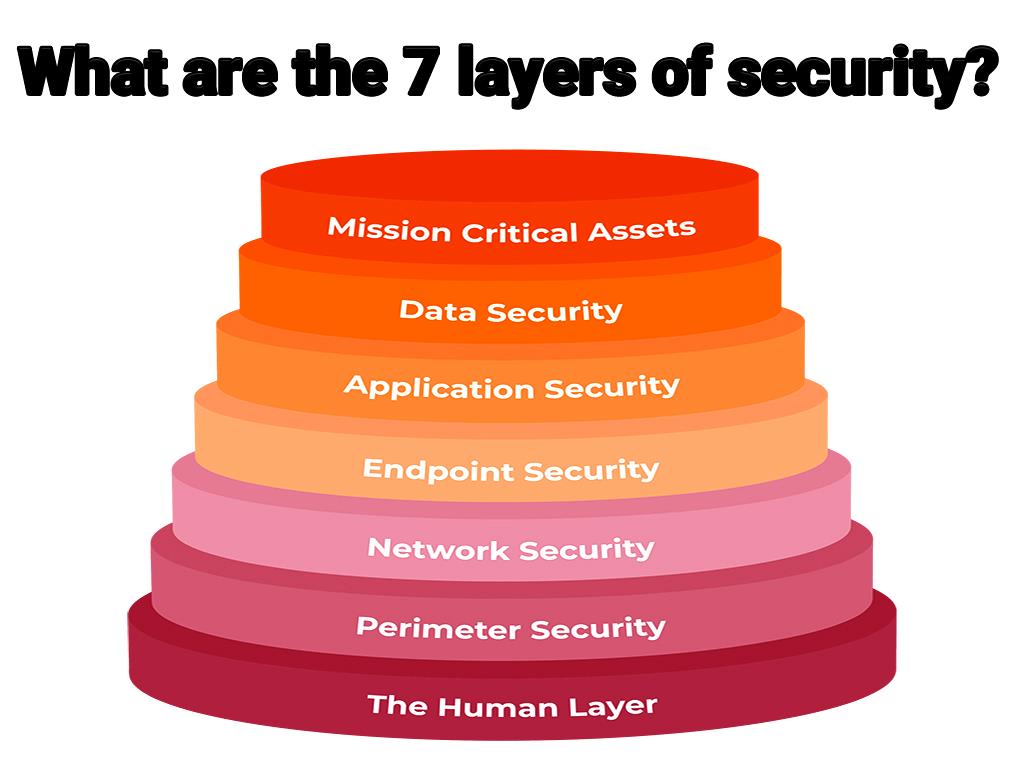
# Cyber security – en introduktion

Cyber security – ett begrepp för att beskriva säkerhet i olika automatiserade system och deras kringliggande miljöer – bygger på en modell där vi ser VAD vi ska skydda som den innersta/översta mest grundläggande definitionen. På/under denna påförs lager/skal av olika sätt, beroende på dels vad hot kan tänkas vara, men också vilken teknik som berörs av hotet.

Inom branschen beskrivs modellen på respektive sätt beroende på kontexten. Nätverkstekniker och ingenjörer med en tradition ur datakom försöker ofta att likna Cybersäkerhetsmodellen med den – för dem – redan kända OSI-modellen (Open Systems Interaction). Ur andra säkerhetsbundna traditioner (bl.a. t.ex. bevakningsbolag, banker etc.) ritas normalt sett samma modell ut fastän som ”en kärna med skal”

Vi kommer att – i den här kursen – använda följande illusatration:



Dessa tekniker skapar olika ”lager” av säkerhet – där en del kommer att bli rutinmässiga, vissa är perimeterorienterade, vissa är skyddande av själva källkoden osv.

Men för att ta sig an några av de vanligaste teknikerna kommer vi idag att koncentrera oss på två ”huvudprinciper” – att använda både perimeter- och process- och källkodsmässigt, för att skydda oss och den miljön vi arbetar i:

## Miljösegmentering och kryptologiska skydd.

### Miljösegmentering:

Med miljösegmentering menas att utvecklingsmiljön är (fysiskt och/eller logiskt) skiljd från andra miljöer – vad det gäller servrar, nätverksinfrastruktur, kanske fysiska arbetslokaler etc. Detta för att säkerställa att OM en säkerhetsincident skulle uppstå i utvecklingsmiljön (vilket är vanligt förekommande – inte minst för att den just är skapad för att utveckla nya tjänster och produkter och att detta ofta sker under tidspress) inte ska kunna sprida sig till andra miljöer (så som produktionsmiljön) eller vice versa.

### Kryptologiska skydd:

Kryptologiska skydd används inom ”säkerhet ” i en mängd olika funktioner – checksummering av filer (för filintegritetsskydd), ”attacksignaturer” i IP-intelligence-databaser i EWS (Early Warning Systems) etc.

Vi ska gå igenom alla de här termerna och titta på dem i lite mer detalj under dagens föreläsning och se några exempel på praktiska lösningar som utvecklats enligt de här olika grundprinciperna.

## Miljösegmentering i praktiken:

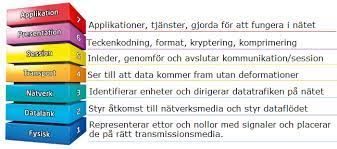
När man pratar om miljösegmentering pratar man ofta om att man segmenterar det nätverk som behövs för sin utvecklingsmiljö från andra (delar av) nätverk på en arbetsplats. T.ex. isolerar man en utvecklingsavdelnings avdelningsnätverk från ekonomiavdelningens, från driftavdelningens, från produktionen, från marknadsavdelningens etc. Detta är vad man på IT-säkerhetsspråk ofta kallar för ”nätverkssegmentering”.

Inom informationssäkerhetsområdet i ett större perspektiv brukar man också prata om ”datasegmentering” – d.v.s. den data som faktiskt finns i dokument, källkod etc. när man pratar om ”informationsklassning” (även detta kommer det en kommande lektion om).

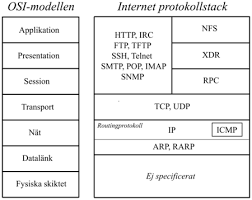
Just nu – men bara precis just nu, på den här lektionen, kommer vi (av tidsskäl) sätta ett (provisoriskt) likhetstecken mellan ”Miljösegmentering” och ”Nätverkssegmentering”, eftersom det är en del av miljösegmenteringen man ofta koncentrerar sig på när man bygger ”säkra utvecklingsmiljöer”. En säker nätverksmiljö har alltså större krav på sig än att *bara* segmentera nätet – men det är en god grund för sin informationsklassning och/eller övriga informationssäkerhet (och kommande datasegmentering) att börja prata ”segmentering”.

### Nätverkssegmentering:

För att förstå nätverkssegmentering – och hur det faktiskt ökar säkerheten måste vi se att det sker på flera nivåer i datakommunikationen mellan två enheter på nätverket. För att bättre förstå kommer vi därför att snabbt återbesöka OSI modellen och dess olika lagers olika funktioner:



Och tittar vi på Internettrafik (där vi använder TCP/IP-stackens olika delprotokoll skulle denna kunna förklaras (snabbt) i följande bild.



Nätverkssegmentering kan sägas ske på två nivåer enligt dessa bilder. Dels på ”nätverksnivå” – alltså nivå 3 – där vi ju hittar IP (som just addresserar noderna i nätet – och i och för sig fragmenterar datat – men det är inte en funktion i IP som är intressant för vår nätverkssegmentering) och dels på ”datalänknivå” (där vi har kontrollen för enheternas access till det fysiska nätverket).

Det gör att vi kan ”segmentera” vårt nät i två led: både fysiskt (genom att koppla in olika saker som servrar och klienter i olika nätverkshårdvara) men också logiskt (genom att att addressera dessa i olika adressrymder, och därför styra trafikvägvalen – routingen – genom nätet).

Segmenteringen kan alltså ses som varandes fysisk eller logisk – eller ofta både ock – samtidigt.

Nätverkssegmentering kan underlätta inte bara säkerhetsarbetet, utan även nätverksdriften. Detta kan lätt förstås när man inser att ju färre enheter som finns inom ett nätsegment – desto mindre är risken för att trafikstörningar, som kastade paket, kollisioner etc.

Vi ska emellertid titta på nätverkssegmentering ur perspektivet ”It-säkerhet för utvecklare” i den här kursen – alltså: Vad kan vi vinna i säkerhet på att segmentera nätet på olika sätt (logiskt eller fysiskt) och vad eller vilka av dessa vinster är användbara för oss just när vi pratar ”utvecklingsmiljö”.

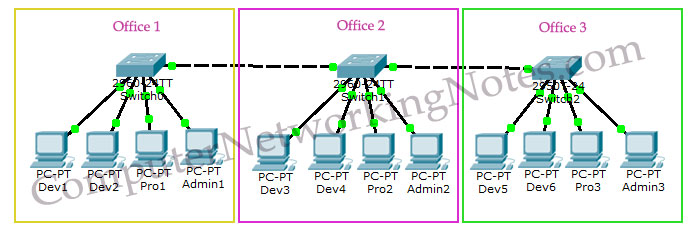
Vi ska titta på ett litet youtube-klipp om just segmentering av nätet.

<https://youtu.be/wZuS_jawlyA>

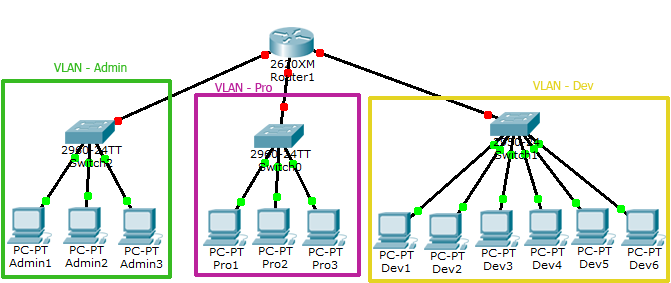
#### Något kort om VLAN (Virtual LAN)

VLAN-tekniken kom till – egentligen inte för att öka säkerheten på ett nätverk, utan för att bättre använda switchad teknik, för att dela upp switchens portar i olika logiska ”underswitchar”. Det sker i switchens mjukvara. Det gör att switchen egentligen har ett ”switchinterface” för varje konfigurerat VLAN, för att identifiera vilka noder som är kopplade ”i vilken port”. För att dessa VLAN ska kunna sträckas ut – över t.ex. flera geografiska platser – behövs antingen direktlänkar (seriellt) mellan switcharna, eller att trafiken routas.

I det senare exemplet kopplas VLAN ID (I switcharna) med samma IP-adress-rymd (IP-nät) och trafiken kan sträckas ut över flera switchar, på flera geografiska platser, men på samma sätt som om alla satt i samma rum.



Så att man får en logisk struktur som ser ut såhär:

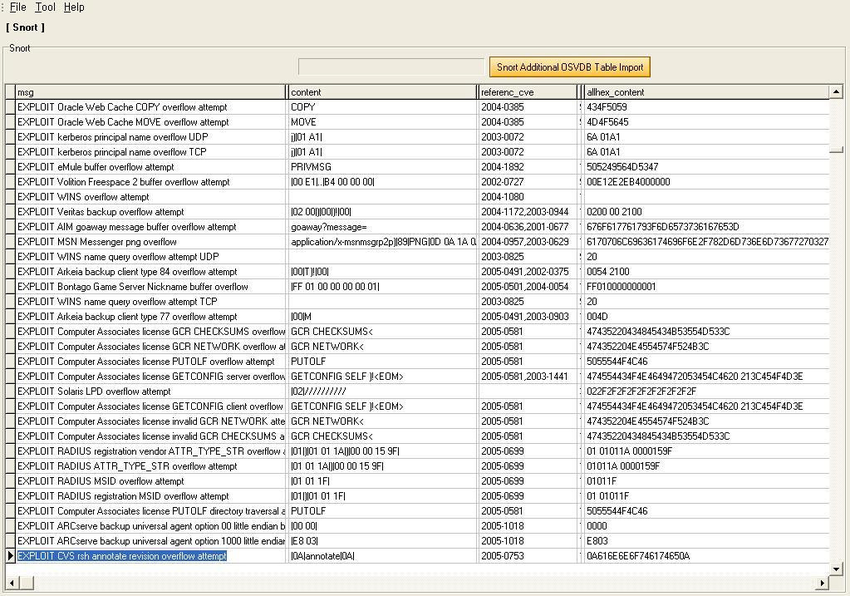


## Kryptologiska skydd:

När vi pratar kryptologiska skydd kan dessa förekommer på flera olika lager i vår säkerhetsmodell som vi pratade om i början på lektionen.

Ofta pratar man om olika sorters implementation av olika kryptotekniker utifrån till vad de används till för att skydda. Användningsområdena kan tyckas oändliga – men vi pratar normalt sett om åtminstone tre huvudtyper av använingsområden när vi pratar ”säker utvecklingsmiljö” – nämligen ”nätverkssäkerhet” – där vi använder själva kryptofunktionen för att kryptera datat som går i nätverket. Detta kan göras antingen genom särskilda kryptoprotokoll (som TLS) – eller att man ”tunnlar” ett protokoll i ett krypterat för att skapa ett VPN (Virtual Private Network).

Det andra huvudexemplet är för att kontrollera någon form av ”integritet” av data (antingen filer, såsom dokument och/eller källkod och/eller liknande)



Det tredje huvudexemplet är ”spårning och detektering”, där t.ex. vissa trafikmönster kan krypteras (och då säkerställas integritetsmässigt) och sedan jämföra dessa trafikmönster med annan trafik, för att kunna detektera huvuriva ett trafikmönster (så som en pågående attack) upprepar sig (och då ”slå på stora larmklockan”.

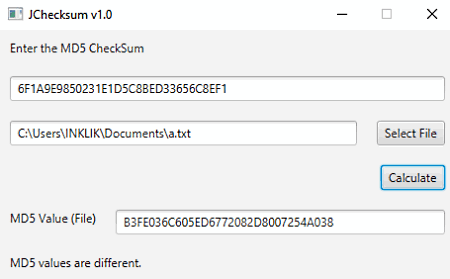
Låt oss gå igenom dessa – var och ett för sig:

Vad det gäller nätverkssäkerhetsimplementationerna av kryptologiska skydd kan egentligen inte sägas så mycket i hänseendet ”säker utvecklingsmiljö” annat än att det inte är ovanligt att man ofta har något sorts VPN mellan sin egen arbetsstation och den miljö man utvecklar i. Detta är för att trafiken – mellan klient och server inte ska kunna ”sniffas upp” (IP signalerar ju all trafik i klartext) av en illasinnad och missbrukas på något sätt.

VPNet ger ”ett säkert rör” att ladda upp/ned filer, editera data, logga in på ”säkra servrar” etc.

Filintegritetscheckar (och liknande) bygger på att man vill kontrollera och förvissa sig om att en fil (så som en konfigurationsfil, ett program, källkod, dokumentation etc.) inte har förändrat sig. Det går till så att man drar hela texten i filen (stdout) genom något slags linjär – symmetrisk – krypteringsalgoritm och får då ut en ”one way hash” – en checksumma på filen. Denna kan sedan lagras (eller skrivas ned) i en lista (eller i en databas, eller liknande) och man kan se att den inte ändrats mellan tillfället den sparats och tillfället man gör en ny check – och därigenom verifiera att datat har bibehållit sin integritet.

Idag bygger t.ex. alla ”säkra” mjutvecklingsverktyg på just olika integritetschecker av datat. Filer som distribueras genom t.ex. github och liknande har alltid med en checksumma (eller flera) samt berättar vilken algoritm som används/använts för att beräkna checksumman.



Vi ska nu titta på ett system för att kontrollera just filintegritet – genom att kolla på en liten videofilm, för att se hur det kan se ut i verkligheten.

<https://www.youtube.com/watch?v=fhPGt2Q4PkM>

(För den gravt intresserade finns många bra artiklar och filmer om det här på nätet – men jag gillar den här artikeln: <https://eng.libretexts.org/Bookshelves/Computer_Science/Operating_Systems/Linux_-_The_Penguin_Marches_On_(McClanahan)/12%3A_Linux_Systems_Security/6.22%3A_Integrity_Check_Checksums)>

Det tredje och sista exemplet – att detektera olika mönster i t.ex. nätverkstrafik – kan sägas vara en utbyggnad av – eller variant på just ”integritetskontroll”. Man spelar helt enkelt in vissa attackmönster – ser vilken data som förändras (eller vilka filer/data/trafikmönster man kan ordna en ”baseline” på att jämföra med) och sedan kan signaturen från attacken jämföras med det kontrollerade systemet. Den vanligaste applikationen av den tekniken är olika sorters anti-virus-skydd, som lagrar ”signaturen” av vanliga virus i en klient/agentbaserad databas. Detta kan givetvis också användas för andra sorters fel och brister – för att detektera konfigurationsfel/förändringar etc.

Idag finns stöd för ”attacksignaturer” hos alla applikationsproxyprogramvaror (FWNG, Antivirus, Endpoint protection etc.) och dessa är idag inte längre företagsunika, utan delas mellan olika företag i branchen. Att dessa kan delas, och läsas och användas av flera olika sorters mjukvara, kallas för ”threat intelligence”.