5. SYSTEM SECURITY MECHANISMS

Magnus Jensen

- Motivation
- Trusted Computing Base
  - Secure Enclave
    - *Apple A-Chip*
- Firewalls
  - *Packet Filtering*
  - *Proxy Firewall*
  - *Statefull Firewall*
- Malware og forsvar
  - Virus skannere
    - Intrusion Detection
    - Dommerkendelse
- Sikkerhedspolitik
  - Lattice
    - *Rettighedseksempel*
    - *Brug af lattice*
    - Bell-Lapadula
- Modeller
  - Chinese Wall
  - Dual Control
- Access Control
  - Access Control Lists
  - User Capabilities

SYSTEM SECURITY MECHANISMS

1. Motivation

Når vi har et system og snakker om sikkerhed, så betyder det vi ønsker følgende:

- **Eksterne angribere** må ikke komme ind i systemet
- **Interne parter** skal ikke kunne manipulere med systemet

Vi vil kun give adgang til systemet, til de der **overholder vores sikkerhedspolitik**.

2. Trusted Computing Base

Vi kan lade **en del af systemet, bestemme** hvem der skal have adgang til hvad og sørger for at regler bliver overholdt.

Dette kaldes en **Trusted Computing Base**. Det er en sikker del af systemet, der **ikke kan ændres ude fra** og hvis **beslutninger** er endelig.

2.1 Secure Enclave

En hardware beskyttet del af en chip, der kan udføre operationer og holde små mængder af data, kaldes en Secure Enclave.

> ***Secure Enclave**: hardware beskyttet del af chip*
>
> - *Kan køre software med bestemt signatur*
> - *Kan holde private-keys*
> - *Kan holde signature*
> - *Køre program og signere output*

Alt sammen, **så man ved en handling blev udført korrekt**.

2.1.1 Apple A-Chip

Et eksempel på en Secure Enclave, er den som sidder i Apples A-chips serie; der blandt andet står for at **godkende operationer baseret på ens biometriske aftryk mm.**

Denne chip har været omtalt en del i medierne, da den har været omdrejningspunkt i flere terror sager i USA; hvor efterforskningstjenester **ikke har kunnet få adgang til vigtig data pga chippen;** som ej heller Apple selv vil kunne udtrække.

3. Firewalls

Okay så store dele af **bogen handler om** hvordan vi opnår **sikker kommunikation**; men for at kan gøre det, **kræver** det at vi **starter** i **usikker kommunikation**; og dette kan udnyttes til at angribe os.

Herfra kommer ideen om Firewalls: **en person kan ikke bryde ind i en computer han ikke kan snakke med**. Kort sagt sørger en firewall for at stoppe de dele af trafik på et netværk som vi ikke ønsker.

> ***Firewall**: stopper dele af trafik på netværk*

3.1 Packet Filtering

Kommunikationen over internettet består af forskellige pakker. Det simpleste type Firewall, bruger Packet Filtering, som simpelt kigger på disse pakker; og **bestemmer om de skal komme igennem eller ej**.

> ***Packet Filtering**: Simpel filtrering*
>
> - *Sort / hvid sortering*

Det kunne f.eks være at **stoppe alle pakker der prøve at åbne en ny TCP forbindelse**.

Men packet filtering er ofte meget sort eller hvidt; og er ikke nødendigvis hvad man ønsker; vi vil gerne have noget der analyser de forskellige packets bedre og mere intelligent.

3.2 Proxy Firewall

Det er her Proxy Firewalls kommer ind i billedet. **Der simpelt vil overtage en forbindelse på vegne af maskinen.**

Altså vil en client forbinde til en firewall, der så forbinder til den eksterne server.

Udefra vil det altså kun se ud som om, at det er firewallen som netværket består af.

Det kan ses som at **bruge et skjold om hele sit netværk** - og firewallen skal hackes igennem før computerne på netværket kan blive ramt.

> ***Proxy firewall**: skjold om sit netværk*
>
> - *Pro: Skjold for mange*
> - *Con: Software skal skrive specielt til det*

3.3 Statefull Firewall

En Statefull Firewall holder styr på alle forskellige connections; og holder lige så styr på om der er tilladelse til at lave nye.

> ***Statefull Firewall**: holder styr på alle forbindelser*
>
> - *Afviser pakker der ikke hører til en forbindelse*
> - *Maskere interne adresser*
> - *Lukker mistænkelige forbindelser ned*

4. Malware og forsvar

I tilfældet af, at en angriber kommer **forbi en firewall** og får adgang til vores maskiner; kan det være at der bliver **installeret ondsindet software** på vores computere.

Det kommer i mange variationer:

> - *Trojanske heste*
> - *Viruser*
> - *Orme*
> - *Ransomeware.*

Vi vil gerne undslippe at dette sker, men i tilfældet af at det sker - hvad gør vi så?

4.1 Virus skannere

Det er dette som Virus Skannere gør. Det **holder øje med filerne på computeren** for at kigge efter **dele af kode** der kendes som skadelig eller andre kendetegn, fra en stor database som konstant opdateres.

På det seneste har skadelig software dog forsøgt at forhindre dette ved at **kryptere sig selv**.

4.1.1 Intrusion Detection

En anden måde end at overvåge filsystemet for skadelige dele; og også at overvåge hvad der aktivt sker på en computer.

Dette hedder Intrusion Detection. Dette minder meget om hvad Statefull Firewall også gjorde mulig; nemlig at se på **data-flowet**; og sker der noget man mener er **mistænkeligt**, kan man tage **aktion**.

4.1.2 Dommerkendelse

Der er to måder at bestemme på, om der foregår noget der ikke må: ud fra Regler og ud fra Statistik.

- *Regler*
- *Statestik*

Reglerne betegner en række prebestemte måder der definere normal opførsel; brydes de så skrides der til handling.

Ellers så kan man statistisk kigge på hvad der er god opførsel; og begynder man at se noget der ikke stemmer overens, træde til handling.

- *Honey Pot*

En sidste måde, kaldet "Honey-pot" måden; minder på mange måder om en mussefælde. Altså at lokke til at angribe en fil, og i tilfælde til træde til handling.

5. Sikkerhedspolitik

Der findes modeller for sikkerhedspolitikker og der findes implementeringer for dem.

Implementeringerne er så, sjovt nok det, man kalder for en sikkerhedspolitik. Og en model er en generalisering.

En sikkerhedspolitik består af:

- *Beskrivelse af systemet*
- *Beskrivelse af foranstaltninger*
- *Muligvis high-level tilgang*

Det kunne f.eks beskrive:

- *Hvem der må hvad*
- *Hvilke situationer der toleres*

5.1 Lattice

Til at hjælpe med at angive **præcise beskrivelser** af en sikkerheds politik; kan vi bruge det der kaldes en **Lattice**.

En lattice består af en endeligt set S og en relation ≤.

> ***Lattice***
>
> - *Sæt* ***S*** *- rettighedsroller*
> - *Relation* ***≤***

Elementerne i S betegner forskellige rettighedsroller; og selve ideen ved latticen er at beskrive forholdet imellem rettigheder.

5.1.1 Rettighedseksempel

For elementerne $a, b, c, \in S$ gælder det at:

> - $a \leq b$, *b har mindst lige så mange rettigheder som a*
> - $a \leq b$ *and* $b \leq a$ *betyder* $a = b$
> - $a \leq b$ *and* $b \leq c$ *betyder* $a \leq c$

5.1.2 Brug af lattice

Vi bruger en lattice ved at skrive:

> - *Et* ***subject s med klasseficering C(s) må udføre en operation*** *på object o med klassificering C(o) hvis og kun* ***hvis C(o)≤ C(s)****.*
> - ***Information må flyde fra a til b hvis b ≤ a****.*

5.1.3 Bell-Lapadula

Den første forsøg på at formaliser en sikkerhedsmodel, var Bell-Lapadula modellen; der af **militæret** blev brugt til at betegne hvordan **information skulle flyde; nemlig opad i herakiet**:

> - ***No read up****: S må læse fra objekt o hvis kun C(o) ≤ C(s)*
> - ***No write down*** *S må skrive til objekt o hvis kun C(s) ≤ C(o).*

En anden: **Biba modellen, virker omvendt -** i det den betegner at vigtig information kommer ovenfra og skal flyde ned af.

6. Modeller

Der findes dog mange af disse slags modeller, her benævnes et par stykker:

6.1 Chinese Wall

Et firma har flere forskellige klienter; nogle der tilmed er i konkurrence med hinanden.

> *Et B2B firma*

Protokollen garanter da, at en **ansat i firmaet ikke kan afsløre information og en klient til en anden klient; hvis de er konkurrenter**.

> *Kunder kan være konkurrenter*

Reglen er så, at den ansatte kun kan få information om en klient, hvis han aldrig har fået information om en af klientens konkurrenter.

> *Ansat må få info, hvis den aldrig har fået for en konkurrent*

6.2 Dual Control

Er en model, hvor en handling kun er tilladt hvis flere autoriserede parter har givet tilladelse.

7. Access Control

Det kan være svært at vide hvornår forskellige subjekter må gøre hvad til hvilke objekter. Det er her en access control matrix kommer ind i spillet.

Betegnes ved en matrice A med en række for hver subjekt s og en kollone for hvert objekt o, så betegner A[s, o] en liste af alle operationer s må udføre på o. Det er dog omfangsrigt at have sådan en matrice, så i praktis bruges andre metoder:

7.1 Access Control Lists

For hvert objekt gemmer vi hvem der har rettigheder til det. Det gør det nemt at se hvem der har adgang, men svært at se til hvilke operationer.

7.2 User Capabilities

For hvert subject gemmes mulighederne. Det gør det nemt at vide hvad en bruger må gøre; men ikke på hvad. Windows bruger denne.