IN1020: Løsningsforslag gruppetime 10 (uke 44)

Oppgave 1: Kryptografi

a) Sikre trygg lagring av data i utrygge lagringsenheter, sikre trygg overføring av data i f.eks åpne nett.
b) Tapt eller kompromittert nøkkel gjør det mulig for en angriper å bryte konfidensialitet, integritet og/eller ektehet for informasjon som er beskyttet av denne nøkkelen. En trygg administrasjon av nøkler som beskytter mot kompromittering eller tap av nøkkel er derfor en faktor som i stor grad påvirker styrken i en kryptografisk sikkerhetsløsning.
c) Formålet med PKI (og sertifikater), er å legge til rette for sterk autentisering av entiteter på internett, samt kryptert overføring av informasjon mellom enheter. PKI skal sikre autentiske offentlige nøkler. PKI er et rammeverk for å legge til rette for å binde en offentlig nøkkel til en navngitt enhet, og denne bindingen kan bekreftes av en betrodd myndighet som utsteder et sertifikat. Et sertifikat forteller/bekrefter følgende: "Offentlig nøkkel K eies av X".

Oppgave 2: Symmetrisk kryptering

a) Alle kan lytte til radiosignaler så hvis nøkkelen blir lekket eller koden knekkes kan det være svært alvorlig.
b) Ikke alle har maskiner som kan spille av DVD-er eller kjøre CD-er. De må også transporteres på en måte. Posten er ikke 100% sikkert og dersom du transporterer den selv er det jo nesten litt unødvendig å kryptere.
c) USB-er er små og kan lett mistes. De kan også holde på mye data, som gjør det ekstra ille å miste dem.
d) Som oftest kan man ikke kryptere selve e-posten. Så mottaker kan se avsender, evt. andre mottakere, emnefelt, all tekst som er inne i selve e-posten og navnene på filer man sender med. Det er kun vedlegg (filer) som er kryptert. Dersom du sender e-posten feil eller noen klarer å få tilgang til e-posten er det altså mye de kan lese ut av denne infoen.

Å overlevere nøkler på en trygg og sikker måte er heller ikke uproblematisk, men det bør alltid skje over en annen kanal enn den man bruker for å overlevere de krypterte dataene.

Oppgave 3: Asymmetrisk kryptering

a) Du bruker en offentlig nøkkel for å kryptere. Kollegaen bruker den tilhørende private nøkkelen for å dekryptere. Dette sikrer konfidensialitet.
b) Du bruker en privat nøkkel for å kryptere og kollegaen din bruker den tilhørende offentlige nøkkelen for å dekryptere. Dette sikrer autentisitet og uavviselighet.

Oppgave 4: Hash-kryptering (sjekksumalgoritme)

a) Hash algoritmer “fordøyer” data ned til en streng som har fast lengde (ofte 16, 32 eller 64 bit og i heksadesimal).

b) Dersom to brukere har samme passord, har de den samme hashen. Det finnes databaser med hashene til vanlige passord, så hvis brukerne deres benytter dårlige passord kan trusselaktører søke opp hashene deres i disse databasene og finne en match. Teamet ditt burde egentlig ikke prøve å lagre brukeres passord selv. Benytt heller google eller facebook (eller andre tilbydere av autentiseringstjenester) til å gjøre det for dere.
c) Først og fremst integritet, men også konfidensialitet ved at passord ikke lagres i klartekst og ikke kan leses av uvedkommende.

Oppgave 5: Sjekksumalgoritme (hash-funksjon)

a) Nei, sjekksummen blir en annen når fila endres. Dette er en grunnleggende egenskap ved sjekksumalgoritmer. Men observer gjerne at sjekksummen blir den samme når du bruker programmet sha256sum på samme fil flere ganger.
b) Man kan sammenligne sjekksum av en datafil opp mot en kjent sjekksum av denne fila. Er den lik? isåfall er fila ikke endret mellom første og andregangskjøring av sjekksum.
c) Sjekksumalgoritmer brukes ofte ved deling av programvare (f.eks. nedlasting fra nett). Tilbyderen av programvare genererer en sjekksum av orignalen (ofte et filarkiv, f.eks. .zip-fil), som publiseres sammen med programvaren. Brukere som laster ned programvare kan enkelt selv kjøre en sjekksumalgoritme etter nedlasting, og kan med det finne ut om programvaren er det den utgir seg for å være, eller er den kanskje endret etter at sjekksummen ble generert? Vil også kunne avdekke hvorvidt en nedlasting har gått greit eller ikke (pakketap?). Helt konkret eksempel: https://www.postgresql.org/ftp/source/v15.0/, nedlasting av kildekode PostgreSQL databaseserver. Her finner du både programvaren og tilhørende sjekksummer for både MD5 og SHA256.

Oppgave 6: SSL

Praktisk oppgave uten løsningsforslag.

Oppgave 7: Lagdeling i datakommunikasjon

a) Fysiske lag
Linklaget
Nettverkslaget (IP)
Transportlaget (TCP/UDP)
Applikasjonslaget
b) Fysiske lag - sikre at signaler som kan tolkes som bits (0 eller 1) kan leveres til neste ledd i kommunikasjonen. F.eks. trådløst, eller over en ledning.

Linklaget - sikre at en gitt mengde bits kan deles opp i håndterbare enheter kalt “frames”, og at disse kan sendes over linken til neste mottaker uten å gå tapt eller miste informasjon.

Nettverkslaget (IP) - Levere en datapakke til en annen vert på Internett, ofte på tvers av flere lokale nettverk (LAN).

Transportlaget (TCP/UDP) - ekstra tjenester i tillegg til adressering. TCP gir f.eks. pålitelighet, forbindelsesorientering, meningskontroll, bytestrøm, levering i samme rekkefølge som data ble sendt, feilsjekking og flytkontroll. UDP leverer et minimum av tjenester, og overlater de mer

avanserte tjenestene til å bli implementert av de som skriver applikasjonene. Transportlaget gjør det også mulig å skille forskjellige applikasjoner fra hverandre som mottakere innenfor én IP-adresse (vertsmaskin) ved hjelp av konseptet "porter".

Applikasjonslaget - Alle programmer som bruker lagene under. F.eks. epost (SMTP) og web (HTTP).

c) Applikasjonslaget og Transportlaget.
d) Applikasjonslaget – SSL
Transportlaget – TCPCRYPT (Lite brukt)
Nettverkslaget – VPN (IPSEC)
Linklaget – Spesielt hvis man bruker broadcast-nettverk, som WiFi – WPA2/WPA3

Oppgave 8: Nettverksprotokoller

a) For å håndtere og overlevere metadata som trengs for å levere tjenestene protokollen tilbyr.
b) Header blir lagt til lag for lag, etter hvert som pakken beveger seg gjennom lagene i sendeprosessen.

Eks fra forelesning: data fra applikasjon får en TCP-header, så en IP-header, så en Ethernet-header.

En router (som skal sende pakker ut av et LAN), vil ta av og bytte ut headere for linklaget. Når pakken kommer frem til målet, blir headerene tatt av lag for lag i motsatt rekkefølge av da pakken ble sendt etter hvert som pakken beveger seg oppover i protokollstakken mot applikasjonslaget.