutom FGPP med lågnivå-tröskel och obligatorisk MFA för alla molnappar.*

4.2.4 Behörigheter och arv i Linux och Windows

Båda världarna behöver svara på samma fråga: *vem får vad, och hur ärvs det vidare?* Svaren ser olika ut i verktyg – men logiken rimmar.

Linux – DAC, grupper och ACL
Linux bygger på **DAC (Discretionary Access Control)**: **ägare**, **grupp**, **andra** med rättigheterna **r/w/x**. För kataloger betyder **x** "få gå igenom/angöra", **w** "skapa/ta bort i katalogen". **setgid** på kataloger gör att **nya filer** får **katalogens grupp** (bra för gemensamma träd). **sticky bit** på gemensamma "inlämningskorgar" gör att man inte kan ta bort andras filer. **POSIX-ACL** kompletterar med finare rättigheter och **default-ACL** som ärvs nedåt när nya objekt skapas. **sudo** ger spårbar eskalering till administrativa kommandon; **polkit** och **Linux capabilities** kan ge mer granulär kontroll per tjänst.

Windows – NTFS-ACL och arv
NTFS använder **ACL (Access Control List)** med **ACE**-poster (Allow/Deny) för användare och grupper. Rättigheter **ärvs** nedåt i träd om inte arv bryts. **Ägarskap** styr rätten att ta över/ändra rättigheter. Effektiv rättighet blir summan av tilldelade **Allow** minus eventuella **Deny** (som vinner). Skillnad mot Linux är att arvet är **explicit synligt** och att **ärvda** poster skiljs från **direkta** poster. I nätutdelningar "multipliceras" **share-rättigheter** med **NTFS** – den mest begränsande gäller.

136

Gemensam nämnare – grupper som språk
I båda miljöer blir **grupper** det praktiska stället där **RBAC** uttrycks. På Linux kopplas grupper till **ägarskap/ACL**; i Windows kopplas grupper till **NTFS-ACL** (och i domänen till **GPO** och **User Rights**). Arv gör att rättigheten följer **strukturen**, inte manuella undantag.

Förtydligande (sammanfattning):

- Linux: **ägare/grupp/andra**, **setgid/sticky**, **default-ACL**; **sudo** för spårbar eskalering.
- Windows: **NTFS-ACL** med arv och **ägarskap**; **share × NTFS** ger effektiv rätt.
- Båda: **grupper** uttrycker roll; låt **arv** göra det tunga lyftet.

Exempel: *Ett ämnesträd på filservern har **setgid** och **default-ACL** som ger "Larare" **write**. I NTFS-världen motsvaras detta av en ärvd **Allow-Modify** för gruppen **G_Larare** på rotmappen. I båda fall räcker det att lägga en ny lärare i rätt **grupp** – inget handpill i djupt liggande mappar.*

4.2.5 Vanliga mönster och missförstånd

Ett par mönster återkommer oavsett plattform: **minsta möjliga rättighet** (ge bara det som krävs), **separation of duties** (dela ansvar så att ingen ensam kan skapa och godkänna sin egen åtkomst), och **förutsägbar arvsväg** (rättigheter definieras högt upp och ärvs ned). Missförstånd uppstår ofta när **individuella undantag** landar på objekt långt ner i ett träd – de fungerar idag, men bryter ihop imorgon. Ett annat är att blanda ihop **OS-rättigheter** (vem får logga in) med **data-rättigheter** (vad får läsas/ändras); de lever i olika lager.

Förtydligande (sammanfattning):

- Satsa på **gruppbaserad** behörighet och **arv**; undvik manuella "lappar och lag".
- Separera **logon-rättigheter** från **filrättigheter** i tanke och praktik.
- Låt **accounting** (loggar) bevisa efterlevnad och underlätta felsökning.

Exempel: *Ett enskilt "Deny" på en djupt liggande mapp "låser ute" en lärare oförutsett. Efter städning tillbaka till rent arv (bara **Allow** via grupp) försvinner sidoeffekterna och felsökning blir begriplig.*

137

📌 Kontrollfrågor

1. Beskriv **AAA (Authentication, Authorization, Accounting)** och ge ett konkret exempel på varje del.
2. Vad skiljer **RBAC** från **ABAC**, och när är ABAC särskilt användbart?
3. Vilka MF-metoder räknas som **phishing-resistenta**, och varför?
4. Vad är **User Rights Assignment** och hur skiljer det sig från filrättigheter?
5. När passar **FGPP (Fine-Grained Password Policies)**, och varför föredras **långa lösenfraser + MFA** framför täta tvångsbyten?
6. Förklara **setgid** och **sticky bit** på kataloger i Linux och hur de hjälper i gemensamma mappar.
7. Hur fungerar **arv** i **NTFS-ACL**, och varför bör **Deny** användas sparsamt?
8. Varför leder **individuella undantag** ofta till oförutsägbart beteende över tid?
9. Ge ett scenario där **ABAC** (t.ex. enhets-compliance) påverkar åtkomstbeslutet.
10. Vilka loggkällor behövs för **Accounting** när man vill kunna spåra “vem gjorde vad, när, från var”?

138

🔗 Fördjupningslänkar

- NIST SP 800-63-3 – Digital Identity Guidelines: https://pages.nist.gov/800-63-3/
- FIDO Alliance – Overview (passkeys/säkerhetsnycklar): https://fidoalliance.org/overview/
- W3C – Web Authentication (WebAuthn) Level 2: https://www.w3.org/TR/webauthn-2/
- Microsoft Learn – User Rights Assignment (Group Policy): https://learn.microsoft.com/windows/security/threat-protection/security-policy-settings/user-rights-assignment
- Microsoft Learn – Fine-Grained Password Policies (FGPP): https://learn.microsoft.com/windows-server/identity/ad-ds/manage/component-updates/fine-grained-password-policies
- Microsoft Learn – Windows LAPS (översikt): https://learn.microsoft.com/windows-server/identity/laps/laps-overview
- man7 – POSIX ACL (`acl(5)`): https://man7.org/linux/man-pages/man5/acl.5.html
- man7 – `chmod(1)` (filrättigheter): https://man7.org/linux/man-pages/man1/chmod.1.html
- sudoers manual (kontrollerad eskalering): https://www.sudo.ws/docs/man/sudoers.man/
- polkit – PolicyKit-dokumentation: https://www.freedesktop.org/software/polkit/docs/latest/

139

✍️ Egna anteckningar

...

140

4.3 Övervakning och loggning

4.3.1 Varför loggar och övervakar man?

Övervakning och loggning handlar om **synlighet**: att kunna svara på *vad hände, när, var, och av vem* – och att upptäcka när något **avviker** innan det blir ett driftstopp eller en incident. Loggar är dessutom bevismaterial när något måste utredas i efterhand. Två praktiska förutsättningar bär hela området: **tidssynk** (utan gemensam klocka går det inte att korrelera händelser) och **centralisering** (att samla loggar på en plats där de kan sökas, korreleras och larma).

I ett modernt nät är loggkällorna många: klienter och servrar, nätverksutrustning (switchar, routrar, brandväggar, **WLAN**-kontroller), katalogtjänster (AD/LDAP), DNS/DHCP, applikationer och identitetstjänster. Poängen är inte att "logga allt", utan att logga **rätt** saker med **lagom detaljer** – och ha en tydlig **ägare** som faktiskt tittar på signalerna.

Förtydligande (sammanfattning):

- *Synlighet:* loggar = svar på "vad, när, var, vem".
- *Grundkrav:* **tidssynk** och **centralisering**.
- *Urval:* logga det som behövs för att se avvikelser och felsöka – inte allt.

Exempel: *En kort driftstörning i inloggning spåras på minuter tack vare gemensamma tidsstämplar: DNS-logg visar misslyckat uppslag, brandväggslogg visar blockerat flöde mot extern resolver, AD-logg visar ökade autentiseringsfel – kedjan blir tydlig och åtgärdas.*

4.3.2 Logghantering: Event Viewer (Windows), syslog (Unix/Linux/nät)

Windows loggar till **Event Viewer** (Händelsevisaren) i kanaler som *System*, *Application*, *Security* och applikationsspecifika loggar. Händelser har **nivåer** (Information/Warning/Error/Critical), **källor** och ofta **händelse-ID** som gör dem sökbara. Säkerhetsloggar (t.ex. inloggning, gruppändringar, policy) blir väldigt användbara när de **korreleras** med nät- och applikationsloggar.

Unix/Linux och nätutrustning använder oftast **syslog** – ett standardprotokoll och ett **format** för loggar. Syslog har **facility** (vilken del av systemet) och **severity** (emerg, alert, crit, err, warning, notice, info, debug). På Linux skrivs loggar numera ofta till **systemd-journald** (den binära journalen), och skickas därifrån vidare till en syslog-tjänst (t.ex. **rsyslog**/**syslog-ng**) för central insamling. Nätutrustning skickar syslog direkt till en **central loggserver**. UDP/514 är klassiskt, men **TCP/TLS** bör användas när det är möjligt för att undvika tapp och skydda innehåll.

För att loggar ska vara **användbara** behövs också: **namnstandarder** (värdnamn som går att tolka), **loggrotation** och **retention** (hur länge sparas vad), **åtkomstkontroll** (vem får läsa), samt **normalisering** (göra olika källor jämförbara) i den centrala plattformen.

Förtydligande (sammanfattning):

- Windows: **Event Viewer** med kanaler, nivåer, event-ID.

- Unix/Linux & nät: **syslog** (facility/severity) + ofta **journald** på Linux.
- Praktik: **TLS/TCP** när det går, tydliga namn, genomtänkt retention och rättigheter.

Exempel: *En switchport "flappar" (går upp/ner). Syslog från switchen visar portens statusändringar, Event Viewer på filservern visar samtidigt tappade nätverksresurser. Tidslinjen kopplar ihop händelserna och gör felet lätt att lokalisera.*

4.3.3 Verktyg: Wazuh, SNMP, SIEM

Wazuh är en **öppen plattform** för säkerhetsövervakning som kombinerar logginsamling med funktioner som **FIM (File Integrity Monitoring)**, sårbarhets-indikering och regelbaserade larm. Den använder agenter (på klient/servrar) men kan också ta emot loggar via syslog. Vanligt är att lagra/söka i en **Elastic/OpenSearch-stack** med färdiga instrumentpaneler. Poängen är att få **sammanhang**: "denna fil ändrades, på denna värd, direkt efter denna inloggning".

SNMP (Simple Network Management Protocol) är nätutrustningens telemetri. Med **GET** kan man läsa mätvärden (interfacestatistik, CPU, temperatur), med **TRAP/INFORM** skickar enheten **händelser** proaktivt (länk ner/upp, fläktfel, överhettning). **MIB/OID** beskriver vad som kan läsas. Version **SNMPv3** ger **autentisering och kryptering** (till skillnad från v1/v2c som använder community-strängar). I övervakning utgör SNMP "hälsopulsen" som kompletterar loggar.

SIEM (Security Information and Event Management) är den centrala plattformen som **samlar, normaliserar, korrelerar** och **larmar** på tvärs av källor: serverloggar, nätflöden, AD-händelser, brandvägg, DNS, proxy. Exempel på funktioner: **korrelationsregler** ("många misslyckade inloggningar + lyckad från ovanlig klient"), **dashboard** med indikatorer, **larmhantering** och **playbooks** för åtgärd. SIEM är inte lika med "dyra licenser" – principen kan uppnås med både öppna och kommersiella byggstenar – men **tidsstämplar, normalisering och ägarskap** är alltid avgörande.

Förtydligande (sammanfattning):

- **Wazuh:** agent + regler + dashboards; binder ihop fil/host/nät-händelser.
- **SNMP:** mätvärden och traps från nät- och infrastruktur; **SNMPv3** för säkerhet.
- **SIEM:** central hjärna för insamling, korrelation och larm.

Exempel: *SNMP visar ökande felräknare på ett switchinterface; Wazuh larmar samtidigt om ovanligt många TCP-återförsök på en server i samma VLAN. SIEM visar korrelationen på en tidslinje – felet ligger på länken mellan dem, inte i applikationen.*

4.3.4 Att tolka loggar och upptäcka avvikelser

Loggar blir meningsfulla när de sätts i **sammanhang**. Ett fungerande arbetssätt börjar med en **fråga** ("varför är inloggning långsam?", "vem skapade denna delning?"), väljer **källor** (AD/brandvägg/klient), och binder ihop händelser via **tidsstämplar** och **identiteter** (användare, värd, IP, process). Två vägar används parallellt:

142

- **Regel- och tröskelbaserat:** “>10 misslyckade inloggningar (Windows Event ID) från samma IP på 5 min” → bruteforce-indikator; “ny medlem i lokal admins” → larma.
- **Avvikelsebaserat:** “denna klient gör plötsligt tusentals DNS-frågor mot externa domäner” eller “den här servern pratar ut på en port den aldrig använt”.

Båda kräver **baslinjer**: hur “ser normalt ut”? Utan baslinje blir allt antingen larm eller brus. Därför finjusteras regler (trösklar, **suppressionsfönster**, **deduplicering**) tills larmen är **få men viktiga**. Och varje “rött larm” behöver en **runbook**: vem äger det, vilka steg tas, när eskaleras?

Förtydligande (sammanfattning):

- *Frågedriven metod:* fråga → källor → korrelation → slutsats.
- *Två blickar:* regler/trösklar **och** avvikelse/baslinje.
- *Tuning:* trösklar, dedup, suppression; **runbooks** för åtgärd.

Exempel: *SIEM larmar: “Många misslyckade inloggningar följt av en lyckad (samma konto), därefter ny RDP-session från ovanlig klient”. Korrelation med AD-logg visar att kontot lades till i en lokal admin-grupp — en tydlig kedja som stoppas genom att spärra kontot och rulla lösenord/nycklar.*

4.3.5 Larm, brus och ansvar (vem gör vad när det blinkar?)

Ett övervakningssystem utan **ägare** blir snabbt en “blinkande tavla”. Definiera **allvarsnivåer**, **SLA** för hantering (t.ex. kritiskt = 15 minuter till åtgärd), och **vägar** för eskalering. Arbeta bort **falskpositiv** genom att justera regler, lägga **undantag** för kända mönster och tidsbegränsa undertryckning. Testa larmkedjan med **table-top-övningar** och simulera verkliga händelser (t.ex. många misslyckade logins) för att se att kedjan håller.

Förtydligande (sammanfattning):

- *Sätt nivåer och tider:* så vet alla vad som förväntas.
- *Minska brus:* justera, vitlista kända mönster, följ upp.
- *Öva:* simuleringar visar var kedjan brister.

Exempel: *Ett “stormlarm” om misslyckade inloggningar visar sig bero på lösenordsbyte i en klass. En suppression-regel införs (1 h för just den källan), men ett separat larm ligger kvar för inloggningar* ***från internet*** *– så att verkliga försök inte tystas.*

4.3.6 Dataskydd och etik

Loggar kan innehålla **personuppgifter** (användarnamn, IP, tid, ibland filnamn). Det gör **dataminimering**, **behörighetsstyrning**, **retention** och **syftesbeskrivning** viktiga. Loggar ska skyddas lika omsorgsfullt som andra system och bara vara tillgängliga för dem som behöver dem i sitt arbete.

Förtydligande (sammanfattning):

143

- *Minimera:* samla det som behövs – inte mer.
- *Skydda:* åtkomstkontroll, kryptering, revision.
- *Rensa:* retention kopplad till syfte och lagkrav.

Exempel: *Säkerhetsloggar sparas 12 månader för incidentutredning och revisionskrav; applikationsloggar med personnära data rensas efter 30 dagar. Åtkomst till SIEM är gruppstyrd och loggas.*

📌 Kontrollfrågor

1. Varför är **tidssynk** och **centralisering** grundläggande för användbara loggar?
2. Vilka är de viktigaste skillnaderna mellan **Event Viewer** och **syslog**?
3. Vad gör **SNMP (Simple Network Management Protocol)** i övervakning, och varför föredras **SNMPv3**?
4. Beskriv hur **Wazuh** kan ge sammanhang mellan loggar, filändringar och inloggningar.
5. Vad betyder **SIEM (Security Information and Event Management)** i praktiken – vilka tre saker gör ett SIEM alltid?
6. Ge exempel på en **regelbaserad** respektive en **avvikelsebaserad** detektion.
7. Varför behövs **baslinjer** och hur påverkar de antalet falsklarm?
8. Vad är en **runbook**, och varför är den central när ett larm går?
9. Nämn tre beslut man måste ta kring **retention** och **åtkomst** till loggar.
10. Vad bör loggas för att kunna utreda “plötsligt långsam inloggning” utan att samla onödigt innehåll?

144

🔗 Fördjupningslänkar

- Microsoft – Event Viewer (översikt): https://learn.microsoft.com/windows/client-management/troubleshoot-event-viewer
- systemd-journald (man-sida): https://www.freedesktop.org/software/systemd/man/latest/systemd-journald.service.html
- rsyslog – dokumentation: https://www.rsyslog.com/doc/
- IETF – Syslog Protocol (RFC 5424): https://www.rfc-editor.org/rfc/rfc5424
- SNMPv3 – översikt (Cisco): https://www.cisco.com/c/en/us/tech/snmp/
- Wazuh – dokumentation: https://documentation.wazuh.com/
- Elastic Security (SIEM-koncept, översikt): https://www.elastic.co/security/siem
- Microsoft Sentinel (SIEM/SOAR-koncept, översikt): https://learn.microsoft.com/azure/sentinel/overview
- FIRST – Common Event Expression & Logging best practices: https://www.first.org/cvss/user-guide & https://www.first.org/resources/papers/logging
- NIST – Guide to Computer Security Log Management (SP 800-92): https://csrc.nist.gov/publications/detail/sp/800-92/final

145

✍️ Egna anteckningar

…

146

4.4 Incidenthantering och backup

4.4.1 Rutiner, roller, dokumentation

Incidenthantering handlar om att begränsa påverkan, återställa drift och lära av händelsen så att den inte händer igen. För att lyckas krävs tydliga rutiner, definierade roller och dokumentation som alla faktiskt hittar och använder. En enkel men effektiv process följer NIST-modellen: **Förberedelse → Identifiera → Inneslut → Åtgärda → Återställ → Lärdomar**. I skolmiljö där många användare delar nät är snabb kommunikation och spårbarhet minst lika viktiga som teknik.

Roller (förtydligande):

- **Incident Manager:** leder arbetet, sätter prioritet/severity, ser till att logg förs och att beslut dokumenteras.
- **Teknikansvarig (t.ex. Nät):** leder felsökning/åtgärd inom sitt område.
- **Skribent (Scribe):** för **tidslinje**, kommandon, observationer och beslut.
- **Kommunikation:** informerar ledning/personal/elever enligt mallar.
- **Ägare efteråt:** äger uppföljning och permanenta förbättringar.

Dokumentation som bör finnas nära till hands:

- **Kontaktlista & eskalering:** jour, leverantörer, ISP, beslutsfattare.
- **Topologi/IP/VLAN:** aktuella diagram, adresstabeller, brandväggsflöden.
- **Runbooks & mallar:** vanliga fel (DHCP, DNS, WLAN), **incidentrapport-mall**, ändringsmall.
- **Backuppolicy:** RPO/RTO per system, lagringsplatser, retention, testplan.

Exempel: *En lärare anmäler “ingen kan logga in i A-huset”. Incident Manager skapar ärende (P2), Teknikansvarig (Nät) leder felsökning, Skribent loggar alla steg. Kommunikatören postar status i personalgruppen var 20:e minut tills felet är löst.*

Mall: Incidentrapport (tom)

- **Ärende-ID:**
- **Datum/tid (start–slut):**
- **Severity (P1–P4):**
- **Påverkan:** *vilka, var, hur mycket*
- **Symptom:** *vad märktes*
- **Tidslinje:**
    - *[klockslag] → åtgärd/observation*

147

    - *[klockslag] → åtgärd/observation*
- **Rotorsak:**
- **Akuta åtgärder:**
- **Återställning (RTO/RPO utfall):**
- **Korrigerande/förebyggande åtgärder (CAPA):** *vad ändras, vem äger, när klart*
- **Bilagor:** *loggar, kommandon, skärmdumpar*
- **Godkänd av / datum:**

Exempel: Incidentrapport (ifylld, kort)

- **Ärende-ID:** 2025-08-14-B-DHCP-VLAN20
- **Datum/tid:** 08:12–09:03
- **Severity:** P2 (elevnät påverkat i ett hus)
- **Påverkan:** ~18 klassrum i B-huset, VLAN 20 (elev), “ingen IP”
- **Symptom:** Klienter fick 169.254.x.x (APIPA)
- **Tidslinje:**
    - 08:12 – anmälan: “ingen internet i B-huset”
    - 08:15 – `show ip dhcp snooping` → uplink ej *trusted*
    - 08:18 – `show interfaces trunk` → VLAN 20 saknas i allowed-lista
    - 08:22 – Åtgärd: trust uplink + tillåt VLAN 20 på trunk
    - 08:26 – `ipconfig /renew` OK på testklient
    - 08:40 – Lägesuppdatering till personal: “åtgärd pågår, förbättring syns”
    - 09:03 – Stängning, alla testpunkter gröna
- **Rotorsak:** Ny accesswitch med ofullständig trunk-mall
- **Akuta åtgärder:** Uppdaterad trunk + trust uplink
- **Återställning:** RTO 51 min, RPO ej relevant
- **CAPA:** Uppdatera mallen “Ny accesswitch” (ägare: Nät, klart 2025-08-20). Lägg auto-validering i övervakning: larma på DHCP-svarstid > X ms.
- **Bilagor:** Kommando-logg, skärmdumpar från övervakning
- **Godkänd:** IT-chef 2025-08-14

148

4.4.2 Backupmetoder: fullständig, inkrementell och differentiell

Tre klassiska metoder balanserar **backupfönster**, **lagringsutrymme** och **återställningstid**. Valet styrs av dina RPO/RTO-mål och hur mycket data som förändras dagligen.

Fullständig (full) backup
En **komplett kopia** av allt valt innehåll varje gång.
Fördel: enklast att återställa (du behöver **en** backupuppsättning).
Nackdel: längst backupfönster, mest lagring.
Passar: små/medelstora datamängder, månatliga "baseline"-kopior, system där **enkel restore** väger tyngst.

Inkrementell backup
Efter en initial **full** lagras **endast förändringar sedan senaste backup** (full **eller** inkrementell).
Fördel: snabb daglig backup, minst lagring.
Nackdel: restore kräver **kedja**: full + **alla** inkrementella till önskad tidpunkt.
Passar: stora datamängder, smala backupfönster, när RTO kan vara längre vid full återställning.

Differentiell backup
Efter en initial **full** lagras **alla förändringar sedan senaste full** (inte sedan senaste diff).
Fördel: restore kräver **två** uppsättningar (senaste full + **senaste diff**).
Nackdel: diff-filer **växer** dag för dag tills nästa full.
Passar: miljöer där **snabb restore** prioriteras över minimalt lagringsutrymme.

Förtydligande (sammanfattning):

- *Full:* **enkel restore**, **dyrt** i tid/lagring.
- *Inkrementell:* **billig** i tid/lagring, **dyrare restore** (kedja).
- *Differentiell:* **mittemellan** – större dag för dag, men **snabb restore** (full + diff).

Exempel: *Datamängd 1 TB, daglig förändring ~5 % (= ~50 GB).*

- *Inkrementell vecka:* Sön **1 TB full**, mån–lör **~50 GB/dag** ⇒ total vecka ~1 TB + 6×50 GB ≈ **1,3 TB**. Restore till torsdag kräver **1 full + 3 inkrementella**.
- *Differentiell vecka:* Sön **1 TB full**, mån **50 GB**, tis **100 GB**, ons **150 GB** … ⇒ restore till torsdag kräver **1 full + 1 diff (150–200 GB)** – snabbare än inkrementell kedja men större dagliga backups sent i veckan.

Exempel: *Skolans filserver (Tier 0):* **Söndag full**, **mån–lör inkrementell**, off-site med **immutabel** retention 30 dagar. RTO säkras via **regelbundna restore-tester** och möjlighet till **filnivå-restore** utan att spela upp hela kedjan.

Exempel: *Administrativa dokument (Tier 1) där snabb återställning efter "misstag" är viktig:* **Söndag full**, **mån–lör differentiell**. Restore = **full + senaste diff** (färre steg i skarpt läge).

Exempel: *Klassrumsklienter:* "Golden image" hanteras separat; använd **inkrementell** för hemkataloger och kör **månatlig full** för att korta kedjorna.

149

4.4.3 Backupstrategier och återställning

Backup är din **sista försvarslinje** mot misstag, hårdvarufel och ransomware. En bra strategi börjar med två mål per system: **RPO** (*Recovery Point Objective* – hur mycket data får vi förlora?) och **RTO** (*Recovery Time Objective* – hur snabbt måste vi vara uppe igen?). Välj sedan teknik som möter målen: full/inkrementell/differentiell, **immutabla/air-gappade** kopior *(immutabla = kan inte ändras/raderas under en vald retention, "WORM"; air-gappade = kopior som är fysiskt eller logiskt isolerade från produktionsnätet, utan ständig åtkomst)* och off-site lagring. Kom ihåg: **snapshots ≠ backup** – snapshots är fantastiska före en ändring, men skyddar inte mot korruption som replikerats eller mot attacker som krypterar primär och replika.

Bra principer (förtydligande):

- **3-2-1(-1-0):** 3 kopior, 2 olika medier, 1 off-site, *(+1* ***immutabel*** *– skrivskyddad/WORM-kopia;* ***air-gap*** *innebär isolerad kopia)*, **0** fel i testad återställning.
- **Klassificera system:** Tier 0 (kritiska), Tier 1 (viktiga), Tier 2 (övriga).
- **Kryptering & nycklar:** kryptera off-site; skydda nycklar separat (MFA/HSM).
- **App-konsistens:** VSS för Windows-servrar; "quiesce" databaser.
- **Retention:** GFS (daglig/veckovis/månatlig), juridik/arkivkrav.

Exempel: *Filserver (Tier 0) på Proxmox: nattlig inkrementell, veckovis full, månatlig off-site till S3-kompatibelt objektlager med immutabilitet 30 dagar. Teståterläsning 1×/vecka (filnivå) och 1×/månad (hel VM till isolerat VLAN). RPO 24 h, RTO 4 h.*

Mall: Backupprofil per system (tom)

- **System/ägar-OU:**
- **Klass (Tier):**
- **RPO / RTO:**
- **Datakälla & metod:** *agent/VSS, konsistens*
- **Schema:** *full/incr/diff – när?*
- **Retention:** *dag/vecka/månad/år*
- **Lagring:** *on-site, off-site, immutabel, media*
- **Kryptering/nyckelhantering:**
- **Återställningstester:** *frekvens, senast utfört, resultat*
- **Kontakt/ansvarig:**
- **Relaterade runbooks:** *steg-för-steg för restore*

150

Exempel: Backupprofil (ifylld, kort)

- **System:** SRV-FIL-01 (OU Skolan\Servrar\File)
- **Tier:** 0
- **RPO/RTO:** 24 h / 4 h
- **Metod:** Agentlös VM-backup + VSS inuti gästen (app-konsistens)
- **Schema:** Full sön 02:00, inkrementell mån–lör 02:00
- **Retention:** Daglig 14 d, veckovis 8 v, månadsvis 12 m
- **Lagring:** On-site NFS (primär), Off-site S3 immutabel 30 d
- **Kryptering:** AES-256; nyckel i separat valv med MFA
- **Test:** Fil-restore varje fredag; hel VM-restore månad 1:a tisdag (senast OK 2025-08-05)
- **Ansvarig:** Nät/Server-admin
- **Runbook:** “Restore SRV-FIL-01 till isolerat VLAN”

4.4.4 Test av redundans och disaster recovery

Redundans som **inte testas** är **inte** redundans. Planera återkommande **spärrtester** där ni medvetet bryter komponenter: primär internetlänk, aktiv gateway (VRRP/HSRP = *förstahoppsredundans*: flera routrar delar en **virtuell gateway-IP**; en är aktiv **master**, övriga **standby**; vid fel tar standby över IP/MAC och klienterna märker bara ett kort avbrott), en DNS-resolver, en DHCP-server, en Proxmox-nod, en lagringslänk. Dokumentera förväntat utfall, mät faktiska tider och uppdatera konfiguration/mallar där gap hittas. På strategisk nivå behövs en **DR-plan (Disaster Recovery)**: *hur* ni startar upp det viktigaste efter ett större avbrott (brand, ransomware, långvarigt strömavbrott).

Att testa (förtydligande):

- **Gateway-failover:** byt aktiv VRRP/HSRP (VRRP = **Virtual Router Redundancy Protocol** – öppen standard. HSRP = **Hot Standby Router Protocol** – Cisco. Båda ger en virtuell gateway-IP med automatisk failover mellan noder.); verifiera att ping/trafik fortsätter (mät “hit”).
- **DNS-redundans:** stoppa ena resolvern; mät namnupplösning och TTL-effekt.
- **DHCP-failover:** stäng en server; bevisa utdelning/lease-förnyelse.
- **WLAN-kontroller/AP:** simulera kontroller-fel; klienter ska återansluta.
- **Proxmox/lagring:** dra ner en nod/länk; VM/IO ska fortsätta enligt design.
- **Internet dual-homing:** BGP/SD-WAN/backup-länk tar över; verifiera NAT/policy.

151

Exempel: *Planerat test “Gateway-failover Q3”: kl 18:00 sänks prioritet på Dist-A (VRRP). Dist-B tar över inom 1 s. Övervakning visar 1 förlorad ping på elev-VLAN, 0 på personal (SLA uppfyllt). Slutsats: höj “track” på internetporten för snabbare detektion.*

DR-plan (kärnpunkter):

- **Utlösare & roller:** vem deklarerar DR-läge, vem leder.
- **Prioritering:** Tier 0 → Tier 1 → Tier 2 (lista system i ordning).
- **Miljö:** isolerat återställnings-VLAN, temporära lösenord, säkra kanaler.
- **Data:** var finns senaste *grön-testade* backupen (hash/rapport).
- **Checklista:** återställ AD/inloggning (om tillämpligt), DNS/DHCP, fil/resurs, sedan övriga.
- **Exit-kriterier:** när lämnar vi DR-läge; post-mortem inom 5 dagar.

Exempel: *Ransomware-övning: filserver låtsas krypterad. DR-teamet isolerar servern, återställer gårdagens backup till ny VM i karantän-VLAN, kör antivirus, verifierar fil-hashar, svänger om DFS-pekare. Tid från “start DR” till “tjänst åter” =* ***2 h 40 min*** *(inom RTO 4 h).*

152

📌 Kontrollfrågor

1. Beskriv de sex stegen i en enkel incidentprocess och syftet med varje steg.
2. Vilka roller vill du alltid bemanna i en P1/P2-incident och varför?
3. Vad ska en **incidentrapport** minst innehålla för att vara användbar i efterhand?
4. Förklara RPO och RTO och ge ett konkret exempel från skolmiljö.
5. Varför räcker inte snapshots som backup, och när är de ändå värdefulla?
6. Redogör för 3-2-1(-1-0)-regeln och vad “-1-0” betyder i praktiken.
7. Vad innebär app-konsistenta backuper (t.ex. VSS) och när krävs de?
8. Lista fem redundanstester du bör köra varje termin och hur du verifierar utfallet.
9. Vad är skillnaden mellan **incidenthantering** och **disaster recovery**?
10. Nämn tre vanliga förbättringar som brukar komma ur en bra post-mortem.

153

🔗 Fördjupningslänkar

- NIST SP 800-61r2 – Computer Security Incident Handling Guide: https://csrc.nist.gov/publications/detail/sp/800-61/rev-2/final
- ISO/IEC 27035 – Information security incident management (översikt): https://www.iso.org/standard/44379.html
- CIS Controls – Incident Response & Recovery: https://www.cisecurity.org/controls
- CISA – Ransomware Guide (beredskap & återställning): https://www.cisa.gov/stopransomware
- Microsoft – Backup best practices (Windows/Server, VSS): https://learn.microsoft.com/windows-server/storage/backup
- Proxmox Backup Server – dokumentation: https://pbs.proxmox.com/docs/
- Veeam – SureBackup / testade återställningar (koncept): https://helpcenter.veeam.com/docs/backup

154

✍️ Egna anteckningar

…

155

📌 Kontrollfrågor

1. Beskriv de sex stegen i en enkel incidentprocess och syftet med varje steg.
2. Vilka roller vill du alltid bemanna i en P1/P2-incident och varför?
3. Vad ska en **incidentrapport** minst innehålla för att vara användbar i efterhand?
4. Förklara RPO och RTO och ge ett konkret exempel från skolmiljö.
5. Varför räcker inte snapshots som backup, och när är de ändå värdefulla?
6. Redogör för 3-2-1(-1-0)-regeln och vad “-1-0” betyder i praktiken.
7. Vad innebär app-konsistenta backuper (t.ex. VSS) och när krävs de?
8. Lista fem redundanstester du bör köra varje termin och hur du verifierar utfallet.
9. Vad är skillnaden mellan **incidenthantering** och **disaster recovery**?
10. Nämn tre vanliga förbättringar som brukar komma ur en bra post-mortem.

156

🔗 Fördjupningslänkar

- NIST SP 800-61r2 – Computer Security Incident Handling Guide: https://csrc.nist.gov/publications/detail/sp/800-61/rev-2/final
- ISO/IEC 27035 – Information security incident management (översikt): https://www.iso.org/standard/44379.html
- CIS Controls – Incident Response & Recovery: https://www.cisecurity.org/controls
- CISA – Ransomware Guide (beredskap & återställning): https://www.cisa.gov/stopransomware
- 3-2-1 Backup Strategy (översikter, leverantörs-agnostiskt): https://www.backupstrategy.com/3-2-1
- Microsoft – Backup best practices (Windows/Server, VSS): https://learn.microsoft.com/windows-server/storage/backup
- Proxmox Backup Server – docs: https://pbs.proxmox.com/docs/
- Veeam – SureBackup / testade återställningar (koncept): https://helpcenter.veeam.com/docs/backup

157

✍️ Egna anteckningar

…

158

5. Fördjupningsdel

5.1 Virtualisering – plattformar och gemensamma mönster

5.1.1 Varför virtualisera? (problembakgrund och mål)

Virtualisering blev standard därför att den **löser flera problem samtidigt**: hårdvaran utnyttjas bättre (fler servrar per fysisk värd), **isolation** mellan tjänster ökar driftsäkerheten, **snabb återställning** blir möjlig med snapshots/kloner/backup, och **automation** (skript, mallar, "infrastruktur som kod") gör att du kan bygga och återskapa miljöer på minuter istället för dagar. För skola och SMB betyder det att labbar och skarp drift kan leva sida vid sida, att uppgraderingar testas **innan** de rullas ut, och att enskilda fel inte behöver ta ner hela miljön. Virtualisering förenklar också **disaster recovery** – när servrar är filer/diskavbilder kan de flyttas och startas på annan hårdvara.

Vanliga drivkrafter: konsolidering och kostnad, snabb provisionering, enklare DR/backup, bättre säkerhetszonering, samt möjligheten att **skala och ändra** utan att byta fysisk hårdvara varje gång.

***Exempel (Exempel:**) Inför nationella prov körs en separat provtjänst i egen VM. Efter provperioden tas en snapshot för arkiv, och resurserna (CPU/RAM) återgår till ordinarie drift.*

5.1.2 Gemensam arkitektur (oavsett plattform)

Alla moderna plattformar delar samma byggstenar: **hypervisor** (kärnan som kör VM:er), **virtuella switchar/bridges** som kopplar VM:er mot nätet, **lagring** (lokal eller delad), och **hantering** (webb-UI/CLI/API). Du arbetar med **mallar** (golden images), **nätprofiler** (VLAN/portgrupper), samt **policies** för resurser och säkerhet. Två viktiga principer:

1. **Snapshots ≠ backup** (snapshot är kortlivad säkerhetslina; backup är separata, helst immutabla/air-gappade kopior).
2. **Segmentera** trafik: management, lagring, backup och produktion i egna VLAN/subnät och med tydliga brandväggsregler.

***Exempel (Exempel:**) Management-VLAN nås bara via bastion/jump host; backupflöden går i separat VLAN så att en överbelastad applikationslänk inte stör backupfönstret.*

5.1.3 Plattformerna (bakgrund, styrkor och begränsningar)

Proxmox VE (KVM/QEMU + LXC, Debianbaserad)
Proxmox bygger på **Debian Linux** och använder KVM/QEMU för fullvirtualisering (bra för Windows Server) samt **LXC** för lätta Linux-containrar. Styrkorna är **öppen källkod**, snabb utveckling, **VLAN-aware Linux bridges**, **ZFS**-stöd, klusterhantering och **Proxmox Backup Server** (dedupe, verifiering, immutability-policy). Passar skolor/labbar/SMB som vill ha kraftfullt och kostnadseffektivt utan licenslåsning. Begränsningar: kräver Linux-vana, färre "färdigpaketerade" enterprise-ekosystem än vSphere.

159

VMware vSphere/ESXi (proprietär standard i många företag)
vSphere består av **ESXi** (hypervisor) och **vCenter** (central hantering). Styrkorna är **mogen drift**, enormt ekosystem (backup, övervakning), **vMotion/DRS** (liveflytt/lastbalansering) och **Distributed vSwitch**. Lämpligt när du behöver beprövad, stor skala och certifierat stöd från många leverantörer. Begränsningar: licenskostnad, beroende av VMware-ekosystemet; avancerade funktioner kräver rätt edition.

Microsoft Hyper-V (Windowsintegrerad hypervisor)
Hyper-V är integrerad i **Windows Server** (roll) och har **virtual switch**, **VHDX**, **Checkpoints** och bra **PowerShell**-stöd. Styrkor: tajt integration med Windows, AD/GPO, SMB3/Storage Spaces, och gott stöd i många backupverktyg. Lämpligt om ni redan är "Windows-tunga" och vill hålla verktygen enhetliga. Begränsningar: mindre ekosystem för Linux-containers, vissa funktioner kräver Datacenter-licens; Microsofts fristående Hyper-V Server är utfasad (fokuset ligger på Windows Server-rollen).

XCP-ng (Xen-baserad, öppen)
XCP-ng bygger på **Xen** och erbjuder enterprise-känsla i open-source-tappning, med **Xen Orchestra** för hantering/backup (GUI/SaaS eller self-hosted). Styrkor: bra balans mellan enkelhet och kapacitet, stabil hypervisor, vettiga snapshot/backupflöden, och aktiv community. Begränsningar: färre "plug-and-play"-integrationer än VMware, viss inlärningskurva kring Xen-terminologi.

libvirt/KVM "ren" (t.ex. virt-manager/terraform-libvirt)
Det "rena" KVM/libvirt-spåret ger maximal kontroll (nära Linux), med **bridge/OVS**, **cloud-init**, och automation via Ansible/Terraform. Styrkor: **full transparens**, inga licenser, perfekt för djup Linuxlabb. Begränsningar: kräver mest **Linux-kompetens**; färre bekvämligheter out-of-the-box för större team.

***Exempel (Exempel:**) *En kommunal IT-avdelning med Windows-tyngd kör Hyper-V för skarp drift (AD, fil, print) men använder Proxmox för kurslabb och snabb prototypning. En partner kör vSphere för "tunga" produktionstjänster med höga SLA, och XCP-ng på filialkontor.*

5.1.4 Nätmönster: bridge, trunk, VLAN och NAT

Oavsett plattform möter du tre mönster:
Bridge till produktion (VLAN-aware): uplink som **trunk** från switchen; skapa portgrupp/bridge som *bär flera VLAN* och tilldela **VLAN-ID** per VM-interface.
Isolerat NAT-nät för labb: en virtuell switch/bridge med NAT mot värden eller särskild edge-VM; perfekt för att låta elever labba utan att röra produktion.
Dedikerad management/backup/storage-väg: separata nät för admin, backup och lagring (t.ex. iSCSI/NFS) för prestanda/säkerhet.

***Exempel (Exempel:**) *En webb-VM i DMZ får vNIC på VLAN 70; brandvägg släpper endast 80/443 in och 443 till intern API-server. En andra vNIC på VLAN 99 används enbart för backupflöden till backupservern.*

160

5.1.5 Lagring, snapshots och backup

Snapshots fryser VM-disk (och ev. minne) på primär lagring; de är perfekta inför ändringar men ska **rensas** efter test. **Backup** skapar en **separerad kopia** (helst **immutabel** och/eller **air-gappad**) som inte påverkas om primär lagring skadas eller krypteras.
Proxmox + **Proxmox Backup Server** ger dedupe/komprimering/verifiering. vSphere har stort stöd i verktyg som Veeam. Hyper-V fungerar med VSS och Windows-/tredjepartsbackup. XCP-ng har XO/XOA-backup. Libvirt/KVM kan använda borg/restic/rsync-strategier och "agent-i-gäst" för app-konsistens.

***Exempel (Exempel:**) Inför patch: snapshot → patch → test → rensa snapshot; ordinarie nattbackup (full + inkrementell) fortsätter och **restore-test** körs varje månad i isolerat VLAN.*

5.1.6 Hög tillgänglighet, kluster och DR

Kluster låter dig hantera flera noder som en enhet (gemensam vy, policyer). **Live migration** flyttar VM utan avbrott vid underhåll. **HA** startar automatiskt om VM:er på annan nod vid fel. För **DR** (större avbrott) behövs replikerad backup, dokumenterad återstartsordning (Tier $0 \rightarrow 1 \rightarrow 2$) och regelbundna **spärrtester** (se kapitel 4.4).

***Exempel:** Kvällsunderhåll: vMotion/Live Migration flyttar VM bort från noden som ska patchas. Under driftfel startar HA upp kritiska VM på frisk nod; användarna märker en kort paus.*

5.1.7 Resurser och licenser (CPU, RAM, disk, Windows Server)

Allokera resurser **konservativt** först och mät. **Overcommit** kan fungera för CPU men var försiktig med RAM och I/O-tunga diskar. Planera **nätprofiler** (vNIC-antal/roller) och håll dig till namngivning/mallar för spårbarhet.
Tänk även på **licenser**: Windows Server **Datacenter** ger rätt att köra obegränsat antal Windows-VM:er på licensierad hårdvara; **Standard** ger rätt för ett mindre antal VM:er per licens. Detta kan styra var du placerar Windows-tunga arbetslaster.

***Exempel:** Filserver-VM: 2 vCPU, 8–12 GB RAM (6–8 GB garanterat), virtio-SCSI + iothread, två vNIC (produktion/backup). Justera efter mätvärden i övervakningen.*

5.1.8 Säkerhet och driftetikett

Separera **management** från **produktion**, använd **MFA** och **bastion/jump host** för adminåtkomst, och logga alla förändringar. Håll **brandväggspolicy** mellan VLAN/zoner även i labb. Vid WLAN-anslutna klienter som når VM-miljön: säkerställ rollbaserade brandväggsregler och övervaka ovanliga flöden.

***Exempel:** Admin når hypervisorer via VPN → bastion i management-VLAN → SSH/RDP vidare till värdar/VM. Inga öppna adminportar mot internet.*

161

5.1.9 Jämförelsetabell (styrkor, när använda, begränsningar)

Proxmox (Debian)

- **Nätmodell (VLAN/NAT):** Linux bridge (VLAN-aware), trunk på uplink, NAT-bridge för labb, bond/LACP.
- **HA/Live migration:** Kluster med Proxmox HA; live migration (delad lagring eller replikering).
- **Backup-ekosystem:** Proxmox Backup Server (dedupe, verifiering, immutability-policy) + 3:e-partsverktyg via standardprotokoll.
- **Styrkor:** Öppen källkod, kostnadseffektiv, LXC + KVM, ZFS-stöd, starkt API/community.
- **Begränsningar:** Kräver mer Linuxvana; färre "stora" certifierade integrationer än vSphere.
- **Lämplighet:** Skola/SMB/labb; mixad Win/Linux; där budget, flexibilitet och transparens väger tungt.

VMware vSphere/ESXi

- **Nätmodell (VLAN/NAT):** vSwitch/Distributed vSwitch, port groups (VLAN-ID), NAT via Edge-/gateway-VM eller NSX.
- **HA/Live migration:** Mycket moget: vMotion/Storage vMotion/DRS/HA.
- **Backup-ekosystem:** Stort (VADP-stöd); Veeam m.fl. med bred funktionalitet.
- **Styrkor:** Mognad, skala, brett ekosystem, leverantörsstöd/certifieringar.
- **Begränsningar:** Licenskostnader och editioner; viss inlåsning i ekosystemet.
- **Lämplighet:** Större/kritiska miljöer med hårda SLA och krav på certifierade integrationer.

Microsoft Hyper-V (Windows)

- **Nätmodell (VLAN/NAT):** Virtual Switch med VLAN-ID, NIC Teaming; NAT via RRAS/Windows NAT eller gateway-VM.
- **HA/Live migration:** Failover Clustering, Live/Storage Migration.
- **Backup-ekosystem:** VSS-integration; Windows Server Backup, Veeam m.fl.
- **Styrkor:** Tajt Windows/AD-integration, PowerShell/automation, låg tröskel i MS-miljö.
- **Begränsningar:** Vissa funktioner kräver Datacenter-licens; mindre fokus på Linux/containers; fristående Hyper-V Server utfasad.
- **Lämplighet:** Windows-tung drift (AD, fil/print, RDS); skolor/organisationer med MS-stack.

XCP-ng (Xen)

162

- **Nätmodell (VLAN/NAT):** Networks med VLAN-taggning; NAT via gateway-VM (t.ex. OPNsense).
- **HA/Live migration:** Live migration/HA med delad lagring; orkestreras via Xen Orchestra (XO/XOA).
- **Backup-ekosystem:** Inbyggt i XO/XOA (delta, replikering) + 3:e-partsverktyg.
- **Styrkor:** Open source med "enterprise-känsla", enkel drift, bra GUI (XO).
- **Begränsningar:** Mindre ekosystem än VMware; vissa features beroende av XO.
- **Lämplighet:** Filial/SMB; behov av HA/backup utan stora licenskostnader.

libvirt/KVM "ren"

- **Nätmodell (VLAN/NAT):** Linux bridge/OVS, trunking; NAT via `virbr0`/dnsmasq eller egen brandvägg.
- **HA/Live migration:** Möjligt (Pacemaker/Corosync/virt-tools), men mer "bygg själv".
- **Backup-ekosystem:** DIY: borg/restic/rsync + `qemu-img`/virsh; app-konsistens via agenter/skript.
- **Styrkor:** Maximal kontroll, inga licenser, ideal för automation/DevOps/labb.
- **Begränsningar:** Mest handpåläggning och Linuxkompetens; färre bekvämligheter för större team.
- **Lämplighet:** Labb/utbildning, CI/CD, speciallösningar där full kontroll prioriteras.

5.1.10 Snabba "recept" per plattform

Proxmox (Debian-bas):
VM på rätt VLAN: Trunka uplink, sätt `vmbr0` **VLAN-aware**, ge vNIC **VLAN-tagg**.
NAT-labb: Skapa `vmbrN` med NAT; lägg kurs-VM:er där.
Snapshot + rollback: Snapshot → ändra → testa → rensa.
Enklast backup: PBS-target, daglig inkrementell + månatlig verifiering.

VMware vSphere/ESXi:
VM på rätt VLAN: Skapa **port group** med VLAN-ID, koppla vNIC dit.
NAT-labb: Använd Edge-VM (pfSense/OPNsense) eller NSX-Edge för NAT.
Snapshot + rollback: Snapshot via vCenter; rensa efter test.
Enklast backup: Veeam-jobb mot vCenter, GFS-retention, regelbunden restore-test.

Hyper-V (Windows):
VM på rätt VLAN: **Virtual Switch** + VLAN-ID på vNIC.
NAT-labb: Intern switch + Routing/NAT-roll i en liten gateway-VM.
Checkpoint + rollback: Checkpoint före ändring; "Apply"/Delete när klar.
Enklast backup: Windows Server Backup eller Veeam med VSS-konsistens.

XCP-ng:
VM på rätt VLAN: Skapa network med VLAN-tagg, koppla vNIC.
NAT-labb: Gateway-VM för NAT (t.ex. OPNsense).
Snapshot + rollback: Via XO/XOA, rensa när klart.
Enklast backup: XO-backup (delta), regelbunden verifiering.

libvirt/KVM:
VM på rätt VLAN: Linux bridge/OVS + trunk uthamnad; vNIC med VLAN-tagg.
NAT-labb: `virbr0` (default NAT) eller egen `dnsmasq`/firewalld-NAT.
Snapshot + rollback: `qemu-img`/virsh snapshot.
Enklast backup: stoppa-konsistens + rsync/borg/restic; för app-konsistens använd agent/scripting.

164

📌 Kontrollfrågor

1. Ge tre skäl till att virtualisera som direkt påverkar tillgänglighet och arbetstakt.
2. Förklara skillnaden mellan **snapshot** och **backup** och när respektive används.
3. Varför bör management, lagring, backup och produktion separeras nätmässigt?
4. När väljer du Proxmox framför vSphere – och tvärtom?
5. Vad talar för Hyper-V i en Windows-tung skolmiljö?
6. Hur sätter du en VM på rätt VLAN i vSphere, Hyper-V respektive Proxmox (kort steg för steg)?
7. Vad innebär **overcommit** och vilka risker finns för RAM och I/O?
8. Nämn två sätt att skapa ett **isolerat NAT-labb** utan att påverka produktion.
9. Vilka komponenter behövs för **HA + live migration** i praktiken?
10. Hur skulle du licensiera Windows-tunga VM-värdar kostnadseffektivt (Standard vs Datacenter)?

165

🔗 Fördjupningslänkar

- Proxmox VE docs: https://pve.proxmox.com/pve-docs/
- VMware vSphere docs: https://docs.vmware.com/
- Microsoft Hyper-V docs: https://learn.microsoft.com/windows-server/virtualization/hyper-v/
- XCP-ng & XO: https://xcp-ng.org/ och https://xen-orchestra.com/
- libvirt/KVM: https://libvirt.org/ och https://qemu.readthedocs.io/
- Linux bridge/OVS grunder: https://wiki.linuxfoundation.org/networking/bridge

166

✍️ Egna anteckningar

...

167

5.2 Fördjupning: Windows Server och GPO

5.2.1 Vad är GPO – och var börjar man?

Tänk **Group Policy (GPO)** som en central **regibok** för Windows-klienter och -servrar. Du beskriver hur datorer och användarsessioner ska se ut och bete sig, och **domänen** ser till att det händer — automatiskt och konsekvent. Det kan vara allt från skrivbordsdetaljer och genvägar till säkerhet, kryptering, brandvägg, programvaror och uppdateringar. Vitsen är **förutsägbarhet**, **spårbarhet** och **fart**: samma in, samma ut, utan handpåläggning på varje maskin.

Du arbetar i **Group Policy Management Console (GPMC)**. Där ser du skog → domän → **Group Policy Objects** (själva "dokumenten") och **OU:er** (behållare där du **länkar** dem). Öppnar du en GPO möts du av två huvuddelar:

- **Computer Configuration** (gäller datorn oavsett vem som loggar in).
- **User Configuration** (gäller användarsessionen på den datorn).

En bra start är att skapa en enkel GPO (t.ex. standardbakgrund), **länka** den till en test-OU och se hur inställningen rullar ut. För att alla admins ska se samma policyspråk och versioner lägger man **administrativa mallar** (ADMX/ADML) i en **Central Store** i SYSVOL. Då blir det lättare att samarbeta och undvika "det såg inte ut så på min dator".

Exempel: *Skapa "GPO-Desktop-Standard". Sätt ett neutralt skrivbordsunderlägg och dölja vissa kontrollpaneler. Länka till en test-OU med några datorer. Vid nästa uppdatering får användarna samma skrivbordsmiljö — inga manuella klick.*

5.2.2 Hierarki och utvärdering – vem "vinner" när policies krockar?

Windows tillämpar policy i ordningen **Lokal → Site → Domain → OU** (L-S-D-O-U). Varje steg kan lägga till eller skriva över inställningar. I en och samma behållare styr även **Link Order**: länken som ligger **sist** kör **sist** — och vinner vid krock.

Två "brytare" påverkar arvet:

- **Enforced (No override):** tvingar igenom en länkad GPO nedåt även om någon längre ner försöker skriva över den.
- **Block Inheritance:** stoppar arv från ovan — men **Enforced** går igenom ändå.

Särfallet **Loopback Processing** gör att **användarinställningar** kan tas från **datorns OU**. Det är ovärderligt för delade maskiner, kiosker och VDI.

- **Merge:** användarpolicyn kombineras; datorns OU kan överstyra detaljer.
- **Replace:** användarpolicyn kommer enbart från datorns OU.

Exempel: *En domänlänkad GPO "Säkerhetsbas" sätter standardregler. I en viss server-OU finns "RDP-Tilldelning". Om de krockar vinner OU-länken — om inte "Säkerhetsbas" är* ***Enforced****.*

Bakgrundsnot: Klienter uppdaterar i bakgrunden med jämna mellanrum. Viss funktionalitet (t.ex. programvaruinstallation) sker dock vid **start/inloggning**. `gpupdate /force` triggar en omedelbar uppdatering, men kräver ibland omstart för att allt ska slå igenom.

5.2.3 Från lätt till tungt – vad kan du styra med GPO?

Börja med **det synliga** — det ger snabb bekräftelse på att kopplingar och arv fungerar — och bygg sedan upp mot säkerhet och drift.

- **Administrative Templates:** färdiga policyer som i grunden skriver välkända registernycklar. Exempel: skrivbordsunderlägg, kontrollpaneler, webbläsarpolicyer, Onedrive/Office-inställningar.
- **Group Policy Preferences (GPP):** lägger till filer, mappar, genvägar, drivmappar och skrivare. Med **Item-Level Targeting** kan du villkora per **grupp**, **OU**, **OS-version**, **plats**, **IP-område**, **datornamn** m.m. Resultatet blir samma precision som skript — men mer robust.
- **Security Filtering:** låt bara en viss **säkerhetsgrupp** påverkas av GPO:t.
- **WMI-filter:** träffa t.ex. bara klient-OS, eller bara bärbara (har batteri). Använd sparsamt — många eller tunga filter ökar inloggningstid.

Exempel: *En GPP-drivmapp syns bara för gruppen “Ekonomi”* ***och*** *endast på datorer i “OU=Ekonomi”. Ingen annan påverkas, även om GPO:t är länkat högre upp.*

5.2.4 Skript via GPO – när är Startup/Logon rätt, och när är Schemalagd uppgift bättre?

Startup/Shutdown körs i **dator-kontekst** (före/efter inloggning) som **Local System**. Perfekt för maskinnära saker: registry, tjänster, protokoll, filkopior som kräver admin.
Logon/Logoff körs i **användarsession** och passar profilnära uppgifter: genvägar, små städjobb, användar-specifika registerjusteringar.

Vill du ha **retry, fördröjning, villkor, tidsgränser** och bättre loggning använder du **Scheduled Tasks via GPP**. Kör som **SYSTEM** eller **tjänstkonto**, trigga “At startup”, “At logon” eller vid schema. Lägg skript i **SYSVOL** (gemensamt, versionsstyrt) och styr **Execution Policy**/kodsignering i GPO. Tänk på att nätverksresurser kanske inte finns tillgängliga precis i uppstart — vill du garantera det aktiverar du policyn *Always wait for the network at computer startup and logon* i en bas-GPO.

Exempel: *Ett Startup-skript stänger av SMBv1 och sätter NTP-källa. En Schemalagd uppgift (SYSTEM) kör varje natt och rensar temporära mappar; resultatet loggas i Event Viewer.*

Litet skript-exempel (Startup, körs som SYSTEM)
Exempel: *Lägg i* `\\<domän>\SYSVOL\<domän>\Scripts\Startup\Hardening.ps1` *och kör via* ***Computer Configuration → Policies → Windows Settings → Scripts (Startup/Shutdown)*** *eller som* ***Scheduled Task (At startup)****.*

```
# Hardening.ps1 - körs som Local System vid startup
```


169

```
# 1) Stäng av SMBv1 (klientkomponent)
Try { Disable-WindowsOptionalFeature -Online -FeatureName SMB1Protocol -
NoRestart -ErrorAction Stop } Catch { }

# 2) Stäng av SMBv1 (serversidan)
Try { Set-SmbServerConfiguration -EnableSMB1Protocol $false -Force } Catch { }

# 3) Sätt NTP-källa (ersätt med er tidskälla)
$ntp = 'time.example.org,0x8'
w32tm /config /syncfromflags:manual /manualpeerlist:$ntp /update | Out-Null
w32tm /resync /force | Out-Null

# 4) Logga enkel status
$msg = "Hardening klar: SMBv1 avstängd, NTP=$ntp"
Write-EventLog -LogName Application -Source "Windows PowerShell" -EntryType
Information -EventId 1001 -Message $msg
```

5.2.5 Programvarudistribution – strategier, val och fallgropar

Utmaningen: Program kommer som **MSI**, **EXE** eller **MSIX**. De har olika tysta switchar, kräver olika **detektion** (hur vet vi att appen redan finns?), har beroenden (drivrutiner, VC++-omfördelningsbara), och vissa kräver **omstart**. En bra strategi ger: spårbarhet (loggar/status), **piloter** (testgrupper), **ringar** (stegvis utrullning) och **rollback**.

A) GPO + MSI (enkelt och stabilt)
Med **Software Installation** i GPO kan du **Assign to Computer** ett **MSI**. Installationen sker vid start, innan inloggning, vilket är robust. Lägg källan på **UNC-share** med läsrätt för datorer. Anpassa via **MST** (transform). Begränsningar: främst MSI, svagt för EXE, enkel rapportering.

B) GPO + skript/GPP (för EXE och specialfall)
Kör EXE med tyst flagga (`/quiet`, `/S`, `/VERYSILENT`) via **Startup-skript** eller **Scheduled Task** (SYSTEM). Bygg in **detektion** (kontrollera fil/reg-nyckel, kör inte igen om det redan finns) och **logga** exit-koder. Bra för mindre/medel miljö — men standardisera loggformat, uninstall och rollback.

C) Intune (MDM) – ringar, självbetjäning och internetnoder
Paketering till **.intunewin**, definiera **install/uninstall**-kommandon, **detektionsregler** och **beroenden**. Kör mot användare eller enheter, gör **Required** (tvingad) eller **Available** i portal. Styrka: ringar, självbetjäning, funkar utanför LAN.

D) Configuration Manager (SCCM/MECM) – djup kontroll on-prem
Full kontroll över **applikationer**, **samlingar**, **phased deployments**, detaljerad status och rapporter. Passar större on-prem-miljöer med höga krav på precision och uppföljning.

Gemensamma råd:

- Ha en **paketstandard** (katalognamn, logg, detektion, uninstall).
- Skilj **installationskonto** från **detektionslogik**; detection ska vara snabb och pålitlig.
- Undvik “eviga” installationsförsök genom att **bryta** på tydliga exit-koder.

170

- Planera **omstarter** (t.ex. via deadlines eller nattfönster).
- Kör **pilot** först; utvärdera innan du breddar.

Hands-on: MSI via GPO (Assigned to Computer)

1. Lägg `App.msi` (ev. `App.mst`) på `\\fileserver\software\App\` och ge **Read** för *Domain Computers*.
2. Skapa GPO "GPO-Apps-App".
3. Öppna GPO → **Computer Configuration → Policies → Software Settings → Software Installation** → *New* → *Package…* → peka på **UNC** → **Assigned**.
4. (Valfritt) Lägg `App.mst` under *Modifications*.
5. **Länka** GPO:t till mål-OU (datorer).
6. Sätt *Always wait for the network at computer startup and logon* = **Enabled** i bas-GPO.
7. Testa: `gpupdate /force`, **omstart**, verifiera i loggar/Program och funktioner.
8. Uppgradera via *Upgrades*-fliken, eller ta bort med **Remove** (ev. *Immediately uninstall*).

Exempel: *"GPO-Apps-Reader" installerar Adobe Reader tyst från* `\\fileserver\software\Reader\AcroRead.msi`*, med* `AcroRead.mst` *för språk och avstängd auto-update.*

5.2.6 Centrala uppdateringar – WUfB, WSUS och ringar

Målet är att få en trygg rytm: små säkra steg ofta, och stora steg när du är redo — med **ringar** som fångar fel tidigt.

- **Windows Update for Business (WUfB):** Klienter hämtar från Microsoft Update, men du styr **deferral** (hur länge de väntar), **deadlines** (när de måste installera/boota) och **ringar** (Pilot → Bred → Sen). Fungerar utmärkt för enheter som inte alltid är på LAN.
- **WSUS:** Du **godkänner** uppdateringar centralt och styr vilka **datorgrupper** som får vad — när. Bra när du behöver lokal cache, strikt kontroll eller avskilda nät.

Tänk i två flöden:

- **Quality Updates** (månatliga säkerhetsfixar): rulla snabbt genom ringar med kort deferral.
- **Feature Updates** (större OS-lyft): längre deferral och tydligare pilot.

Glöm inte **Delivery Optimization** så klienter delar paket i LAN; det sparar internetkapacitet. Bestäm hur du hanterar **drivrutiner**, **.NET** och **Microsoft 365 Apps** (de kan ha egna kanaler).

Hands-on: WUfB ring-setup (Pilot → Bred → Sen)
A. Grupper och GPO:n

1. Skapa tre **säkerhetsgrupper**: `WUfB-Pilot`, `WUfB-Bred`, `WUfB-Sen`.

171

2. Skapa tre **GPO:n**: “WUfB-Pilot”, “WUfB-Bred”, “WUfB-Sen”. Länka dem högt upp, men låt **Security Filtering** styra att rätt grupp får rätt GPO.

B. Policys (Computer Configuration → Administrative Templates → Windows Update)

1. **Select when Feature Updates are received** = **Enabled** → *Deferral days*: Pilot **0**, Bred **7–14**, Sen **21–28**.
2. **Select when Quality Updates are received** = **Enabled** → *Deferral days*: Pilot **0**, Bred **2–7**, Sen **10–14**.
3. **Specify deadlines for automatic updates and restarts** = **Enabled** (sätt rimliga deadlines och grace).
4. **Configure Automatic Updates** = **Enabled** (auto-nedladdning och schemalagd install).
5. **Delivery Optimization**: sätt läge för LAN-delning.
6. Har ni haft **WSUS** tidigare? Säkerställ att intranät-WSUS **inte** längre pekas ut i dessa GPO:n (WUfB och WSUS samtidigt skapar konflikt).

C. Rulla

1. `gpupdate /force` på Pilotenheter. Kontrollera att inställningarna syns på Windows Update-sidan.
2. Följ upp status med dina verktyg (oavsett om det är Intune, SCCM eller egen inventering).
3. När piloten är grön: flytta fler enheter in i **Bred**. Upprepa mot **Sen**.

Exempel: *Pilot har 0 dagars deferral och 7 dagars deadline. Bred har 14/14. Sen har 28/21. Om något fallerar i Piloten pausar du ringen eller höjer deferral i Bred/Sen tills fix finns.*

5.2.7 Felsökning – se vad som faktiskt gäller

Börja alltid med **vad klienten tror är sant**:

- **`gpresult /h rapport.html`** visar **vilka GPO:er** som tillämpats (för dator och användare), vilka som filtrerats bort (säkerhet/WMI) och varför.
- **RSOP.msc** ger en snabb översikt av “Resultant Set of Policy”.
- **Event Viewer → Microsoft → Windows → GroupPolicy/Operational** avslöjar timing och fel.
- I **GPMC** kan du **modellera** förväntat utfall (vilka GPO som *bör* gälla) och jämföra med **Results** från en faktisk dator.

Vanliga bovar:

- **Fel OU/ingen länk** → klienten nås inte av GPO:t.
- **Security Filtering** träffar inte datorn/användaren.

172

- **WMI-filter** för snävt eller långsamt.
- **Nätverk inte klart vid boot** → programvaruinstall missar (aktivera “Always wait for the network…”).
- **AD-replikering** släpar → testa på DC där klienten hämtar policy.
- **SYSVOL-behörigheter**/delningsproblem → MSI-källa når inte fram.

Exempel: *En skrivare mappas inte.* `gpresult` *visar att GPO:t filtrerats bort av ett WMI-filter för “Bärbar”. Datorn är stationär. Lösning: flytta villkoret till* ***Item-Level Targeting*** *(OU/grupp) eller justera WMI-villkoret.*

173

📌 Kontrollfrågor

1. Vad vinner du på att styra Windows via GPO istället för att manuellt konfigurera varje maskin?
2. Förklara **L-S-D-O-U** och hur **Enforced** respektive **Block Inheritance** påverkar utfallet.
3. När väljer du **Loopback (Merge/Replace)** och hur påverkar det användarpolicyn?
4. Ge ett exempel där du kombinerar **Administrative Templates**, **GPP** och **Item-Level Targeting**.
5. När är **Startup/Logon-skript** rätt val, och när är **Scheduled Task via GPP** bättre?
6. Beskriv ett komplett **MSI via GPO**-flöde och hur du planerar **uppgradering/rollback**.
7. Hur hanterar du **EXE-appar** centralt med detektion och loggning — utan tredjepartsverktyg?
8. Vad skiljer **WUfB** från **WSUS**, och när väljer du det ena framför det andra?
9. Hur designar du **ringar** (Pilot → Bred → Sen) för både **Quality** och **Feature** Updates?
10. Bygg en enkel felsökningskedja när ett GPO inte tillämpas (vilka verktyg, i vilken ordning, och varför).

174

🔗 Fördjupningslänkar

- Group Policy – översikt: `https://learn.microsoft.com/windows-server/identity/ad-ds/manage/group-policy/group-policy-overview`
- Group Policy processing (LSDOU, Enforced/Block): `https://learn.microsoft.com/windows-server/identity/ad-ds/manage/group-policy/group-policy-processing`
- Loopback processing: `https://learn.microsoft.com/troubleshoot/windows-server/group-policy/loopback-processing-of-group-policy`
- Group Policy Preferences & Item-Level Targeting: `https://learn.microsoft.com/windows-server/identity/ad-ds/manage/group-policy/group-policy-preferences`
- WMI-filter – skapa och använda: `https://learn.microsoft.com/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/jj717288(v=ws.11)`
- Installera program via GPO (MSI): `https://learn.microsoft.com/troubleshoot/windows-server/group-policy/use-group-policy-to-install-software`
- Intune – Win32 app management: `https://learn.microsoft.com/intune/intune-service/apps/apps-win32-app-management`
- Windows Update for Business – konfigurera: `https://learn.microsoft.com/windows/deployment/update/waas-configure-wufb`
- WSUS – kom igång: `https://learn.microsoft.com/windows-server/administration/windows-server-update-services/get-started/windows-server-update-services-wsus`
- gpresult – visa effektiv policy: `https://learn.microsoft.com/windows-server/administration/windows-commands/gpresult`
- Central Store för ADMX/ADML: `https://learn.microsoft.com/troubleshoot/windows-client/group-policy/create-and-manage-central-store`

175

✍️ Egna anteckningar

...

176

5.3 Fördjupning: Linux-server och policy-styrning

5.3.1 Varför Linux på server?

Om man vill bygga stabila, resurssnåla och flexibla serversystem är Linux ofta ett naturligt val. Man får stor kontroll över vilka tjänster som körs, kan automatisera nästan allt och skala från en liten VM till ett helt kluster. Licensmodellen (öppen källkod eller abonnemang) gör också att ekonomi och support kan anpassas efter verksamheten. Tanken är att beskriva hur systemet **ska** se ut och uppföra sig, och sedan låta verktygen upprepa det — varje gång, på varje server.

Exempel: *En gemensam "bas" med samma SSH-policy, brandvägg och loggning används för flera servrar. Varje maskin får sedan sin roll (webb, fil, proxy), men grundbeteendet är likadant och enkelt att återskapa.*

5.3.2 Grunden: distribution, paket, konton och baspolicy

Om man är ny och ovan vid att sätta upp Linuxservrar är det lämpligt att välja en **LTS-distribution** (t.ex. Ubuntu Server LTS eller Debian Stable; Rocky/Alma om man vill ha RHEL-kompatibilitet). LTS brukar innebära längre säkerhetsstöd och lugnare förändringstakt. Paket hanteras typiskt via **APT** (Debian/Ubuntu) eller **DNF** (RHEL-familjen); många upplever att det blir tryggast att i första hand hålla sig till distributionens egna arkiv.

När det gäller **konton och rättigheter** börjar många med vanliga användarkonton och låter administratörsrätt gå via **sudo**. Det kan kännas tryggt att logga förhöjda kommandon, så att man i efterhand kan se vad som körts och av vem. Åtkomsten sker ofta via **SSH**; då väljer många att använda **nycklar** i stället för lösenord och, där riskbilden motiverar det, komplettera med tvåfaktor via PAM.

Baspolicy i praktiken: tidsynk (chrony eller systemd-timesyncd), en enkel brandvägg (ufw eller firewalld), ett MAC-skydd (AppArmor eller SELinux), central loggning (rsyslog) och förutsägbar loggrotation. Det är inte komplicerat — men gör stor skillnad över tid.

Exempel: *En nysatt server får ett administratörskonto med sudo, SSH-nyckelinloggning, brandvägg som bara öppnar 22/80/443, tidsynk mot intern NTP, och loggar som skickas till en central loggtjänst.*

5.3.3 Nätverk på Linux: IP, VLAN, bond och brygga

När du ska konfigurera nätverket i en Linuxserver är det viktigt att man håller sig till **ett** konfigurationssätt (netplan/systemd-networkd, NetworkManager eller distributionsspecifika filer) och inte blandar. Det underlättar felsökning och minskar risken för konflikter där flera komponenter försöker göra samma sak. I serverdrift handlar nätet ofta om några återkommande byggstenar: **statisk IP** med gateway/DNS, **VLAN** för segmentering, **bonding/LACP** för redundans/kapacitet, och ibland en **bridge** om värden ska husera KVM-VM:er eller containrar.

177

Exempel: *En webb-VM i DMZ (VLAN 70) pratar bara med reverse-proxyn. Backuptrafik går på ett andra interface i VLAN 99 till backupservern — skilt från produktionstrafiken.*

Så här kan en minimal netplan-definition se ut (statisk IP + VLAN):

```
# /etc/netplan/01-server.yaml
network:
  version: 2
  ethernets:
    eno1:
      dhcp4: no
  vlans:
    vlan40:
      id: 40
      link: eno1
      addresses: [192.0.2.10/24]
      gateway4: 192.0.2.1
      nameservers:
        addresses: [192.0.2.53, 192.0.2.54]
```

5.3.4 Vanliga tjänster: SSH/SFTP, SMB/NFS, webb och certifikat

I många miljöer börjar man med **SSH/SFTP** för administration och filöverföring. Det är vanligt att stänga av lösenordsinloggning, styra vilka användare eller grupper som får logga in och låta åtkomst gå via en bastionvärd i stället för direkt från internet. För filresurser väljer många **Samba/SMB** när Windows-klienter ska ansluta och **NFSv4** när det är Linux-till-Linux — båda kan använda POSIX-ACL för finmaskiga rättigheter. För webb är **Nginx** ett vanligt val, ofta som **reverse proxy** framför applikationer, med automatiserad hantering av TLS-certifikat (t.ex. via ACME/certbot) så att förnyelser sköts utan handpåläggning.

Exempel: *En publik webbapp kör bakom Nginx. Endast 80/443 är öppna i brandväggen. SSH är åtkomligt enbart från bastion, inte från internet i stort.*

5.3.5 Central identitet och “policy”: AD/SSSD, FreeIPA och polkit

Om man vill slippa separata lokala konton på varje server går det fint att koppla Linux till **central identitet**. Med **realmd/SSSD** kan system autentisera mot **AD** och tilldela sudo-rättigheter till vissa grupper. I Linuxvärlden finns också **FreeIPA**, som kombinerar LDAP, Kerberos, certifikat och host-baserad åtkomstkontroll. Lokalt på varje maskin handlar det ofta om **sudoers** och **polkit** (vad en användare får göra när en tjänst ber om höjda privilegier), samt **PAM** för hur inloggningar ska gå till.

Exempel: *Gruppen “ops-admin” i AD får sudo på alla Linux-servrar (styrt via SSSD och en enkel fil i* `/etc/sudoers.d/`*). Gruppen “ops-ro” får bara läsa journalen och köra ett fåtal diagnostikkommandon.*

5.3.6 “Policy som kod”: Ansible som Linux-motsvarighet till GPO

När man vill att servrar ska se likadana ut över tid, oavsett vem som satt dem, brukar **Ansible** vara ett lättillgängligt verktyg. Idén är att beskriva önskat tillstånd i **YAML**; körningen är **idempotent**,

så bara det som faktiskt behöver ändras ändras. Många börjar med att standardisera paket, tjänster, sshd-inställningar, brandvägg, användare och filrättigheter — och lägger allt i ett versionshanterat repo.

Så här kan en liten bas-playbook se ut (illustration):

```
- hosts: linuxservrar
  become: true
  tasks:
    - name: Installera baspaket
      package:
        name: [vim, curl, htop, fail2ban]
        state: present

    - name: Hårda sshd (stäng lösenord)
      lineinfile:
        path: /etc/ssh/sshd_config
        regexp: '^PasswordAuthentication'
        line: 'PasswordAuthentication no'
      notify: restart ssh

    - name: Enkelt brandväggsläge
      ufw:
        state: enabled
        rule: allow
        port: "{{ item }}"
      loop: [22, 80, 443]

  handlers:
    - name: restart ssh
      service:
        name: ssh
        state: restarted
```

Exempel: *En nattlig körning applicerar “Bas-Linux” på alla servrar. Om någon råkat ändra en konfigfil för hand, återställs den till överenskommet läge.*

5.3.7 Programvarudistribution och uppdateringar på Linux

För själva programvaran upplever många att det blir smidigast att låta distributionens pakethanterare (APT eller DNF) göra jobbet. Det kan vara hjälpsamt att tänka i **ringar** även här: en liten pilotgrupp först, därefter bredare utrullning, och sist de mest känsliga systemen. Säkerhetsuppdateringar tenderar att rullas snabbare, medan funktionsuppdateringar ofta får passera en testperiod. På Debian/Ubuntu kan **unattended-upgrades** sköta säkerhetsfixar automatiskt; på RHEL-familjen finns **dnf-automatic**. Kärnuppdateringar kan kräva omstart, så planerade fönster gör livet enklare — i vissa miljöer använder man livepatch för att minska antalet omstarter.

Så här kan ett första steg mot automatiska säkerhetsfixar se ut (illustration):

```
# Debian/Ubuntu
sudo apt install unattended-upgrades
sudo dpkg-reconfigure unattended-upgrades

# RHEL/Rocky/Alma
sudo dnf install dnf-automatic
```


179

```
sudo systemctl enable --now dnf-automatic.timer
```

Exempel: *Tre inventariegrupper (pilot → bred → sen). Säkerhetsfixar går direkt till alla; större förändringar provar man en vecka i pilot innan de släpps vidare.*

5.3.8 Lokala paketförråd (repos) – försörjningskedjans säkerhet i praktiken

När många servrar ska uppdateras regelbundet vill man ofta ha **en egen, kontrollerad källa**. Ett lokalt repo — som spegel, caching-proxy eller **kuraterad** källa — gör att man kan granska och signera innehåll, frysa en viss versionsmängd och "promota" den från **dev → test → prod**. Det minskar exponeringen mot internet, ger förutsägbara byggen och gör rollback enklare.

Exempel: *Produktion pekar bara mot …`/prod/` och kräver signerat index. Test pekar mot …`/test/` och får nya paket en vecka tidigare. Upptäcks en regression i test, stannar promotion — produktion påverkas inte.*

Så här kan det se ut i Debian/Ubuntu-spår (översikt):
— repo-servern har en egen GPG-nyckel; indexfiler signeras.
— klienterna installerar **din** keyring och pekas mot **din** URL (med `signed-by=`).
— pinning kan låsa så att inget annat arkiv används av misstag.

Och i RHEL-spår:
— metadata genereras (t.ex. med `createrepo_c`), signeras och exponeras via HTTPS,
— klienterna pekar på ditt `baseurl` och kräver signatur/verifiering,
— externa `.repo`-filer avaktiveras och egress kan stängas i brandvägg.

5.3.9 Loggning och övervakning

Om man vill kunna förstå vad som hänt i efterhand — eller upptäcka avvikelser i tid — blir central loggning och egna mätvärden värdefulla. I Linux börjar det i **systemd-journald**; sedan skickas loggar ofta vidare till t.ex. rsyslog och en central loggtjänst. För mätvärden använder många en liten agent (t.ex. node_exporter) och ett system som kan larma när värden avviker från det normala.

Exempel: *SSH-inloggningar, sudo-försök och tjänstestarter skickas till en central vy. Ett enkelt larm reagerar om flera misslyckade inloggningar sker på kort tid.*

5.3.10 Backup och återställning

När man pratar backup handlar det i grunden om att bestämma **RPO** (hur mycket data man får förlora) och **RTO** (hur snabbt det måste gå att komma tillbaka). På Linux är verktyg som **restic** eller **borg** vanliga för filnivåbackup (deduplicering och kryptering), medan hypervisorn ofta tar **VM-backup**. Det brukar bli mest robust att kombinera båda: en bild av hela VM:n **och** en applikationskonsistent backup av själva datan i gästen. Snapshots (t.ex. ZFS/LVM) kan hjälpa vid förändringar — men ses inte som backup i sig.

Exempel: *Varje natt kör restic till ett S3-kompatibelt objektlager (med immutabilitet). En gång i veckan återställs en verklig mapp till en "karantän-server" och jämförs med checksumma.*

180

5.3.11 Säkerhetsbaslinje – hur “bra” brukar se ut

Många utgår från några återkommande byggstenar: **SSH med nycklar** (och gärna avstängd lösenordsinloggning), **default-deny** i brandväggen, **fail2ban** eller motsvarande för att dämpa bruteforce, ett **MAC-skydd** (AppArmor/SELinux) som faktiskt är påslaget, och enkel **filintegritetskontroll** på känsliga system. Principen “minsta privilegium” gäller i sudo/polkit — tillräckligt för att lösa uppgiften, men inte mer.

Exempel: *En bastion får samma bas som övriga, men med extra hårdning: enbart nyckel-SSH, tvåfaktor via PAM, stram polkit och realtidsloggning till SIEM.*

📌 Kontrollfrågor

1. Vilka fördelar får man av att beskriva Linux-servrar “som kod”, och hur påverkar det felsökning och återställning?
2. När kan en LTS-distribution vara ett klokt val, och vad vinner man på att främst använda standardarkiv?
3. Hur skulle du motivera nyckelinloggning i SSH och loggning av förhöjda kommandon för en ny admin?
4. Varför blir det lättare att felsöka om man håller sig till **ett** nätkonfigurationssätt, och när kommer VLAN/bond/bridge in i bilden?
5. I vilka lägen känns Samba/SMB respektive NFS mest naturligt, och hur påverkar valen rättighetshanteringen?
6. Vad kan AD/SSSD eller FreeIPA lösa i central identitet, och vad behöver fortfarande styras lokalt (sudo/polkit)?
7. Hur hjälper Ansible dig att hålla en baspolicy intakt, och vad betyder idempotens i vardagen?
8. Hur kan man lägga upp uppdateringar i ringar för Linux, och varför brukar säkerhetsfixar gå snabbare?
9. Vad vinner man på lokala repos (spegel, proxy eller kuraterat), och hur hänger signering och promotion ihop med rollback?
10. Hur kombinerar du VM-backup och gäst-backup för att nå RPO/RTO, och hur visar du att återställning faktiskt fungerar?

182

🔗 Fördjupningslänkar

- Ubuntu Server Guide: `https://ubuntu.com/server/docs`
- Debian Admin Handbook: `https://debian-handbook.info/`
- Red Hat Enterprise Linux – Security & Hardening: `https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/`
- Ansible Documentation: `https://docs.ansible.com/`
- Netplan Examples: `https://netplan.io/examples`
- OpenSSH manpages: `https://man.openbsd.org/ssh_config`
- Samba Documentation: `https://www.samba.org/samba/docs/`
- FreeIPA Documentation: `https://www.freeipa.org/page/Documentation`
- restic (backup): `https://restic.net/`
- Prometheus node_exporter: `https://prometheus.io/docs/guides/node-exporter/`

183

✍️ Egna anteckningar

...

184

5.4 MDM i praktiken

5.4.1 Vad är MDM, EMM och UDM/UEM — likheter och skillnader

Om man vill styra datorer och mobila enheter centralt möter man tre begrepp: **MDM**, **EMM** och **UDM/UEM**. De bygger på samma idé men har olika räckvidd.

- **MDM (Mobile Device Management)** fokuserar på *enheten*: registrera den, tilldela profiler (WLAN, VPN, certifikat), slå på kryptering/brandvägg, rulla ut appar och vid behov fjärrlåsa eller rensa.
- **EMM (Enterprise Mobility Management)** bygger vidare med app-hantering och dataskydd i mobila appar (t.ex. separera jobbdata från privat).
- **UEM/UDM (Unified Endpoint Management/Device Management)** samlar *alla slutpunkter* (Windows, macOS, iOS/iPadOS, Android — och ofta även Linux/särskilda enheter) i ett och samma system och knyter ihop **identitet**, **compliance**, **villkorad åtkomst**, **patch** och ibland **endpoint-skydd**.

I vardagen används orden ibland blandat; viktigast är vad plattformen faktiskt klarar och hur djupt den når i varje operativsystem.

Exempel (Exempel:) *En verksamhet börjar med MDM för att få kryptering och WLAN-profiler på bärbara. När behoven växer (app-livscykel, villkorad åtkomst, patch) landar man i en UEM-plattform som knyts till identitetstjänsten.*

5.4.2 Varför MDM/UEM behövs — automatisering, förutsägbarhet och säkerhet

Om man ska hantera många enheter över tid fungerar inte manuellt pill. MDM/UEM gör konfiguration **upprepad och spårbar**: nya enheter går från kartong till drift på minuter, förändringar rullas ut kontrollerat och säkerhetskrav kan **bevisas** (t.ex. “alla bärbara är krypterade, brandväggen är på, och OS är patchat”). Utan MDM/UEM uppstår ofta driftsspridning, bortglömda uppdateringar, delade WLAN-lösenord och osäker onboarding/offboarding.

Exempel (Exempel:) *En borttappad laptop med MDM är krypterad, spärras omedelbart och kan fjärr-rensas. Utan MDM vet ingen om den var krypterad eller vad den fortfarande når.*

5.4.3 Enrollment — manuell, automatisk och “zero-touch”

Enrollment kopplar enheten till plattformen så att policyer och appar börjar gälla.

- **Manuell enrollment** passar BYOD: användaren installerar “företagsportalen”, loggar in och får en arbetsprofil (Android) eller hanteras via enhetens hanteringsvy (iOS/macOS).
- **Automatisk/zero-touch** passar företagsägda enheter: serienummer/hårdvaru-ID registreras i förväg hos återförsäljaren. När enheten startar vet den vem som äger den och låser sig till rätt MDM.
    - **Windows:** Autopilot (användardrivet, self-deploying eller pre-provisioned).

185

- **Apple:** Automated Device Enrollment via Apple Business Manager (MDM-tvingning).
- **Android:** Android Enterprise med zero-touch/QR/KNOX Mobile Enrollment. **Knox** är **Samsungägt/styrt** och ger extra djup på Samsung-enheter.
- **Chromebook:** enrollment via Google Admin.
- **Linux:** ingen universell MDM-kanal; man använder agent-/policyverktyg i en UEM-lösning (compliance, skript, patch).

Exempel (Exempel:) *En företagsägd Android startas, känner igen sin ägare via zero-touch och skapar en arbetsprofil. Policyer och företagsappar landar utan att IT rört enheten.*

5.4.4 Microsoft Intune — arkitektur, förutsättningar och koppling till Windows AD

Om man vill ha en modern, molnburen plattform som kan styra många olika operativsystem är ***Microsoft Intune*** *ett naturligt val. I praktiken fungerar Intune som ett* ***styrplan****: tjänsten pratar den inbyggda* ***MDM-kanalen*** *mot varje klient (Windows, iOS/iPadOS, macOS, Android och i viss mån Linux) och ser till att profiler, policyer, appar och uppdateringsregler landar där de ska. Intune krokar djupt i* ***Microsoft Entra*** *(identitet och grupper), i* ***villkorad åtkomst*** *(bara "gröna" enheter får nå känsliga appar) och samspelar ofta med* ***Microsoft Defender*** *för att koppla säkerhetsstatus till åtkomstbeslut. I kontrast till äldre, lokala lösningar (som enbart* ***GPO*** *eller en fristående* ***SCCM/MECM****) är Intune designad för att fungera var enheten än befinner sig — på kontoret, i hemmet eller på resa — så länge den når internet.*

För att Intune ska kunna arbeta behöver några grundbultar vara på plats. Själva **tenanten** måste ha licenser som ger Intune/UEM-funktionalitet, och man sätter **MDM-auktoriteten** till Intune så att plattformen verkligen är "ägaren" av MDM-kanalen. Därefter kopplar man varje plattform till sitt respektive ekosystem: Apple-enheter kräver ett **APNs-certifikat** och blir starkast när de hanteras via **Apple Business Manager** (där både **ADE** för automatisk enhetsregistrering och **VPP** för app-licenser finns). Android-sidan kopplas via **Android Enterprise-bindning**, och i företagsägda scenarier använder man gärna **zero-touch**; på Samsung-enheter finns dessutom **Knox-stödet** som ger extra djup. Windows-datorer registreras smidigast via **Autopilot**, där enhetens hårdvaru-ID (hash) gör att rätt profil och rätt appar rullar in redan vid första uppstarten.

Så fort man vill använda **certifikat** för saker som **WLAN (EAP-TLS)** eller **VPN** behöver man också en **certifikatkedja**. Det löses vanligtvis med en **Intune Certificate Connector** på en server **on-prem** som pratar med organisationens **AD CS**. För Windows-klienter kan man antingen dela ut cert direkt (**PKCS**) eller via **SCEP/NDES** för mer skalbarhet och automation. Poängen är att Intune bestämmer *vem* som ska få vilket certifikat och *när*, medan AD CS faktiskt **utfärdar** det; klienterna kan därefter ansluta till rätt **WLAN** (operativsystemet kallar det ofta "Wi-Fi", men i boken använder vi begreppet **WLAN**) och rätt VPN utan delade lösenord.

Hur Intune hänger ihop med ett befintligt **Windows AD** beror på vilken väg man väljer. I ett **moln-först-upplägg** blir enheten **Entra-joinad** (d.v.s. ansluten till molnets katalog) och styrs helt via

186

Intune; GPO minimeras och identitet/åtkomst villkoras i molnet. Det här brukar vara enklast att förstå för nya klienter eftersom all styrning kommer från samma håll, och enheten fungerar lika bra på kontoret som utanför. I ett **hybrid-join-upplägg** lever enheten samtidigt i **on-prem AD** och i **Entra**. Då kan vissa historiska GPO:er finnas kvar medan Intune tar över det mesta av klientstyrningen (MDM-policy, appar, uppdateringar). Om man dessutom vill köra **Autopilot Hybrid Join** behövs en liten komponent — **Intune Connector for Active Directory** — som kan skapa en "offline domain join"-blob mot en DC under första uppstarten. Slutligen finns **co-management**, där en miljö med **SCCM/MECM** gradvis delar på ansvaret med Intune; till exempel kan **Windows Update for Business** och compliance flyttas till Intune medan vissa appar fortsatt levereras via SCCM. Oavsett väg är grundidén densamma: låt Intune vara **policy-motorn**, låt **Entra** vara **identitetslagret**, och använd **AD CS** som **tillitsfabrik** för certifikat.

När allt detta sitter på plats blir flödena begripliga. En ny bärbar dator som är registrerad i **Autopilot** startar upp, känner igen sin ägare, **enrollerar** sig själv i Intune och hämtar **basprofiler**: kryptering (BitLocker), brandvägg, uppdateringsring, certifikat och **WLAN-profil**. **Compliance** sätter ribban för vad som är "grönt", och **villkorad åtkomst** använder den statusen när användaren försöker nå exempelvis e-post eller affärssystem. Om datorn istället ärver några kvarvarande **GPO** (i ett hybridläge) gäller det att hålla ansvarsfördelningen tydlig så man undviker policykrockar: det som ligger i molnet bör i första hand också styras från molnet, medan servernära eller speciella arv kan få leva kvar i GPO under en övergångsperiod.

Exempel: *En organisation med AD on-prem synkar användare och grupper till Entra för att slippa dubbla kataloger. Nya Windows-klienter körs som* ***moln-först*** *med Entra-join och Intune för all klientstyrning. Certifikat delas ut via SCEP-connectorn, och klienterna ansluter till skolans* ***WLAN*** *med* ***EAP-TLS****. Några särskilda labbmiljöer ligger kvar i en* ***hybrid-OU*** *där ett fåtal GPO:er används — men allt som rör klienternas säkerhetsbas, appar och uppdateringar ligger i Intune. Resultatet blir snabb onboarding, jämn säkerhetsnivå och tydlig spårbarhet.*

5.4.5 Intune på djupet — bygg en bra grund (varför, hur, och vad som händer om man låter bli)

- **Gruppdesign (dynamiska grupper)**
  Varför: policyer/ appar ska landa rätt automatiskt (t.ex. "alla Windows-laptops").
  Hur: använd attribut (plattform, enhetstyp/ägande, taggar). Håll grupperna få och begripliga.
  Om man låter bli: manuell målning → felträffar, glapp i säkerhet, långsam utrullning.
- **Basprofiler för nät & certifikat (WLAN/VPN + PKCS/SCEP)**
  Varför: EAP-TLS ersätter delade lösenord, minskar läckage och support.
  Hur: sätt upp certifikatconnector + AD CS; skapa cert- och WLAN-profiler för aktuella SSID.
  Om man låter bli: delade PSK sprids, byten blir stökiga, obehöriga kan "läcka in".
- **Compliance + villkorad åtkomst**
  Varför: endast "gröna" enheter (kryptering, brandvägg, OS-nivå) får nå känsliga appar.

187

Hur: definiera compliance per plattform; lås kritiska appar bakom villkorad åtkomst.
Om man låter bli: stulna/eftersatta klienter når data; svårt att bevisa efterlevnad.

- **Uppdateringsringar (WUfB)**
  Varför: kvalitetsuppdateringar snabbt; funktionsuppgraderingar i ringar (Pilot → Bred → Sen).
  Hur: skapa ringar med rimliga deferrals/deadlines; följ upp regelefterlevnad.
  Om man låter bli: ojämn patch-nivå, fler incidenter och “stora hopp” som skapar driftstopp.

- **App-livscykel (Win32-paket, detektion, beroenden)**
  Varför: rätt version till rätt grupp; repeterbar installation/avinstallation.
  Hur: definiera install/uninstall-kommandon, **detektionsregler** och beroenden.
  Om man låter bli: “zombie-appar”, dubbletter/konflikter, oklar klientstatus.

- **Autopilot + Enrollment Status Page (ESP)**
  Varför: “kartong → färdig” utan handpåläggning; ESP blockerar inloggning tills basen är klar.
  Hur: registrera enheter i Autopilot; välj scenario (user-driven/self-deploying/pre-provisioned); aktivera ESP.
  Om man låter bli: lång onboarding, fler manuella fel, användare börjar på halvfärdig klient.

- **RBAC och förändringsstyrning**
  Varför: minska risken för fel i stor skala och skapa spårbarhet.
  Hur: rollstyr åtkomst (minsta privilegium), inför dev/test/prod-taggning och change-fönster.
  Om man låter bli: svår root-cause, oavsiktliga ändringar, större påverkan vid misstag.

- **Skript & proaktiv remediering**
  Varför: hitta/laga små driftsavvikelser automatiskt, innan de blir ärenden.
  Hur: PowerShell (Windows) / shell (macOS/Linux där stött); kör schemalagda kontroller + korrigeringar.
  Om man låter bli: små fel byggs upp och skapar störningar och manuellt efterarbete.

Exempel (Exempel:) *“Bas-Windows”: dynamisk grupp för laptops, BitLocker-policy + compliance, Defender + ASR, WUfB-ringar, cert-/WLAN-profiler, Autopilot med ESP, samt kärnappar. Resten läggs på i lager.*

5.4.6 Konfigurationsexempel — utförliga typfall

- **Windows (företagsägt) — Autopilot user-driven**
  Flöde: klienten startas, kopplar till tenant, ESP visar förlopp; BitLocker, brandvägg och Defender/ASR slås på; WUfB-ring och cert/WLAN-profiler landar; Office + affärsappar installeras.
  Resultat: användaren loggar in på en färdig, krypterad klient.
  Utan detta: manuell “gör i ordning”-process, osäker säkerhetsnivå och ojämnhet mellan klienter.

188

- **macOS (företagsägt) — ADE + FileVault**
  Flöde: enheten binds i ABM, enrolleras automatiskt, FileVault krypterar, systeminställningar låses där det behövs; cert/WLAN och appar (VPP/pkg) landar.
  Resultat: enhetliga Apple-klienter; återställningsnycklar hanteras enligt policy.
  Utan detta: kryptering hoppas över, lokala "hemfixar", svår support och efterlevnad.
- **iOS/iPadOS — ADE, låst layout och selektiv wipe**
  Flöde: auto-enrollment, PIN-krav, WLAN, e-postprofil; appar pushas; hemskärm layoutas för arbetsflöde; vid förlust → fjärr-rensa.
  Resultat: förutsägbart gränssnitt och låg supportbelastning.
  Utan detta: spretig användarupplevelse, appar saknas, data blandas med privat.
- **Android — Work Profile (BYOD) och COPE/COBO (företagsägt)**
  Flöde: BYOD får arbetsprofil där jobbdata hålls separerat; företagsägt kan vara låst (COBO) eller blandat (COPE). Zero-touch/Knox underlättar.
  Resultat: balans mellan kontroll och integritet.
  Utan detta: jobbdata blandas med privat; svårt att rensa vid avslut.
- **Linux (utvalda distron) — agentbaserad compliance**
  Flöde: agent registrerar enheten; compliance kräver kryptering, brandvägg och minsta OS-nivå; skript åtgärdar avvikelser.
  Resultat: Linux-klienter uppfyller samma "grönt-krav" innan de når resurser.
  Utan detta: parallell värld med oklara regler och högre risk.
- **WLAN via certifikat (EAP-TLS)**
  Flöde: Intune delar ut klientcertifikat; en WLAN-profil ansluter mot SSID som kräver cert; tar man bort behörighet tappar enheten nätåtkomst utan lösenordsbyte i infrastrukturen.
  Resultat: inga delade PSK; enkel offboarding; hög säkerhet.
  Utan detta: PSK sprids och lever för länge, byten blir stökiga, onödig exponering.

5.4.7 Säkerhetsaspekter vid fjärrhantering — tre saker som ger störst effekt

1. **Kryptering + skärmlås** på alla bärbara/telefoner.
2. **Compliance + villkorad åtkomst** så att bara "gröna" enheter når känsliga appar.
3. **Certifikat + segmentering** (EAP-TLS för WLAN/VPN, rätt VLAN/roll i nätet).

Lägg till **RBAC** för MDM-admins, tydlig **loggning** av fjärråtgärder och **sekretess** för BYOD (MDM ska inte kunna läsa privat data). Det bygger förtroende utan att sänka säkerheten.

Exempel (Exempel:) *En hittad men misstänkt kompromitterad telefon blockeras direkt, jobbprofilen rensas, cert spärras. Åtkomst öppnas först efter ny enrollment och grön compliance.*

189

5.4.8 Jämförelse i korthet — Intune, Jamf, Workspace ONE, Miradore och Linux-vänliga alternativ

- **Microsoft Intune:** bred täckning (Windows, iOS/iPadOS, macOS, Android; visst Linux-stöd), stark koppling till Entra och villkorad åtkomst, Autopilot.
- **Jamf Pro:** djup och snabb för Apple-plattformar; kan samspela med Intune för compliance.
- **VMware Workspace ONE (AirWatch):** stark mobilhantering, brett OS-stöd, mogna appflöden.
- **Miradore/Hexnode/Kandji/Meraki SM:** lättare insteg; funktionellt djup varierar. **Miradore** är ett praktiskt multi-OS-alternativ.
- **Linux-inriktat:** JumpCloud, Canonical Landscape, SUSE Manager, Red Hat Satellite (agent/policy/patch/inventering i UEM-anda).

Exempel (Exempel:) *En blandad miljö kör Intune som "nav" (compliance, villkorad åtkomst, Autopilot). Apple-delen hanteras i Jamf men rapporterar compliance till Intune.*

5.4.9 Vanliga misstag — och hur man undviker dem

- Starta för brett: börja med **bas** (kryptering, skärmlås, uppdateringar, cert-WLAN).
- Otydliga grupper: bygg **dynamiska** målgrupper och testa på pilot.
- Blanda företagsägt och BYOD: håll isär; **selektiv wipe** för BYOD.
- Lösenords-WLAN i produktion: gå tidigt till **EAP-TLS**.
- Ingen återställningsplan: *öva* fjärr-wipe, återställning och om-enrollment.

Exempel (Exempel:) *En pilotgrupp får "Bas 1.0". När den är stabil lägger man på appar och finjusteringar. Färre överraskningar och färre supportärenden.*

190

📌 Kontrollfrågor

1. Förklara skillnaden mellan **MDM**, **EMM** och **UEM/UDM** och när man typiskt växlar upp.
2. Vad vinner man på att koppla **compliance** i MDM till **villkorad åtkomst**?
3. När passar **manuell** enrollment bäst, och när är **zero-touch** att föredra?
4. Vad är **Samsung Knox**, och varför spelar det roll för Android-enheter från Samsung?
5. Vilka **förutsättningar** behöver ofta vara på plats innan Intune kan användas fullt ut (per plattform)?
6. Hur samverkar **Intune** med **Windows AD** i **moln-först**, **hybrid-join** och **co-management**?
7. Varför är **EAP-TLS** för WLAN ett lyft jämfört med delade lösenord, och vad händer om man inte inför det?
8. Beskriv en **WUfB-ringdesign** (Pilot → Bred → Sen) och vad som kan gå fel utan ringar.
9. Hur hjälper **dynamiska grupper**, **RBAC** och **ESP** till att göra utrullningar robusta?
10. Välj ett av konfigurationsexemplen och förklara vad som händer “om man låter bli” det steget.

191

🔗 Fördjupningslänkar

- Microsoft Intune (översikt och guider): `https://learn.microsoft.com/mem/intune/`
- Windows Autopilot (översikt): `https://learn.microsoft.com/mem/autopilot/`
- Apple Business Manager (ADE): `https://support.apple.com/guide/apple-business-manager/welcome/web`
- Android Enterprise (enrollment-alternativ): `https://developer.android.com/work`
- Samsung Knox (KME/Knox Suite): `https://www.samsungknox.com/`
- Jamf Pro: `https://learn.jamf.com/`
- VMware Workspace ONE: `https://www.vmware.com/products/workspace-one.html`
- Miradore: `https://www.miradore.com/`
- JumpCloud (device + identity): `https://jumpcloud.com/`
- Google Admin — ChromeOS enhetsregistrering: `https://support.google.com/chrome/a/answer/1360534`

192

✍️ Egna anteckningar

193

5.5 Framtiden för nätverk

5.5.1 SDN – från lågnivåkonfiguration till intent och API:er

I dag bygger mycket nätverksdrift fortfarande på att tekniker konfigurerar enheter var för sig: port för port, ACL för ACL. **Software-Defined Networking (SDN)** byter perspektiv. Styrplanet flyttas upp i en **kontroller** som förstår hela topologin, och nätet styrs via **API:er** och deklarativa policys: "det här ska vara tillåtet mellan de här tjänsterna, med den här säkerhetsnivån". Kontrollen fördelas sedan ned till hårdvaran (switchar, routrar, brandväggar) som ren dataplanslogik. I datacenter är **EVPN/VXLAN** och fabric-kontrollers vardag; i WAN är **SD-WAN** redan brett infört; i campus kommer "intentbaserad" segmentering och identitetsstyrd åtkomst steg för steg.

Konsekvensen är att **feltyperna förändras**. I ett klassiskt nät var fel ofta fysiska (länk ned, loop). I ett SDN blir fel oftare **logiska**: en policy träffar fel grupp, ett objekt saknar tagg, ett API-samtal misslyckas. Det ställer nya krav på **observability** (telemetri, flödesdata, modellinsyn) och på hur vi arbetar: versionering av förändringar, **Git som sanning**, **CI/CD** för nät, och **"pre-change"-tester** i sandlåda eller digital tvilling.

För användare märks skiftet som **snabbare förändring** (nya VLAN/WLAN, ny QoS, ny site) och **jämnare kvalitet**. För driftteamet innebär det att **CLI-hantverket** kompletteras – inte försvinner – av **automation**: YANG/NETCONF/RESTCONF, gNMI, Ansible/Terraform, Python. Rollen glider mot **policy-, data- och livscykelarbete** snarare än "klicka i rätt ruta". Den som kan läsa logik, förstå modellering och skriva tester blir lika viktig som den som känner varje kommando utantill.

Exempel (Exempel:*)* En ny applikation ska ut i tre zoner. I stället för att skapa ACL:er och VLAN på femton switchar beskriver du i kontrollern vilka roller som får prata med vilka, och kontrollerna skapar underliggande flöden. En rollfelmappning gör att bara testzonen får trafik; telemetri och "intent verification" visar exakt var policyn stoppar.*

5.5.2 Zero Trust – identitet först, nätet som transport

Zero Trust utgår från antagandet att **inget** är pålitligt per automatik – inte ens "insidan". Åtkomst beslutas **varje gång**, baserat på **identitet**, **enhetens skick (compliance)**, **kontext** och **risk**. Nätet blir mer av **transport**, medan kontrollen sker i kombination av **identitetstjänst**, **MDM/UEM**, **applikationsproxy** och **mikrosegmentering**. I campus/edge innebär det att **VLAN som grov kniv** ersätts av **finmaskiga policys** nära klient eller arbetsbelastning. I datahall och moln betyder det att arbetslaster isoleras per **tjänst/roll** snarare än per IP-subnät.

För användare betyder Zero Trust fler **tysta kontroller** (enhetens status, plats, beteende) och färre "stora" lösenordshinder – men **MFA** och **device posture** blir vardag. För driftteamet betyder det **färre platta nät**, **fler policys** och **mer samarbete** med identitetsteam och applikationsägare. Den praktiska utmaningen är **komplexitet**: policyexplosion om man inte modellerar roller rätt, och **friktion** om kraven inte är realistiskt satta. Nyckeln blir **enkla basregler**, tydliga undantagsprocesser och **telemetri** som visar *varför* något nekas.

194

***Exempel (Exempel:**)* *En klient med gammalt OS tappar åtkomst till HR-systemet för att compliance-regeln kräver uppdaterad kernel och kryptering. Användaren får guidning i klientportalen; access öppnas igen när enheten är grön.*

5.5.3 IPv6-implementering – från dual-stack till IPv6-only

IPv6 har länge "varit på gång", men de senaste åren har förutsättningarna mognat: operatörer kör IPv6 i ryggraden, stora innehållsleverantörer exponerar tjänster i IPv6, och klient-OS har stabil **privacy**-logik. Nästa steg är **IPv6-only** i allt fler miljöer, med **NAT64/DNS64** eller 464XLAT för det som ännu saknar IPv6. Vinsten är **enklare adressering i stor skala**, **färre NAT-lager** och möjlighet till **renare end-to-end-säkerhet**. Samtidigt blir **säkerhetsmodellen tydligare**: NAT var aldrig en brandvägg – IPv6 kräver **uttalade filter**, god **ICMPv6-hygien** (annars dör funktioner), och skydd mot **RA-spoofing** och felkonfigurerad **SLAAC/DHCPv6**.

För användare märks lite – oftast **bättre reachability** och färre märkliga fel. För driftteamet märks mycket: **adressplanering i /64-tänk**, **brandväggsregler per tjänst**, uppdaterad **övervakning** (flow/NetFlow/IPFIX som förstår v6), och **felsökning** där man komfortabelt läser **Neighbor Discovery** och **ICMPv6**. Övergången kräver **dual-stack-period**, **piloter**, tydlig **inventering av v4-beroenden**, och planerad **avveckling** av gamla NAT-antaganden. I campus och WLAN handlar det om att RA-guard och DHCPv6-snooping fungerar lika bra som IPv4-kontrollerna. I datacenter och moln om att **säkerhetspolicys** (SG/NSG/ACL) migreras utan att varken över- eller under-skydda.

***Exempel (Exempel:**)* *Ett elev-VLAN blir IPv6-only. DNS64/NAT64 ger fortfarande åtkomst till IPv4-webb. Loggarna visar lägre felsökningsbelastning, men brandväggen fick expanderade v6-regler för att ersätta gamla "NAT-skyddet".*

5.5.4 AI och automatisering – från NMS till beslutssystem

Automatisering har länge handlat om skript och verktyg som Ansible. Nästa steg är **AI-stödd drift**: system som **förutser** fel, **förklarar** avvikelser och i kontrollerade fall **åtgärdar** dem. I WLAN ser vi redan självlärande **radio-optimering** (kanal/effekt), i WAN **policy-styrd ruttval/QoS**, i DC **intent-verifiering** som säkerställer att dataplansflöden matchar avsikten. Med **AIOps** blir brusiga larm till **händelser med sammanhang**: "allt pekar på att DHCP-svar i VLAN 20 stoppas på denna trunk". Med **LLM-stöd** kan runbooks bli **konversationsbara**: "visa sannolik rotorsak och föreslå återställning".

Vinsten är **hastighet och kvalitet**. Risken är **överförtroende**: en modell kan "hallucinera", hamna i **feedbackloopar** eller agera på **partiska data**. Därför blir **skyddsräcken** centrala: förändringar ska gå genom **pull-requests**, testas i **sandlåda**, **stegvis rullas** och **mätas** efteråt. Data måste hanteras med **integritet**: nättelemetri kan innehålla personuppgifter eller känsliga mönster. Rollen för nätverksteknikern blir mer **ingenjör/mätare**: skriva **policyer**, definiera **SLO:er** (t.ex. anslutningslatens i WLAN), designa **mätningar** och granska **maskinens förslag**. För användare kan AI betyda **stabilare upplevelse** – om man samtidigt accepterar **mindre "magi" och mer transparens**: varför fattades ett beslut, och hur ångras det?

195

***Exempel (Exempel:**)* *Ett AIOps-system ser mönstret “AP-roaming tar >300 ms” i en byggnad, kopplar det till en ny störkälla i 2,4 GHz och föreslår kanalplanering/effektjustering. Förslaget körs i nattfönster mot en testzon och sprids först efter mätbar förbättring.*

5.5.5 Etik och cybersäkerhet – data, styrka och ansvar

När nätet blir mer **mätbart** och mer **automatiserat** ökar både möjligheten och frestelsen att **se mer**. Telemetri, loggar och flöden kan avslöja arbetsmönster, platser och beteenden. **Integritet** måste därför vara en **första-klass-fråga**: samla bara det som behövs, **pseudonymisera** där det går, begränsa **åtkomst** och **lagringstid**, och var **öppen** med vad som samlas. I skola och offentlig sektor blir detta särskilt viktigt. På säkerhetssidan fortsätter **segmentering**, **Zero Trust-tänk**, **härdning** och **snabb patchning** att vara grund. Till det kommer **leverantörskedja** (firmware, SBOM, signering), **out-of-band-hantering** (så att man kan rädda även vid ransomware), och uppmärksamhet på **kryptoframtid** (post-kvant kommer påverka hur vi ser på nycklar och livslängd).

Yrkesrollen rör sig också etiskt: fler beslut om **övervakningens gränser**, **modellers skälighet**, **vendor-låsning** och **hållbarhet** (energi, värme, livslängd). Teknikern blir inte bara den som “får det att fungera”, utan den som **formar hur det bör fungera** – i balans mellan nytta, risk och rättigheter.

Sammanfattning att ta med sig

- **SDN** gör nät till **policy + API**; hantverk blir livscykel, test och modell.
- **Zero Trust** flyttar förtroendet till **identitet och enhetens skick**; nätet segmenteras finmaskigt.
- **IPv6** går från “parallellt” till **IPv6-only**; NAT ersätts av **tydlig filtrering** och god ICMPv6-hygien.
- **AI/AIOps** höjer hastighet och kvalitet – men kräver **skyddsräcken**, **mätning** och **mänsklig granskning**.
- **Etik & säkerhet** blir inbyggt: minst data, tydliga roller, snabb patch, stark kedja från firmware till policy.

Frågor att fundera på

- Vilken **minsta uppsättning mätvärden** behöver vi för att kunna lita på ett AI-förslag i *vårt* nät?
- Var går gränsen mellan **nödvändig telemetri** och **onödig övervakning** i en skolmiljö?
- Om vi gör **IPv6-only** i ett delnät, vilka **tio saker** måste vara på plats för att det ska bli en förbättring – inte en risk?
- Vilka **färdigheter** (språk, verktyg, metoder) behöver en nätverkstekniker lägga till de kommande två åren för att må bra i SDN/Zero Trust-världen?
- Hur beskriver vi **intentionen** för nätet så att den är testbar före och efter varje förändring?

196

📌 Kontrollfrågor

1. När ett nät styrs via en SDN-kontroller och “intent”, vilka nya feltyper uppstår jämfört med klassisk CLI-konfiguration – och hur märker man dem i drift?
2. Hur skulle du beskriva skillnaden mellan **mikrosegmentering** i ett Zero Trust-tänk och traditionell segmentering med VLAN/ACL – ur både säkerhets- och förvaltningsperspektiv?
3. Om ni inför **Zero Trust**: vilka tre datakällor (identitet, enhet, kontext) måste vara pålitliga för att besluten ska bli rimliga – och hur validerar ni dem?
4. Du får uppdraget att göra ett **IPv6-pilot** i ett campus-VLAN. Vilka tio praktiska kontroller vill du se “gröna” innan pilotens användare märker något negativt?
5. Vid **IPv6-only** med NAT64/DNS64: vilka applikationstyper brukar strula, och vilka mätetal sätter du upp för att snabbt fånga fel?
6. Nämn tre **skyddsräcken** du vill ha på plats innan ni låter automation/AI göra förändringar i nät (tänk kodflöde, test, övervakning).
7. Vilka **SLO:er** (service level objectives) är mest meningsfulla i ett modernt WLAN – och hur kopplas de till åtgärder i radionät/klientpolicy?
8. Vad är ett bra **minsta datalager** för AIOps (telemetri, flöden, loggar) så att analys blir möjlig utan att tumma på integriteten?
9. Hur förändras nätverksteknikerns **kompetensprofil** när SDN, Zero Trust och automatisering blir vardag – och hur tränar man det effektivt i skolan?
10. Välj en realistisk **etikfråga** (t.ex. klienttelemetri i skolmiljö). Var går gränsen mellan nödvändig mätning och onödig övervakning, och hur förklarar du det för användarna?

197

🔗 Fördjupningslänkar

- ONF – *Software-Defined Networking (SDN) Definition*: `https://opennetworking.org/sdn-definition/`
- RFC 7348 – *VXLAN*: `https://datatracker.ietf.org/doc/rfc7348/`
- RFC 7432 – *BGP EVPN*: `https://datatracker.ietf.org/doc/rfc7432/`
- NIST SP 800-207 – *Zero Trust Architecture*: `https://csrc.nist.gov/publications/detail/sp/800-207/final`
- RFC 8200 – *Internet Protocol, Version 6 (IPv6) Specification*: `https://datatracker.ietf.org/doc/rfc8200/`
- RFC 4861 / RFC 4862 – *Neighbor Discovery* / *SLAAC*: `https://datatracker.ietf.org/doc/rfc4861/` | `https://datatracker.ietf.org/doc/rfc4862/`
- RFC 4443 – *ICMPv6*: `https://datatracker.ietf.org/doc/rfc4443/`
- RFC 6146 / RFC 6147 – *NAT64* / *DNS64*: `https://datatracker.ietf.org/doc/rfc6146/` | `https://datatracker.ietf.org/doc/rfc6147/`
- Google SRE – *Service Level Objectives*: `https://sre.google/sre-book/service-level-objectives/`
- NIST AI RMF 1.0 – *AI Risk Management Framework*: `https://www.nist.gov/itl/ai-risk-management-framework`

198

✍️ Egna anteckningar

…

199

Appendix I – Begrepp

Not: I brödtexten använder vi termen ”WLAN”. ”Wi-Fi” syftar här på certifieringsprogrammet/varumärket och används i denna begreppsruta.

ABM (Apple Business Manager) – Portal för enhets- och appinköp; grunden för automatisk Apple-enrollment (ADE) och VPP-licenser.

ACL – Åtkomstlista som avgör vilken trafik som tillåts eller nekas.

ACME – Protokoll för automatisk hantering av TLS-certifikat (t.ex. certbot).

AD (Windows Active Directory) – Katalogtjänst för identiteter, datorer och policyer.

AD CS – Certifikatinfrastruktur (CA) i Windowsmiljö.

AIDE – Filintegritetskontroll som larmar när filer ändras otillåtet.

AIOps – AI-stött drift som korrelerar larm, hittar orsaker och föreslår åtgärder.

Air-gap – System eller kopior som är fysiskt/logiskt isolerade från nätet.

Android zero-touch – Förregistrering av företagsägda Android-enheter för “zero-touch”.

Ansible – Automationsverktyg som beskriver systemtillstånd i YAML och kör idempotent.

Anycast – Samma IP annonseras från flera platser; trafik hamnar hos närmaste nod.

API – Programmeringsgränssnitt; i SDN/UEM används för styrning och automation.

AP (Accesspunkt) – Trådlös basstation för klientanslutning i ett **WLAN** (OS kallar ofta “Wi-Fi”).

APIPA – Automatisk IPv4-adress 169.254.0.0/16 när DHCP saknas.

AppArmor – MAC-säkerhet i Linux som begränsar programrättigheter.

ARP – Protokoll som översätter IPv4-adresser till MAC-adresser i L2.

Autentisering – Bevis av identitet (t.ex. lösenord, certifikat, MFA).

Autopilot (Windows) – Zero-touch-registrering och uppsättning av Windows-klienter.

Backup – Säkerhetskopia av data för återställning vid fel eller incident.

Bandbredd – Maximal datamängd per tidsenhet på en länk.

Bastion/Jump host – Härdad mellanvärd för säker SSH/RDP-åtkomst.

BGP – Externt routingprotokoll som utbyter vägar mellan autonoma system.

BGP community – Attribut som styr policyn för hur rutter hanteras.

BGP ECMP – Lastdelning över flera likvärdiga vägar i BGP.

BGP route reflector – Minskar krav på full-mesh i iBGP.

BNC – Bajonettkontakt för koaxialkabel (historiskt/nischnät).

Bonjour/mDNS – Lokal namn- och tjänsteupptäckt i LAN.

BPDU – STP-kontrollramar som används för loopdetektering.

Bridge (brygga) – L2-koppling mellan segment; i servervärd: virtuell switch.

BSSID – MAC-adress för accesspunktens radiosändare (en SSID-cell).

BYOD – “Bring Your Own Device”; privatägd enhet i arbetsmiljö.

CA (Certificate Authority) – Utfärdare av digitala certifikat.

CAPA – Korrigerande och förebyggande åtgärder efter incident (Corrective/Preventive).

CASB – Cloud Access Security Broker; policy och kontroll för SaaS-åtkomst.

Cat6A (kabel) – Vanlig kopparkabel för 10G-Ethernet upp till 100 m.

Ceph – Distribuerad objekt/blocks/fil-lagring; vanligt i hypervisor-kluster.

Certificate pinning – Låsning av klient till specifik certifikatnyckel.

200

Certifikat – Kryptografisk legitimation kopplad till identitet/nyckel.
CIDR – Notation/metod för prefixlängd (t.ex. /24) och aggregering.
CI/CD – Automatiserat bygg, test och driftsättning (även för nätkonfig).
CLI – Kommandorad för konfiguration och felsökning.
Cloud-join (Entra-join) – Windows-klient ansluten direkt till Microsoft Entra ID.
Co-management – Samdrift av MECM/SCCM och Intune i samma klientpark.
COBO – Företagsägd, företagsanvänd Android (låst läge).
COPE – Företagsägd, personligen använd (balans jobb/privat).
Container – Isolerad applikationsmiljö (t.ex. Docker/LXC).
Controller (WLAN) – Central styrning/övervakning av många accesspunkter.
CoS (802.1p) – L2-prioritering i VLAN-taggen.
CRC – Felkontrollsumma för att upptäcka bitfel.
DAC-kabel – Direktkopplingskabel för SFP+/QSFP-port till korta avstånd.
DAD (Duplicate Address Detection) – IPv6-kontroll för att undvika adresskollision.
DAI (Dynamic ARP Inspection) – Skydd mot ARP-förfalskning i L2.
Data plane – Där paket faktiskt forwardas (jfr control/management plane).
Deduplicering – Eliminering av dublettdata i backup/lagring.
Default route – Standardväg när ingen mer specifik rutt matchar.
DFS (WLAN) – Dynamic Frequency Selection; kanalreglering i 5/6 GHz.
DFS (Windows) – Distributed File System; virtuella filnamnrymder/pekare/replikering.
DHCP – Dynamisk tilldelning av IP-konfiguration till klienter.
DHCP relay – Vidarebefordrar DHCP mellan VLAN/subnät.
DHCP snooping – Switchskydd som stoppar falska DHCP-servrar.
DHCPv6 – IPv6-varianten av DHCP (komplement till SLAAC).
DNS – Namnuppslagning mellan domännamn och IP-adresser.
DNSSEC – Signerad DNS för att skydda mot förfalskning.
DoH/DoT – DNS över HTTPS/TLS för integritet.
Docker – Containerplattform för att paketera och köra applikationer.
Domän – Administrativ enhet i AD eller en DNS-zon.
DPI (Deep Packet Inspection) – Klassning/inspektion på applikationsnivå.
DTP – Cisco-protokoll för att förhandla trunk/access (bör undvikas i produktion).
DR (Disaster Recovery) – Återstart av kritiska system efter större avbrott.
DSCP – QoS-märkning i IP-huvud för prioritering.
EAP – Ramverk för nätverksautentisering (802.1X).
EAP-PEAP – 802.1X-metod som kapslar lösenord i TLS-tunnel.
EAP-TLS – Certifikatbaserad 802.1X-autentisering (WLAN/VPN).
EAP-TTLS – 802.1X-metod med tunnlad autentisering utan klientcert.
Edge – Nätets ytterkant (site/klient-nära lager).
EIGRP – Cisco-routingprotokoll (dynamiskt, hybridmodell).
ESP (Enrollment Status Page) – Intune-sida som blockerar tills baskrav är uppfyllda.
ESP (IPsec) – Encapsulating Security Payload; kryptering/integritet i IPsec.
EVPN – BGP-baserad L2/L3-VPN, ofta med VXLAN i datacenter.

201

Fail2ban – Blockerar upprepade felaktiga inloggningsförsök.
Failover – Automatisk övertagning när primär komponent faller.
FHRP – Protokoll för redundant gateway (t.ex. VRRP/HSRP/GLBP).
FileVault – Diskkryptering i macOS.
Firewall (tillståndsfull) – Brandvägg som förstår och spårar sessionsflöden.
FreeIPA – Linuxidentitet (LDAP/Kerberos/CA/HBAC), likt AD.
FTP/SFTP – Filöverföring; SFTP går över SSH och är säkert.
Gateway – Standardväg ut ur ett subnät (L3-gräns).
Grafana – Visualisering av tidsserier (dashboards).
GRE – Tunnelprotokoll för inkapsling av trafik.
GFS (Grandfather-Father-Son) – Backup-retentionsmodell dag/vecka/månad.
Git – Versionshantering; grund för GitOps även i nät.
GNS3/EVE-NG – Emulering/virtualisering för nätlabbar.
GPO – Group Policy Objects; Windows-policyer kopplade till AD.
GPO-länkning – Att koppla GPO till Site/Domain/OU i AD-trädet.
GUA – Globala IPv6-adresser som routas på internet.
HA (High Availability) – Design för att undvika enkel felpunkt.
HBA (Host Bus Adapter) – Gränssnitt mellan server och SAN/lagring.
HCI (Hyper-Converged Infrastructure) – Kluster som förenar compute/lagring/nät.
Helm/Kubernetes – Orkestrering av containeriserade applikationer (paket/utrullning).
Honeypot – Avsiktligt lockbete för att observera angripare.
Hop – Ett routingsteg mellan källa och destination.
Host – Slutenhet/servrar/klienter som använder nätet.
Host-ID – Värddelen av en IP-adress (allt utanför Net-ID).
HSRP – Ciscos redundanta gateway (FHRP), liknande VRRP.
HTTP/HTTPS – Webprotokoll; HTTPS säkrat över TLS.
Hypervisor – Mjukvara som kör och isolerar virtuella maskiner.
IBN (Intent-Based Networking) – Nät styrt av deklarerad avsikt/policy.
ICMP – Kontrollprotokoll (ping, felmeddelanden m.m.).
ICMPv6 – IPv6-kontroll: Neighbor Discovery, fel, MTU m.m.
IdP (Identity Provider) – Identitetsleverantör (t.ex. Entra, ADFS).
IDS/IPS – Intrångsdetektering/-prevention i nätverk.
IGP/EGP – Routing inom ett AS (OSPF/IS-IS) respektive mellan AS (BGP).
Immutability (backup) – Skrivskyddad retention; skydd mot ransomware.
iSCSI – Blocklagring över IP (SAN).
Intune – Microsofts UEM/MDM-plattform.
IP – Internet Protocol; adressering och leverans av paket.
IP-adress – Identifierar ett nätverksinterface i IP-nät.
IP Source Guard – Skydd mot IP-spoofing på accessportar.
IPFIX/NetFlow – Flödesmätning (käll/dest, portar, bytes).
IPsec – Krypterade tunnlar (site-to-site/klient-VPN).
IPv4 – 32-bitars IP; begränsat adressutrymme.

202

IPv6 – 128-bitars IP; stort adressutrymme och modern funktionalitet.
IS-IS – Länkstatusrouting (IGP), alternativ till OSPF.
JSON/YAML – Dataformat vanliga i API/automation.
Journald – systemd-loggtjänst på Linux.
JumpCloud – Moln-identitet/UEM för bl.a. Linux/Windows/macOS.
Kerberos – Biljettbaserad autentisering (AD/FreeIPA).
Kernel – Operativsystemets kärna.
Keyring (GPG) – Nyckelring för signering/validering av paket/repo.
Knox (Samsung) – Samsungägt ramverk/portal som ger extra MDM-djup.
KVM – Linux-hypervisor (KVM/QEMU) för VM-drift.
L2-switch – Växlar ramar via MAC-tabell inom samma broadcastdomän.
L3-switch – Switch med routingfunktioner mellan VLAN/subnät.
LACP – Länksammanslagning (802.3ad) för redundans/kapacitet.
LACP fast rate – Snabbare LACP-handtryck (~1 s intervall).
LAPS – Hantering av unika lokala adminlösenord (Windows).
Latens – Fördröjning från sändare till mottagare.
Layer 2/3/7 – OSI-lager: Datalänk/Network/Applikation.
LC/SC/MPO – Vanliga fiberkontakter (duplex/multifiber).
Link-local – Lokalt giltig adress (IPv4 APIPA; IPv6 fe80::/10).
Linux bridge – Programvarubrygga; virtuell switch i Linux.
Live migration – Flytt av VM mellan värdar utan driftstopp.
LLDP/CDP – Protokoll som visar grannar och portinformation.
LLM (språkmodell) – AI-modell som kan assistera felsökning/runbooks.
LTS – Långtidsstödd version av ett OS.
LUKS – Linux-diskkryptering (dm-crypt).
MAC-adress – Unik datalänksidentifierare (L2).
MAC-flapping – MAC växlar port ofta; tyder på loop/länkproblem.
MACsec – L2-kryptering på länknivå (802.1AE).
MAN – Metropolitan Area Network; stadsnät.
Management plane – API/CLI/SNMP-kanal för styrning/konfig.
MAB (MAC Authentication Bypass) – 802.1X-alternativ för “dumma” klienter.
MECM/SCCM – Microsofts klienthantering lokalt (f.d. SCCM).
MFA – Multifaktorautentisering (t.ex. lösenord + app).
Mesh (topologi) – Nät med flera redundanta vägar mellan noder.
Microsegmentering – Finmaskig policy per applikation/roll i stället för grova VLAN.
MIMO/MU-MIMO – Flerantennteknik i WLAN (flera samtidiga strömmar).
MPLS – Etikettbaserad forwarding; VPN/QoS i operatörsnät.
MSTP – STP-variant med multipla instanser per VLAN-grupp.
MTTR – Genomsnittlig återställningstid (Mean Time To Repair/Restore).
MTU – Maximal ramstorlek på länknivå.
MSS – Maximal TCP-nyttolast utan fragmentering.
NAC – Nätverksåtkomstkontroll (802.1X/MAB/posture).

203

NAT – Adressöversättning mellan privata och publika adresser.
NAT64/DNS64 – Överbryggar IPv6-klienter till IPv4-resurser.
NAT-T – IPsec över UDP för att passera NAT.
ND (Neighbor Discovery) – IPv6-grannprotokoll (ersätter ARP).
NDES – Microsofts SCEP-endpoint för certifikat via Intune.
Net-ID – Nätverksdelen i en IP-adress (prefix/mask).
NETCONF/RESTCONF – Modellbaserad konfig via YANG (SSH/HTTP).
Netplan – YAML-baserad nätverkskonfiguration i Ubuntu.
NetworkManager – Linux-tjänst/verktyg för nätverkskonfiguration.
NFS – Filresursprotokoll, främst Linux ↔ Linux.
Nginx – Webbserver/reverse proxy.
Node exporter – Agent som exponerar Linux-mätvärden för Prometheus.
NPTv6 – Prefixöversättning i IPv6 (utan portöversättning).
NTP/chrony – Tidssynkronisering i nät/operativsystem.
NTP-stratum – Nivå i tidssynkhierarki (avstånd till referens).
OOB (Out-of-Band) – Separat hanteringsnät, används även vid kris.
OpenSCAP – Efterlevnadskontroll mot säkerhetsprofiler.
OpenTelemetry – Standard för insamling av spår/logg/metrics.
Overlay/Underlay – Logiskt nät ovanpå fysisk fabric (t.ex. VXLAN över IP).
OSPF – Länkstatusrouting (IGP) i interna nät.
OSPF area – Hierarkisk uppdelning; Area 0 är ryggraden.
OSPF cost – OSPF-metric; baseras ofta på bandbredd.
OT (Operational Technology) – Produktions-/styrsystemsnät.
Packer – Bygger automatiserade OS-avbilder.
PAM – Pluggbar autentiseringsram i Linux/Unix.
PAT – Portbaserad NAT (många interna → en publik).
PE/CE – Provider Edge/Customer Edge i operatörsnät.
PFsense/OPNsense – BSD-brandväggar för routing/NAT/VPN.
PHY – Fysiska lagret: kablage, modulering, optik.
PKI – Infrastruktur för nycklar/certifikat (CA, CRL, kedjor).
PKCS (Intune) – Direkt certifikatutfärdande till enheter/användare.
Polkit – Policy för privilegier i Linux (höjda rättigheter).
Port (L2) – Fysisk/logisk anslutning på switch/router.
Port forwarding – Vidarebefordran av portar från utsida till insida.
Portfast – STP-optimering för accessportar (snabb upp).
PoE – Ström över Ethernet till t.ex. AP/kameror.
PPPoE – Inkoppling över Ethernet (vanligt på WAN/operatör).
Prefixdelegation (PD) – Automatisk IPv6-prefixutdelning (CPE → LAN).
Private VLAN (PVLAN) – L2-isolering inom samma VLAN.
Prometheus/Alertmanager – Mätinsamlare och larmmotor.
Proxmox VE – Debian-baserad hypervisor (KVM/LXC) med ZFS-stöd.
Proxmox Backup Server (PBS) – Backup-lösning i Proxmox-ekosystemet.

204

Proxy (reverse) – Front som tar emot klienttrafik och vidarebefordrar till backend.
PXE – Nätboot av klienter (DHCP/TFTP).
QoS – Prioritering/klassning av trafik för bättre upplevelse.
QoS shaping/policing – Buffra jämnt vs hårt strypa trafikflöden.
RA (Router Advertisement) – IPv6-utdelning av prefix/standardväg.
RA-guard – Skydd mot falska IPv6-RA på access.
RADIUS – Central autentisering/auktorisering (t.ex. 802.1X).
RBAC – Rollbaserad behörighetskontroll för administratörer.
REST API – Webbaserat API (HTTP/JSON) för automation/integration.
Restic – Krypterande, deduplicerande backupklient.
RJ45 (8P8C) – Vanlig kontakt för koppar-Ethernet (twisted pair).
RMM (Remote Monitoring & Management) – Fjärrövervakning/klienthantering.
RPO – Hur mycket data som får förloras vid återställning.
RRM (Radio Resource Management) – Automatisk kanal/effekt-optimering i WLAN.
RTO – Hur snabbt system ska vara uppe efter fel.
RIP – Enkelt distans-vektor-routingprotokoll (mindre nät).
RSTP – Snabbare spanning-tree (802.1w).
Route summarization – Aggregera prefix för mindre routingtabeller.
Routing – Framledning av paket mellan nät/subnät (L3).
Router – Enhet som kopplar samman flera nät (L3-gräns).
RSSI/SNR – Signalstyrka och brusförhållande i trådlöst.
SASE/SSE – Säker nätarkitektur där skydd/åtkomst läggs i molnkant.
SBOM – Förteckning över programvarans komponenter (supply-chain).
SCEP – Förenklat protokoll för certifikatutfärdande (via NDES/Intune).
SDA (Software-Defined Access) – SDN i campus; policyn vid kanten.
SDN – Programvarustyrt nät; policy/intention via controller och API.
SD-WAN – Policy-styrt WAN över flera länkar/operatörer.
Security baseline – Minimikrav för konfiguration och säkerhet.
SELinux – MAC-säkerhet i Linux (RHEL-familjen).
sFlow – Provtagning av trafik för flödesanalys (alternativ till NetFlow).
SIEM – Central logg/säkerhetsanalys (korrelation och larm).
SLA – Service Level Agreement; avtalad tjänstenivå.
SLAAC – Automatisk IPv6-adressättning via RA (stateless).
SLO – Mätbar målnivå för en tjänst (t.ex. latens/anslutning).
SFP/SFP+/QSFP – Modulplatser/optik i switchar/routrar.
SMF/MMF (OS2/OM3/OM4) – Single/Multi-mode-fiber och deras klassningar.
SNAT/DNAT – Käll- och destinations-NAT.
SNMP – Övervakningsprotokoll för status/konfiguration.
Snapshot – Ögonblicksbild av data/VM.
SPAN (port mirroring) – Kopiera trafik till analysport (t.ex. Wireshark).
Spanning-tree guards – BPDU-/root-/loop-guard mot L2-fel.
SR-IOV – Delar upp NIC i virtuella funktioner för hög prestanda.

205

SSH – Krypterad fjärrterminal och filöverföring (scp/sftp).
SSL/TLS – Kryptering av applikationsprotokoll (t.ex. HTTPS).
SSID – Nätverksnamn som en AP annonserar i ett **WLAN**.
Stateful/stateless brandvägg – Med eller utan sessionskännedom.
Sticky MAC – Lås MAC-adress till en accessport.
Storm control – Begränsa broadcast/multicast-stormar.
Subnät – Del av nät med eget prefix/mask.
Switch – L2-enhet som kopplar ihop värdar i samma LAN.
Syslog – Standard för loggtransport (UDP/TCP/TLS).
Systemd – Init-/tjänstehanterare i moderna Linux.
TACACS+ – Autentisering/auktorisering för nätverksenheter.
Taggning (802.1Q) – VLAN-identifiering i Ethernet-ramar.
Terraform – "Infrastructure as Code" för moln och nät.
TFTP – Enkel filöverföring (UDP); används bl.a. vid PXE.
Throughput/Goodput – Total genomströmning vs faktisk nyttolast.
TLS – Krypteringsprotokoll för säker kommunikation.
TPM/UEFI Secure Boot – Plattformsskydd och nyckelmodul i klient/servrar.
TOTP – Engångslösenord i MFA-appar.
Trunkport – Switchport som bär flera VLAN (802.1Q-taggning).
Trunk pruning – Tillåt endast nödvändiga VLAN på en trunk.
TWT (Target Wake Time) – Strömsparfunktion i 802.11ax (WLAN).
UDLD – Upptäcker enkelriktade länkar för att undvika loopar.
UDP – O-tillförlitligt transportprotokoll (låg overhead, realtid).
UDM/UEM – Samlad enhetshantering för alla plattformar (endpoint-fokus).
UEFI Secure Boot – Startskydd som verifierar bootkomponenter.
UFW – Enkel brandväggsfrontend i Ubuntu/Debian.
ULA – Unika lokala IPv6-adresser (ej internet-routade).
Uplink – Överordnad anslutning (access → distribution → core).
UTM/NGFW – "Nästa generations" brandvägg (applikationsmedveten).
VLAN – Logisk L2-segmentering i ett LAN.
VM – Virtuell maskin (OS i hypervisor).
VPN – Krypterad tunnel mellan site/klient och nät.
VRRP – Standardiserat FHRP för redundant gateway.
vSwitch – Virtuell switch på hypervisor/värd.
VSS (Windows) – Volymskuggkopior; app-konsistens vid backup.
VXLAN – L2-overlay över IP/UDP (24-bitars VNI).
WAF – Web Application Firewall (lager-7-skydd).
WAN – Wide Area Network; nät över större avstånd.
Wazuh – Öppen plattform för säkerhet/övervakning/EDR.
WDS/MDT – Windows-avbildning/utrullning lokalt (om inte Autopilot används).
Wi-Fi (Wi-Fi Alliance) – Certifieringsprogram/varumärke för interoperabilitet enligt IEEE 802.11.

Många operativsystem benämner trådlösa nätverksgränssnitt som "Wi-Fi". I boken använder

206

vi WLAN för själva tekniken (802.11) och reserverar “Wi-Fi” för certifieringen.

WLAN – Trådlöst lokalt nät (OS kallar ofta gränssnittet “Wi-Fi”).

WMI – Windows Management Instrumentation.

WMI-filter – GPO-filtrering baserat på systemegenskaper.

WPA2/WPA3 – Säkerhetsstandarder för **WLAN**-kryptering.

WPA2-Enterprise – 802.1X/EAP-baserad **WLAN**-autentisering.

WRED – Köhantering som droppar tidigt för att förebygga kollaps.

WUfB – Windows Update for Business; ringstyrda uppdateringar.

WireGuard – Modernt VPN-protokoll (snabbt, enkelt).

Wireshark – Protokollanalysator för nätverkstrafik.

ZFS – Avancerat filsystem/volymhantering (checksumma, snapshot, replika).

Zero Touch/ZTP – Automatisk provisionering/registrering utan handpåläggning.

Zero Trust – Åtkomst beslutas per begäran utifrån identitet, enhetens skick och kontext.

Zigbee/Z-Wave – Lågströmsprotokoll för IoT (hem/byggnad).

207

Appendix II - CLI-REFERENS

Struktur: Linuxkommandon (ip/ss/mtr) presenteras först i denna referens, följt av Windows- och Ciscoexempel; rubriker harmoniseras för lättare uppslag.

Kodsnipplets att använda

Sammanfattning

När vi jobbar med nätverkskonfiguration handlar mycket av arbetet om planering, kontroll och felsökning. Är det något som inte fungerar, kontrollera paketets väg, från början till slut för att se var det har blivit fel.

Ta ALLTID för vana att placera ut saker organiserat och döp allting med exempelvis ip-adresser eller korrekta hostnames. Även det kommer göra felsökningen mycket enklare.

Många av de kommandon som används skrivs sällan ut som hela kommandon, då du kan använda förkortningar istället. Ofta när du söker hjälp på google m.m. så kommer förkortningarna upp. Exempel på detta är när du ansluter till en port eller skall sätta på en port. I de fallen skrivs bara förkortningar till kommandot ut

Den första fråga du får när du ansluter till enheten är: *Would you like to enter the initial configuration dialog? [yes/no]*, där svarar vi ALLTID no.

Att använda sig av ett frågetecken, ?, listar alla tänkbara kommandon som finns på nivån eller som fortsättning på kommandot du matat in. Det är bra att använda sig av om du är osäker.

För att avbryta pågående kommando trycker du **Ctrl+Shift+6**

Det sista tipset är att alltid ansluta en dator till enheten som du skall konfigurera. Detta för att minimera risken att du råkar göra ändringar i portinställningarna på enheten.

Lycka till

Grundläggande CLI-kommandon

enable

Används för att logga in i enheten som administratör. Om du har konfigurerat användarnamn och lösenord får du fylla i det efteråt.

syntax:

enable

Exempel: Router>enable

208

reload

Används för att starta om en enhet. Tänk på att alla kommandon du har lagt in kommer att nollställas, om du inte har sparat dem.

syntax:

reload
Exempel: Router>reload

show

Är ett “grundkommando” som används för att visa inställningar och/eller anslutna enheter till både switch och router.

syntax:

Exempel: Router>show [vad du vill visa]
Exempel: Router>show interfaces status
Exempel: Router>show running-config

exit

Används för att backa ur den nivå du befinner dig i. Om du har konfigurerat en port, och vill logga ut från det läget är detta kommandot du använder dig av

syntax:

exit
Exempel: Router (config-if)# exit

configure terminal

Ändrar enhetens konfiguration, exempelvis portar m.m. Här behöver du vara för att göra alla inställningar.

syntax:

configure terminal eller
conf t
Exempel: router#configure terminal eller
Router#conf t

209

hostname

Vi arbetar med många enheter i nätverket och något som är viktigt är att ha koll på vad de heter. Generellt döps routrar till R, och nummer i ordningen, till exempel: R1. Switchar döps på samma sätt och heter SW istället

syntax:

hostname [namn]
Exempel: Router (config)#hostname R1

interface

Används för att ansluta till en port och göra inställningar, brukar också skrivas *int*. Här är det vanligt att använda förkortningar för porten istället för portens hela namn.

syntax:

interface [portnamn]
Exempel: Router (config)#interface GigabitEthernet0/0 eller:
Router (config)#int g0/0

shutdown

Används för att stänga av en port. Skrivs no framför blir det omvänt, dvs no shutdown blir: sätt på porten

syntax:

shutdown eller sh
Exempel för att stänga av: Router (config-if)#shutdown eller:
Router (config-if)# sh

Exempel för att sätta på: Router (config-if)#no shutdown eller:
Router (config-if)# no sh

ip address

Används för att tilldela en IP-adress till porten. Vanligt att förkorta

syntax:

ip address [ip-adress subnätmask]eller ip ad [ip-adress subnätmask]
Exempel: Router (config-if)#ip address 192.168.0.1 255.255.255.0 eller:
Router (config-if)#ip ad 192.168.0.1 255.255.255.0

210

ip route

Används för att stänga skapa en statisk route, det vill säga hur trafiken går från en routers nätverk till en annan routers nätverk. Tänk på att du oftast måste göra en statisk route åt båda håll så att datorerna i det andra nätverket kan svara också.

syntax:

ip route [nätverksadress subnätmask next_hop]

Exempel: Router (config)#ip route 192.168.0.0 255.255.255.0 10.0.0.1

dhcp

Används för att skapa en dhcp-utdelning av ip-adresser. Den skapas i flera steg. Det första du anger är poolens namn. En pool är alla ip-adresser i ett subnät. Det andra du anger är nät-id för subnätet samt nätmask. Tredje saken är vilken gateway som nätverket skall ha. Avslutningsvis kan du (men är inget krav), lägga in DNS-serverns adress

syntax:
Router (config-if)#ip dhcp pool [poolnamn]
Router(dhcp-config)#network [nät-id] [subnätmask]
Router(dhcp-config)#default-router [gateway-ip]
Router(dhcp-config)#dns-server [dns ip]
Exempel:

Router (config-if)#ip dhcp pool polen
Router(dhcp-config)#network 192.168.0.0 255.255.255.0
Router(dhcp-config)#default-router 192.168.1.1
Router(dhcp-config)#dns-server 8.8.8.8

Precis som med ip-adresser, i ett VLAN, på en multilayerswitch, går det även att upprätta dhcp på en L3 switch. Kommandona är de samma, men du måste vara i VLAN-config-läge (se mer under rubriken VLAN)

dhcp-avancerat - Ranges

Om du använder dig av DHCP kan du vilja ta bort vissa adresser eller ranges

syntax:

Router(dhcp-config)# ip dhcp excluded-address [start-IP] [slut-IP]
Exempel:

211

Router(dhcp-config)# ip dhcp excluded-address 192.168.0.2 192.168.0.9

dhcp-avancerat - MAC-reservation

Om du använder dig av DHCP kan du vilja låsa en specifik mac-adress mot en viss ip

syntax:

Router(dhcp-config)# host [IP-adress] [subnetmask]

Router(dhcp-config)# client-identifier [MAC-adress]

Exempel:

Router(dhcp-config)# host 192.168.0.1 255.255.255.0

Router(dhcp-config)# client-identifier 0100.1A2B.3C4D.5E6F

VLAN

Skapar ett virtuellt nätverk på en port, så kallade sub-portar. VLAN standard är att de döps till jämnt 10-tal

syntax:

int [portnamn.vlan-id]

encapsulation dot1q [vlanid]

Exempel: Router (config)#int g0/0.10

Router(config-subif)# encap dot1q 10

Om du utför detta på en multilayerswitch behöver du inte skapa vare sig subport eller encapsulation. Där räcker det med att du skapar VLAN och tilldelar portar

syntax:

[vlan-id]

name [vlan-name]

Switch(config)#vlan 10

Switch(config-vlan)#name vlan-name

Exempel: Switch(config)#vlan 10

Switch(config-vlan)# name vlan10

Därefter måste du fortfarande tilldela portar (enligt nedan), ip adresser, dhcp m.m.

212

switchport

Används på switch för att hantera VLAN. Porten ställs antingen som access eller trunk. En trunk-port har hand om all trafik från samtliga VLAN och är ofta den port som leder tillbaka till routern. Access är den port som enheter ansluter till.

syntax:

switchport mode [typ av anslutning]

Exempel trunk: Switch(config-if)#switchport mode trunk

Exempel access: Switch(config-if)#switchport mode access

Switch(config-if)#switchport access vlan 10

Skall du göra flera portar samtidigt kan du använda kommandot int range [startport-slutport]

ip routing

Vill du låta din multilayerswitch hantera trafiken måste du aktivera intern routing i den

syntax:

ip routing

Exempel: Switch(config)#ip routing

Kommandon i Windows

ipconfig

Kommando för att se dina nätverksanslutningar i datorn. Den visar alla olika, ethernet, wlan, bluetooth m.m. Alla med en enkel översikt över ip adress m.m.

ipconfig /all

Kommando för att se dina nätverksanslutningar i datorn men, **all information**. Den visar alla olika, ethernet, wlan, bluetooth m.m. Alla med en enkel översikt över ip adress m.m.

ipconfig /release

Kommando för att släppa den ip-adress du har blivit tilldelad

ipconfig /renew

Kommando för att begära ny ip-adress

213

ipconfig /displaydns

Visar innehållet i DNS-cache

ipconfig /flushdns

Töm innehållet i DNS-cache

nslookup

Ger information om domännamn och ip-adresser

Syntax:

```
nslookup google.com
```

ping

Skickar ett ICMP-paket till en specifik ip adress, i syfte att kontrollera anslutningen

Syntax:

```
ping 192.168.0.1
```

tracert

Visar hur nätverkstrafiken routras till en viss ip. Du kan även (om du är ansluten till en DNS) skriva in en URL (webbadress)

Syntax:

```
tracert 192.168.0.1
```

Kommandon i Linux (Debian)

ip -brief addr
Kommando för att snabbt se aktiva nätverksgränssnitt och deras IP-adresser (kompakt översikt).

ip addr show
Kommando för att se dina nätverksgränssnitt i detalj: IPv4/IPv6-adresser, prefix, flags m.m.

ip link show
Visar länkstatus, MAC-adresser och MTU per interface (lager 2-vy).

214

ip route show

Visar routningstabellen (standardgateway, statiska rutter m.m.).

Syntax:

```
ip route show
ip route get 192.168.0.1
```

dhclient -r

Kommando för att släppa (release) din DHCP-lease på ett interface.

Syntax:

```
sudo dhclient -r -v eth0
```

dhclient

Kommando för att begära en ny DHCP-lease (renew) på ett interface.

Syntax:

```
sudo dhclient -v eth0
```

resolvectl status

Visar DNS-resolvrar och cache-statistik om **systemd-resolved** används.

Syntax:

```
resolvectl status
resolvectl query google.com
```

resolvectl flush-caches

Tömmer DNS-cache (gäller systemd-resolved; annars töm respektive cache-tjänst).

dig

Ger detaljerad information om domännamn och IP-adresser (ersätter ofta nslookup).

Syntax:

```
dig +short google.com
dig -x 192.168.0.1 +short
```

ping

Skickar ICMP-ekon till en adress för att testa anslutning och rundtid.

tracepath

Visar vägen (hop för hop) som paket tar till en destination, utan root-krav.

mtr

Kombinerar ping och traceroute i realtid (fördröjning/förluster per hop). Kan behöva installeras (`sudo apt install mtr`).

Syntax:

```
mtr google.com
mtr -6 2607:f8b0:4003:c00::6a
```


215