# **IDATT2104 - Oblig 1**

# Applikasjonslaget

## **WEB/HTTP**

Denne oppgaven bruker en ferdig Wireshark pakkefangst til analysen. Fila heter *WS-HTTP-Capture* og er publisert sammen med oppgaven.

Velg linje 5 i pakkefangsten, høyreklikk og velg «Follow/TCP stream». Dette viser kommunikasjonen for den første av de samtidige forbindelsene som ble etablert mot webtjener.

1. Undersøk vedvarende forbindelser.   
   Linje 5 fra klient og linje 13 fra tjener viser headerlinjene for å holde en vedvarende forbindelse. Hva sender klient og hva svarer tjener på dette?

* Klient sender: GET HTTP/1.1 (med Connection: keep-alive)
* Tjener svarer: HTTP/1.1 200 OK (med parametere for timeout og maks bytes for den gjeldende forbindelsen)

1. Når ble bildet fra webtjener sist endret? Hvordan kan denne informasjonen brukes til å spare senere nedlasting ved oppslag på samme domenenavn, og hva spør klienten om da i sin forespørsel??

Dersom vi f.eks. ser på bildet «/bitmaps/bok\_datakommunikasjon.jpg» som spørres etter i linje 27 og som blir sendt til klienten på linje 53, så kan vi lese av det siste modifikasjonstidspunktet fra «Last-Modified»:

*Last-Modified: Wed, 06 Jul 2005 11:22:38 GMT\r\n*

Dette gjør at klienten kan bildet med det den allerede har cachet lokalt. Klienten vil da inkludere sin «If-Modified-Since» med et tidspunkt som tjeneren kan sammenligne med, og eventuelt sende den oppdaterte utgaven dersom tjener har en nyere.

1. Hva er stikkord for HTTPS; HTTP/2 og HTTP/3?

* HTTPS

Tilrettelegger for en sikker overføring av HTTP-forespørsler ved hjelp av TLS

Gir HTTP kryptering, for sikker overføring

Port 443 i stedet for 80 (tjeneren vil omdirigere klienter til port 443 dersom de forsøker å benytte seg av port 80)

Overførte meldinger er fremdeles i HTTP/1.1, men de er krypterte vha. TLS

* HTTP/2

Effektivisert bruk av nettverksressurser

Kompresjon av header-felt – mindre krevende overføring

Mulighet for flere samtidige operasjoner/utvekslinger per forbindelse

* HTTP/3

Bygger videre på HTTP/2

Bedre kompresjon av header-felt

QUIC i stedet for TCP (kryptering, kontroll av pakketap og flyt på nettverket)

## **Navnetjenesten/DNS**

Start Wireshark og gjør et oppslag *>nslookup ntnu.no*. Sett display-filter til dns.qry.name==«ntnu.no».

1. Hvilke to typer ressursrecords spørres det automatisk etter?

A og AAAA

1. Hva er time-to-live for disse svarene? Hvorfor er det hensiktsmessig å sette en utløpsverdi?

TTL på A er 587s, mens AAAA er på 294s.

Det er fornuftig med en verdi på TTL fordi ulike tjenere kan ha ulike formål. Én type server kan ha godt av å holde på ressursrecorden lengre enn en annen (i cachen). En lavere TTL vil føre til høyere trafikk på nettverket, men en høyere sikkerhet ifm. kritiske tjenester (da det oppdateres oftere).

1. En annen type ressursrecord er MX for eposttjener. Hva viser svaret *for nslookup -type=MX ntnu.no*?

Det viser oss hvilke epost-tjenere som NTNU har. I dette tilfellet har vi en MX-ressursrecord under mx.ntnu.no.

1. Finn IPv4-adressen til denne eposttjeneren. Hva er adressen, og kan den nås fra Internett?

Vi finner IPv4-adressen ved å kjøre «nslookup mx.ntnu.no» og får da også en IPv6-adresse.

*IPv4: 129.241.56.67*

*IPv6: 2001:700:300:3::67*

Denne adressen kan nåes fra Internett, da dette er en offentlig IPv4-adresse. Vi kan kjøre nslookup i revers og få et svar direkte fra serveren vha. «nslookup 129.241.56.67». Vi fikk ikke direkte kontakt/svar fra serveren i den første nslookup-en, da denne var en «Non-Authorative», som vil si at vi fikk svar fra en server som hadde lagret DNS-oppslaget til denne (som en mellomlagring).

# **Transportlaget**

## **TCP**

Pakkefangst med Wireshark skal også brukes til å undersøke egenskaper ved TCP. Gjør samme filtrering som i oppgave 1

1. 3-Way handshake. Hva er resultatet av de tre første pakkene i pakkefangsten? Hvilke flagg i pakkene benyttes for dette?

I en 3WHS så vil klienten først sende et SYN-flagg, deretter vil tjeneren sende både et SYN og et ACK flagg, før klienten til slutt sender et ACK-flagg.

I den første pakken fra klienten (med SYN-flagget) så sendes det en pakke med et initialisert *sekvensnummer* (tilfeldig 32-bit tall) (kvitteringsnummeret blir ofte satt til 0 her, da det ikke er satt noe ACK-flagg), for å fortelle tjeneren at den ønsker å opprette en forbindelse.

Tjeneren svarer med en pakke med både et SYN- og et ACK-flagg. SYN-flagget markerer at tjeneren ønsker å opprette en forbindelse tilbake (altså «returveien»), mens ACK-flagget settes for å kunne bekrefte at tjeneren har mottatt pakken fra klienten. Det settes altså en verdi som er sekvensnummeret + 1 (for å vise at det er dette som er det neste forventede sekvensnummeret på klientens neste pakke). I tillegg settes det også et sekvensnummer for tjeneren, som klienten kan bruke som en referanse til kvittering av mottatte pakker.

1. Pålitelig overføring

I pakke nr 5 ber klienten om indeks-filen for domenenavnet *datakom.no*. I det påfølgende kommer det en serie pakker for overføring av denne. Det skal undersøkes sammenhengen mellom TCP sekvensnummer, kvitteringsnummer og nyttelast i disse pakkene frem til og med pakke 14.   
OBS: sekvensnummer og kvitteringsnummer er 32 bit tall, men du skal benytte relativt nummer som vises i Wireshark. Fyll inn følgende tabell (grå farge fra klient og gul farge fra tjener)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nr. | Innhold | KLIENT | | | TJENER | | |
|  |  | Sekv.nr | Kvitt.nr | TCP Length/ Payload | Sekv.nr | Kvitt.nr | TCP Length/  Payload |
| 5 | HTTP | 1 | 1 | 619 bytes | - | - | - |
| 8 | ACK | - | - | - | 1 | 620 | 0 bytes |
| 9 | TCP | - | - | - | 1 | 620 | 1460 bytes |
| 10 | TCP | - | - | - | 1461 | 620 | 1460 bytes |
| 11 | TCP | - | - | - | 2921 | 620 | 1460 bytes |
| 12 | TCP | - | - | - | 5841 | 620 | 1460 bytes |
| 13 | HTTP | - | - | - | 6958 | 620 | 1117 bytes |
| 14 | ACK | 620 | 6958 | 0 bytes | - | - | - |

Kontroll:

1. Skriv opp og regn ut summen av sekvensnummer og nyttelast i pakke 10 og sammenlikne denne med sekvensnummer i pakke 11.

Sekvensnummer før = 1461

Nyttelast/payload: 1460

Neste sekvensnummer (i linje 11) = 1461 + 1460 = 2921

Alle bytes ble overført.

1. Sammenlikne summen av TCP nyttelast fra tjener og siste kvittering fra klient.

Tjener total nyttelast = 6958 (siste sekvensnummer) – 1 (første sekvensnummer) = 6957 bytes

Siste kvittert mottatt byte fra klient: 6958

Alt ok her.

1. Hva er det da sekvensnummer og kvitteringsnummer forteller oss?

Sekvensnummeret forteller hvor mange bytes som har blitt sendt hittil, mens kvitteringsnummeret forteller oss hvilke byte fra den motstående siden som ble mottatt sist.

## **TLS**

HTTP/1.1 benytter TLS for å etablere en sikker forbindelse. Start Wireshark og gjør et oppslag på NTNU. Bruk display-filter *tls.handshake.extensions\_server\_name==»ntnu.no»*  
Finn den første «Client Hello» fra NTNU. Denne pakken tilbyr et sett av krypteringssuiter for kommunikasjonen. Tjener velger en av disse og sender beskjed i «Server Hello».

1. Hva heter den suiten som tjeneren velger?

Cipher Suite: TLS\_AES\_128\_GCM\_SHA256 (0x1301)

1. Hvilke to metoder inneholder denne suiten for kryptering og hashing?

* Kryptering: AES (Advanced Encryption Standard)
* Hashing: SHA256 (Secure Hashing Algorithm 256-bit)

## **Digitale sertifikater**

1. Digitalt sertifikat
   1. NTNU-web bruker et digitalt sertifikat for autentisering og sikker kommunikasjon. Hvilken *signaturalgoritme* er brukt i dette sertifikatet for å lage sertifikatets fingeravtrykk?

Det er brukt SHA256 for å lage et fingeravtrykk.

* 1. Hvordan brukes signaturalgoritmen for å lage sertifikatets fingeravtrykk?

Den beregner en sjekksum av en melding, som blir helt unik for det gitte sertifikatets data.

* 1. Hvordan kan mottaker av sertifikatet kontrollere at sertifikatet er ekte, dvs ikke forfalsket?

Sertifikatets gyldighetstid (fra og til)

I tillegg vil vi kunne sjekke egen sjekksum av sertifikatet (fingeravtrykket) opp mot sjekksummen av fingeravtrykket med utsteders offentlige nøkkel. Dersom vi sammenligner disse to og ender opp med kun 1-er bits så har vi et gyldig sertifikat.

# Nettverkslaget

**Undersøkelser av konfigurering på egen PC på campus**

1. Kjør en ipconfig /all og ta et skjermklipp. Hva er *nettverksadressen* til eget IPv4-Subnett?

Vi finner nettverksadressen ved å for eksempel sjekke egen IPv4-adresse med subnettmasken som filter.

IPv4-adressen er: 10.22.3.45

Subnettmasken er: 255.255.252.0

Dette vil si at vi har (8+8+6) = 22 bit til nettverksadressen (fordi vi fjerner to bits fra plass nr. 3 for å gå fra 255 ned til 252 i subnettmasken). Vi har da altså et skille ved bit nr. 22, der de resterende 10 går til nodeadresser (2^10 = 1024 nodeadresser).

Nettverksadressen vår er altså 10.22.0.0. (Fordi vi går fra 10.22.0.0 til 10.22.3.255 for å få tilgang til 1024 nodeadresser).

1. Er IPv4-adressen til DNS, DHCP og default Gateway del av eget IP-subnett?   
   Begrunn svaret.

Vi vet altså at vår nettverksadresse (vårt IP-subnett) ligger innenfor 10.22.0.0 til 10.22.3.255. Da kan vi altså sjekke om DNS, DHCP og Default Gateway har mellom 0 og 3 på plass 3.

DNS: 129.241.0.200 (og 129.241.0.201)

DHCP: 129.241.0.206

Default Gateway: 10.22.0.1

Vi kan altså se at det kun er Default Gateway som er en del av IP-subnettet jeg er tilkoblet. Dette kan vi enkelt se da både DNS og DHCP starter med 129.

1. Finn din egen offentlige IPv4-adresse ved f.eks. oppslag på <https://whatismyipaddress.com/> . Hvilken konklusjon kan vi trekke av dette?

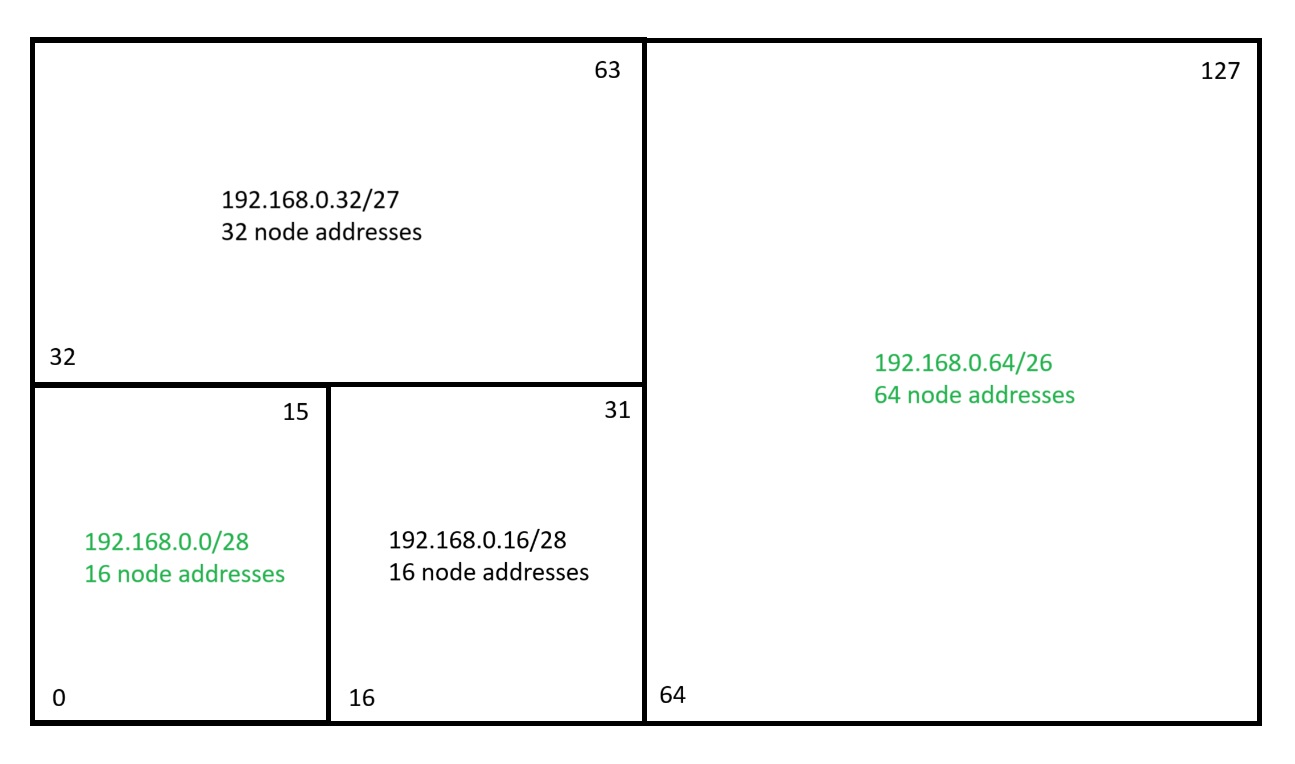
Her ser vi en helt annen IP-adresse enn den vi ser ved «ipconfig». Adressen vi ser på nettsiden er vår «offentlige» IP-adresse, noe som er representert via ruteren vår. Den har en adresse som starter på 129.241.X.X. Dette er den offentlige IP-adressen som ruteren har når den kommuniserer/befinner seg på Internett.

1. Frivillig undersøkelse *på Mobil*: Finn mobilens offentlige IPv4-adresse mens du er koplet på Eduroam. Slå deretter at wi-fi og prøv på nytt. Hvilken konklusjon kan du trekke av dette?

### **Subnetting med variable lengde på subnettmasken (VLSM)**

1. Bruk «firkantmodellen» fra forelesning til å illustrere hvordan et /25-nettverk kan deles i et /26-nettverk og et /28 nettverk som følger kravet at /28-nettverket skal ha det laveste adresserommet. Det skal klart gå frem av illustrasjonen hvilke subnett som finnes totalt.

Dersom vi har et /25-nettverk (de siste 7 bytesene er ledige til nodeadresser mens de første 25 brukes til nettverksadressen), så vil vi kunne ha 2^7=128 ledige nodeadresser som enheter kan koble seg på i IP-subnettet vårt (teknisk sett 126, da vi mister 1 til IP-kringkastingsadressen og 1 til selve nettadressen).



Vi ser her at vi delt opp /25-nettverket i to /26, før vi videre deler den ene /26-en opp i to /27, før vi deretter deler opp den ene /27-en til to /28-nettverk. Vi ser da på subnet 1 og 4 (de grønne). Det kan kanskje være mulig å endre det slik at subnett 4 har nodeadressene 16 til og med 80 (ettersom at vi kanskje ikke ønsker å bruke subnett 2 og 3), men dette gjorde jeg ikke i illustrasjonen.

1. List opp følgende opplysninger for det spesifiserte /28-nettverket gitt at det opprinnelige /25-subnettet har nettverksadresse 192.168.0.0

Nettverksadresse : 192.168.0.0

Laveste nodeadresse : 0

Høyeste nodeadresse : 15

Kringkastingsadresse : 192.168.0.15

1. List opp de samme opplysningene for /26-nettverket

Nettverksadresse : 192.168.0.64

Laveste nodeadresse : 64

Høyeste nodeadresse : 127

Kringkastingsadresse : 192.168.0.127

Tar her utgangspunkt i illustrasjonen jeg lagde.

# Funksjoner **(følgende oppgaver er testet hjemme, ikke på campus)**

## ARP

ARP (Address resolution Protocol) brukes til å finne kopling fra IPv4-adresser til MAC-adresser på noder innenfor et subnett. Dette lagres dynamisk i en arp-tabell på PC.

1. Beskriv hva ARP i hovedtrekk gjør og vis dette med et skjermklipp i Wireshark.

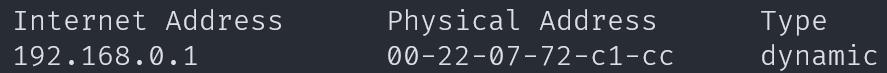
ARP gjør at vi får en oversikt over andre enheter sine MAC-adresser (gitt at de er innenfor det samme subnettet vår enhet er tilkoblet). Dette tilrettelegger for at vi kan sende pakker direkte ved å bruke mottakerens MAC-adresse i stedet for å måtte gå fullstendig via ruteren. Vi holder oss altså «lokalt».

sEt bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Utklippet fra Wireshark viser at vi har en ARP-forespørsel der enheten vår spør ruteren om det finnes en enhet innenfor samme subnett med IP-adressen 192.168.0.79. Dersom det gjør det, kan ruteren gi et svar som inneholder denne enhetens MAC-adresse. Denne blir da lagt til å vår enhet sin ARP-tabell for eventuelt senere bruk.

1. List ut din lokale IPv4 arp-tabell (>arp -a) og identifiser minst en kjent IP-adresse listet som dynamisk.



Her ser vi den dynamiske adressen til vår ruter sin Default Gateway. Denne har en MAC-adresse som vi har lagret i vår lokale ARP-tabell, og som brukes til å kommunisere med ruteren via vår enhet sitt nettverkskort.

## TRACEROUTE

Bruk Traceroute til vg.no *(>tracert vg.no*) mens Wireshark kjører. Sett displayfilter ICMP eller ICMPv6.

1. Hva er verdi på TTL/Hop Limit i svaret fra første ruter (default gateway) og siste ruter?

TTL på svaret fra Default Gateway er på 64 sekunder:





TTL på svaret fra den siste ruteren er på 245 sekunder:

