## IDATT2104 - Datakom – Arbeidskrav 2

Oppgaven omhandler Websocket og SSH på applikasjonslaget.

Legg vekt på at skjermklipp fra Wireshark skal være lesbare. Det betyr at utsnittet må avgrenses til å vise kun relevant informasjon.

Rapporten kan utføres individuelt eller i frivillige grupper på inntil tre deltakere. *Grupper med to eller tre deltakere skal dokumentere funksjonene med klient og tjener på to separate maskiner, altså uten å bruke loopback-adresse 127.0.0.0.* Adressebruken skal dokumenteres.

**Websocket**

## Teori:

1. Hva er Hovedforskjell mellom HTTP og Websocket

HTTP er en forespørsels-responsprotokoll som brukes til å overføre data over internett. Det er en tilstandsløs protokoll, som betyr at hver forespørsel behandles uavhengig, uten kunnskap om tidligere forespørsler. HTTP brukes vanligvis for nettlesing, der en klient (vanligvis en nettleser) sender en forespørsel til en server for en bestemt ressurs (som en nettside), og serveren svarer med den forespurte ressursen.

WebSocket er derimot en toveis protokoll som gjør det mulig sanntidskommunikasjon mellom en klient og en server. Den gir en vedvarende tilkobling mellom klienten og serveren, slik at begge parter kan sende og motta data når som helst, uten behov for en ny HTTP-forespørsel. Websocket utnytter full Duplex kontinuerlig over TCP, som betyr at begge sider kan sende data uten å vente på forespørsel fra annen part. WebSocket brukes vanligvis for applikasjoner som krever sanntidsoppdateringer, for eksempel online spill, chatapplikasjoner osv.

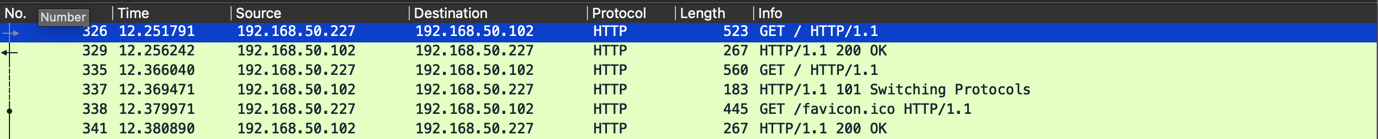
## Hva er sikkerhetsmekanismen i Websocket.

Sikker tilkobling (WSS): WebSocket-protokollen støtter en sikker variant kalt "WebSocket Secure" (WSS), som bruker Transport Layer Security (TLS) for å kryptere data som sendes mellom klienten og serveren. Dette kan bidra til å beskytte dataintegriteten og forhindre avlytting. For å bruke WSS, endres skjemaet i WebSocket-URLen fra ws:// til wss://.

Origin-sjekk: En annen sikkerhetsmekanisme er å verifisere WebSocket-tilkoblingens opprinnelse. Når en klient prøver å koble seg til en WebSocket-server, sender den en "Origin"-header i håndtrykkforespørselen. Serveren bør sjekke denne headeren for å sikre at den kun aksepterer tilkoblinger fra pålitelige kilder. Dette kan bidra til å beskytte mot cross-site WebSocket-hijacking (CSWSH) og andre sikkerhetstrusler.

## Dokumentasjon av hvordan Websocket bruker HTTP for å:

* Utføre handshake mellom klient og tjener:



Figur Handshake gjennom HTTP mellom en klient og tjener. Her er klient 192.168.50.227 og tjener 192.168.50.102

## Hvordan meldinger sendes fra klienter

* Hvordan meldinger sendes fra klienter



Figur Her sender klient (192.168.50.89) en melding til tjeneren (192.168.50.102).

Hvordan tjener sender meldinger til tilknyttede klienter



Figur 3 Her sender tjener (192.168.50.102) melding tidligere mottatt av en klient, til alle tilknyttede klienter (192.168.50.227 og 192.168.50.89)

Dokumentasjon (med Wireshark skjermklipp) av hvordan Websocket bruker HTTP for å:

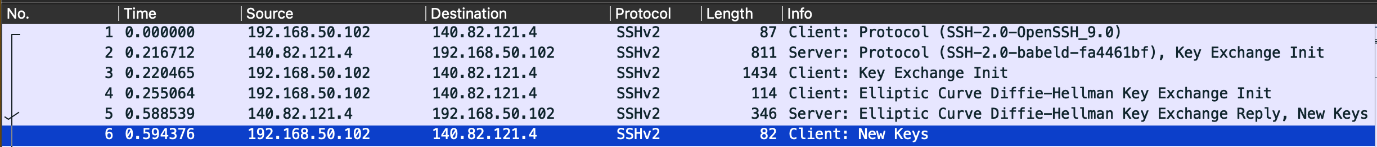
* Utføre handshake mellom klient og tjener
* Hvordan meldinger sendes fra klienter
* Hvordan tjener sender meldinger til tilknyttede klienter

**Continuous Integration & Deployment**

Dere har ikke laget program for SSH, med derimot brukes SSH for CI/CD. Oppgaven er derfor å dokumentere hvordan en kryptert forbindelse etableres. Forklar hvilken informasjon som utveksles og vis dette med skjermklipp fra Wireshark.

## Hvordan etableres en sikker forbindelse (før autentisering)

1. En klient sender en tilkoblingsforespørssel til en server, med innhold om SSH Protokoll versjon, og klient protokoll versjon. (Protocol (SSSH-2.0-OpenSSH\_9.0)
2. Serveren svarer med å sende informasjon om SSH protokoll versjon, og server protokoll versjon. (Protocol (SSH-2.0-babeld-fa4461bf). Deretter vil den også gi sin Public Key til klienten gjennom «Key Exchange Init».
3. Klienten tar imot Public Key fra server, og bruker denne til å kryptere en tilfeldig sesjonsnøkkel.
4. Klienten sender så den krypterte sesjonsnøkkelen til serveren, «Client: Elliptic Curve Diffie-Hellman Key Exchange Init»
5. Serveren bruker sin private nøkkel til å dekryptere sesjonsnøkkelen mottatt fra klienten. Man har da fått opprettet en felles «hemmelig» nøkkel for server og klient. «Server: Elliptic Curve Diffie-Hellman Key Exchange Reply». Denne felles nøkkelen blir laget med den kryptografiske protokollen Diffie-Hellman, som gjør det mulig å opprette felles hemmelige nøkler, over usikre kanaler, uten å måtte dele hemmelige nøkler i forkant.
6. Når disse krypterte sesjonsnøklene er klare til bruk, er autentiseringen over. Klienten og serveren kan da starte å bruke disse til å kryptere data.

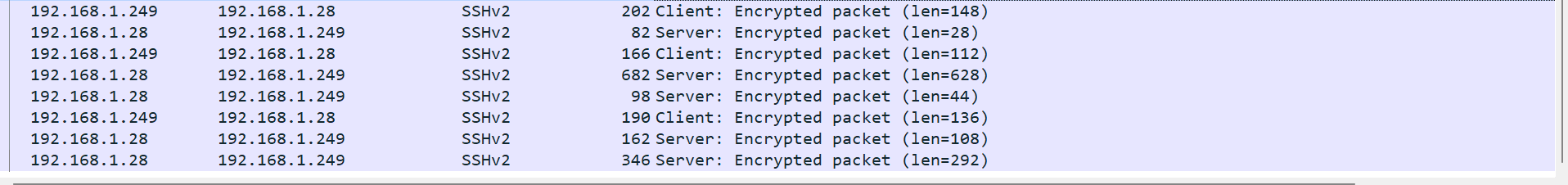


## Hvordan autentiseres brukeren

A picture containing text

Description automatically generated

Figur : ssh meldinger sendt før man skriver inn passord, der .249 er klienten og .28 er serveren



Figur : ssh meldinger sendt etter passord er satt inn, der .249 er klienten og .28 er serveren

Passordet sendes noe plass her og siden oppsettet vårt var gjennom ssh, er passord sending over nettverk kryptert og ikke sendt i klartekst.

Når du bruker passordbasert autentisering med SSH, involverer prosessen for å bekrefte passordet på serversiden vanligvis følgende trinn:

* Klienten starter en SSH-tilkobling til serveren og sender en forespørsel om autentisering ved hjelp av et brukernavn.
* Serveren mottar autentiseringsforespørselen og ber klienten angi passordet.
* Klienten skriver inn passordet og sender det til serveren.
* Serveren mottar passordet og sjekker det mot brukerens passord som er lagret i autentiseringsdatabasen lokalt.
* Hvis passordet stemmer, sender serveren en melding til klienten som indikerer at autentiseringen var vellykket.
* Hvis passordet ikke stemmer, sender serveren en melding til klienten som indikerer at autentiseringen mislyktes.
* Hvis autentiseringen mislykkes, kan klienten bli bedt om å angi passordet på nytt eller tilkoblingen kan bli avsluttet.