34334 Hjemmeopgavesæt 1

Daniel Brasholt s214675

Oktober 2022

Opgave 1: DNS, TCP og TLS

Spm 1.1 Meddelelse 1-6

Når brugeren tilgår `http://www.dtu.dk` ved computeren endnu ikke hvilken adresse, den skal tilgå. De første 6 beskeder i `PCAP`-filen er da DNS-opslag og svarene på opslagene. Der er både svar fra en `A`-record og fra en `AAAA`-record. Forskellen er blot, at `A`-record er til en IPv4-adresse og en `AAAA`-record er til en IPv6-adresse.

For DNS-forespørgsel på A-record får klienten en besked tilbage om en CNAME-record, som siger, at `www.dtu.dk` bor på `external8.sitecore.dtu.dk`. Derudover står der i svaret, at `external8.sitecore.dtu.dk` bor på adressen `192.38.84.35`.

For DNS-forspørgsel på AAAA-record får klienten igen besked om (gennem CNAME-record), at siden bor på `external8.sitecore.dtu.dk`. Dog er der ikke en adresse med i svaret. Klienten sender så en forespørgsel mere (besked 5) om adresse, men svaret kommer tomt tilbage, da siden ikke har en AAAA-record.

Spm 1.2 Meddelelse 7-13

Efter browseren ved hvilken adresse, `http://www.dtu.dk` bor på, kan en forbindelse til serveren oprettes. Dette gøres med en TCP-forbindelse. Besked 7-9 er da TLS 3-way handshake med `SYN`, `SYN, ACK`, `ACK`. Herefter anmoder browseren om hjemmesiden med en `HTTP-GET`-request (meddelelse 10). Denne kvitteres med en `TCP-ACK` (meddelelse 11). Serveren svarer da med en `HTTP 301`-status, som betyder permanent redirect (besked 12). Denne indeholder også den nye `URL`, som browseren bedes tilgå - `https://www.dtu.dk/`. Browseren bliver altså bedt om at tilgå samme side, men med et `TLS`-handshake. Besked 13 er TCP-forbindelsen, der kvitterer med en `ACK`.

Spm 1.3 Antallet af TCP-forbindelser

Browseren henter ikke kun data fra hjemmesiden med én TCP-forbindelse. Sættes filteret `tcp.flags.syn == 1`, kan man se, at der i alt oprettes 7 TCP-forbindelser, som det kan ses på figuren nedenfor. 3 af disse oprettes parallelt i besked 77-79 og serveren svarer på dem alle 3 i besked 80, 82 og 84. Man kan endvidere se, at den første TCP-forbindelse bliver oprettet på port 80; men efter browseren bliver henvist til `https://`-versionen af siden, bliver de resterende henvendt port 443, som bliver brugt til TLS.

No.	Time	Source	Destination	Protocol	Length	Info
7	0.014260	10.0.2.15	192.38.84.35	TCP	74	58118 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=574562561 TSecr=0 WS=128
8	0.029024	192.38.84.35	10.0.2.15	TCP	60	80 → 58118 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
14	0.305799	10.0.2.15	192.38.84.35	TCP	74	53284 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=574562853 TSecr=0 WS=128
15	0.314933	192.38.84.35	10.0.2.15	TCP	60	443 → 53284 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
77	1.656030	10.0.2.15	192.38.84.35	TCP	74	53288 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=574564203 TSecr=0 WS=128
78	1.658624	10.0.2.15	192.38.84.35	TCP	74	53290 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=574564205 TSecr=0 WS=128
79	1.661879	10.0.2.15	192.38.84.35	TCP	74	53292 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=574564209 TSecr=0 WS=128
80	1.666977	192.38.84.35	10.0.2.15	TCP	60	443 → 53288 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
82	1.675093	192.38.84.35	10.0.2.15	TCP	60	443 → 53290 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
84	1.675247	192.38.84.35	10.0.2.15	TCP	60	443 → 53292 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
87	1.685250	10.0.2.15	192.38.84.35	TCP	74	53294 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=574564232 TSecr=0 WS=128
92	1.692848	192.38.84.35	10.0.2.15	TCP	60	443 → 53294 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
94	1.693315	10.0.2.15	192.38.84.35	TCP	74	53296 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=574564240 TSecr=0 WS=128
144	1.708094	192.38.84.35	10.0.2.15	TCP	60	443 → 53296 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

Figur 1: Meddelelser med `SYN`-flaget sat. Figuren viser oprettelsen af 7 TCP-forbindelser.

Spm 1.4 TLS-handshake

1.4.a)

I den første Client Hello-besked, der er i PCAP-filen, sender klienten TLS-versionerne 1.3 og 1.2 under supported versions. Kigger man under den tilsvarende Server Hello, kan man se, at serveren vælger version 1.2.

1.4.b)

Et TLS-handshake består af et TCP 3-way handshake; en ClientHello; en ServerHello med certifikat og ServerHelloDone; en ClientKeyExchange med ChangeCipherSpec og Finished; og en ChangeCipherSpec med Finished fra serveren. I PCAP-filen svarer dette til pakkerne 14 til 32. 14-16 er TCP handshake, 17 er Client Hello, 19 Server Hello, 27 ServerHelloDone og 31 den sidste ChangeCipherSpec. Pakke 32 er da ACK for den modtagne ChangeCipherSpec. Pakke 14 bliver sendt ved $time = 0.305799$ og pakke 32 afsendes ved $time = 0.353664$. TLS-handshake (hvis TCP-handshake inkluderes) har da taget $0.353664s - 0.305699s = 0.047965s \approx 48ms$. Medregnes ikke TCP-handshake tog det ca. $36ms$.

1.4.c)

`HTTP GET /` bliver sendt i pakke nr. 10. Dette bliver gjort efter DNS-opslag er udført, men før TLS-handshake er påbegyndt. Dette er eftersom klienten først forsøger at tilgå `http://www.dtu.dk`, men så bliver bedt om at tilgå den sikre version ved `https://www.dtu.dk` i stedet, hvorefter TLS-handshake bliver udført.

1.4.d)

Andet TLS-handshake tager blot $20ms$ fra Client Hello til sidste ChangeCipherSpec-besked. Denne kan instantieres hurtigere, da en ny TLS-forbindelse kan genbruge parametre fra TLS-sessionen.

1.4.e)

Den første TLS-forbindelse bliver termineret ved at klienten sender en Encrypted Alert i besked 1507. Derefter bliver TCP-forbindelsen også nedlagt ved at klienten først sender en `FIN, ACK` i besked 1508 og serveren sender sin egen `FIN, ACK` i besked 1533. Forbindelsen er officielt termineret efter klienten kvitterer med en `ACK` i besked 1534.

Opgave 2: 802.11

Spm 2.1 Beacon

2.1.a)

Det access point, der broadcaster beacon-rammerne med for SSID 34334, hedder: `Tp-LinkT_7d:8e:7a`. Dette kan ses ved hver Beacon Frame under `IEE 802.11 Beacon Frame -> Source Adress`.

2.1.b)

Beacon-intervallet kan findes ved hver Beacon Frame under
`IEE 802.11 Wireless Management -> Fixed parameters -> Beacon Interval`.
For SSID 34334 er beacon-intervallet cirka $0.205s$.

2.1.c)

Ved hver Beacon Frame under
`802.11 radio information -> Channel`
står kanalen, som access pointet fungerer på, angivet. For AP, der broadcaster SSID 34334, er dette kanal 1.

Spm 2.2 Authentication

2.2.a)

En authentication frame bliver sendt fra access pointet ved frame nummer 1266. Dette var som svar til den probe request og authentication frame, som blev afsendt fra enheden ved navn `XiaomiCo_ee:11:f9` i frame 1261 og 1264. Authentication fra access point bliver sendt cirka ved $time = 16.4$.

2.2.b)

Kigger man i den authentication frame, der blev sendt, under `radio information`, kan man se, at dataraten var $1.0 Mb/s$.

2.2.c)

Der udveksles ingen nonces under denne authentifikation. Disse bliver først udvekslet senere.

Spm 2.3 Associering

2.3.a)

Association Request og Association Response sker i ramme 1268 og 1270. 1268 er enheden, der sender en request til access point. 1270 er access point, der besvarer anmodningen.

2.3.b)

Både klienten og access point har 4 datarater markeret som understøttede: 1, 2, 5.5 og 11 $Mbit/s$

Spm 2.4 Four Way Handshake

2.4.a)

Protokollen, der bliver brugt til at gennemføre 4 way handshake, er 802.1X EAPOL.

2.4.b)

Wireshark har markeret et 4-way handshake med EAPoL i rammerne 1273, 1284, 1287 og 1298, startet af access pointet. Disse bliver sendt over cirka $20ms$.

2.4.c)

En nonce indenfor kryptografi er en værdi, der kun bliver brugt 1 gang. I EAPOL bliver 2 nonces brugt; en ANonce (Access point Nonce) og en SNonce (Station Nonce) brugt sammen med en delt hemmelighed til at forhandle en PTK[1].

2.4.d)

En replay counter bliver talt op, når AP og klient sender beskeder under 4-way handshake. Pointen med dette er at forsøge at opdage replay attacks, hvor gamle beskeder bliver gensendt. Frame 1 og 2 vil da have replay counter 1, mens 3 og 4 har replay counter 2.

Opgave 3: Intrusion Detection

Spm 3.1 IP adresse

Følgende kommando kan bruges til at finde IP adressen på server:

```
sudo ip netns exec server ip addr
```

Dette giver følgende output:

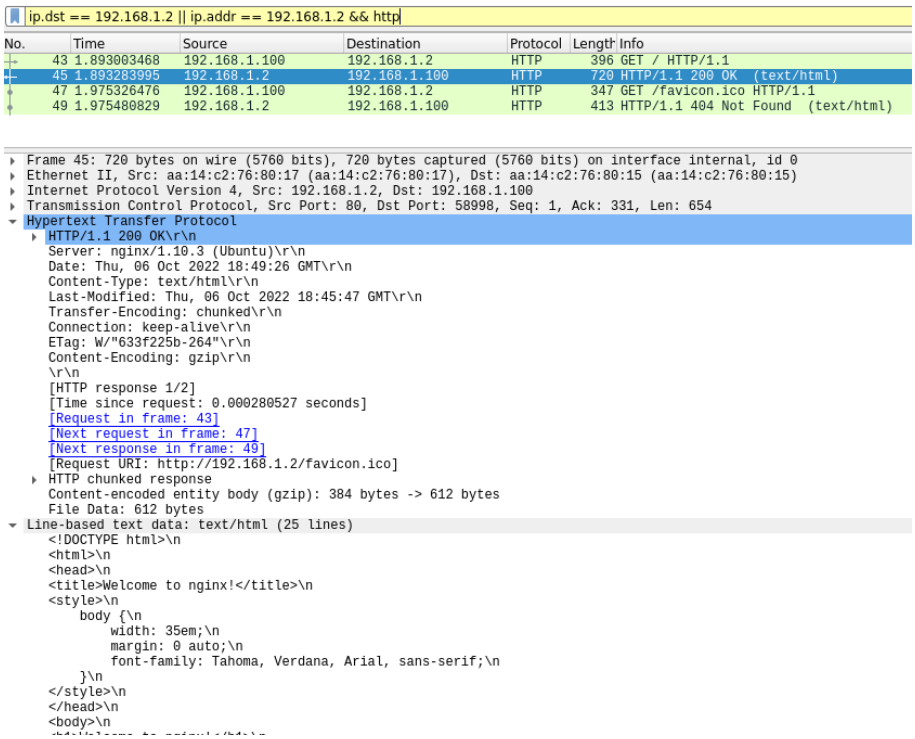
[1] Ifølge slides en "Pairwise Temporal Key"; sagde mange online resourcer "Pairwise Transient Key".

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
   valid_lft forever preferred_lft forever
59: internal@if58: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    noqueue state UP group default qlen 1000
link/ether aa:14:c2:76:80:17 brd ff:ff:ff:ff:ff:ff link-netns internal
inet 192.168.1.2/24 scope global internal
   valid_lft forever preferred_lft forever
```

Her kan IP adressen på "server" aflæses til `192.168.1.2`.

Spm 3.2 HTTP-beskeder fra webserveren

Følgende figur 2 er screenshot fra Wireshark, mens det lytter på "snort" namespace. Figuren viser en HTTP GET-request og et svar fra server indeholdende HTTP status 200 og sidens HTML. Derefter spørger browseren med endnu en GET-request om `/favicon.ico`, men denne besvares med en `404 Not Found`:



Figur 2: Screenshot fra Wireshark, der viser HTTP-beskeder efter siden `192.168.1.2` er besøgt.

Spm 3.3 Alarmer fra SNORT

Nedenstående figur 3 viser output fra /var/log/snort/alert, altså de alarmer, der blev aktiveret, da en browser tilgik webserveren på adressen vist i afsnit Spm 3.1.

```
[**] [1:100000001:0] Min regel 2 [**]
[Priority: 0]
10/05-20:11:35.174465 192.168.1.2:80 -> 192.168.1.100:58934
TCP TTL:64 TOS:0x0 ID:61463 IpLen:20 DgmLen:706 DF
***AP*** Seq: 0x888F579  Ack: 0x991B804B  Win: 0x1FB  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4003073413 3324541002

[**] [1:100000002:0] Min regel 3 [**]
[Priority: 0]
10/05-20:11:35.174465 192.168.1.2:80 -> 192.168.1.100:58934
TCP TTL:64 TOS:0x0 ID:7389 IpLen:20 DgmLen:706 DF
***A**** Seq: 0x888F579  Ack: 0x991B804B  Win: 0xFA80  TcpLen: 32

[**] [1:100000001:0] Min regel 2 [**]
[Priority: 0]
10/05-20:11:35.261410 192.168.1.2:80 -> 192.168.1.100:58934
TCP TTL:64 TOS:0x0 ID:61465 IpLen:20 DgmLen:399 DF
***AP*** Seq: 0x888F807  Ack: 0x991B8164  Win: 0x1F9  TcpLen: 32
TCP Options (3) => NOP NOP TS: 4003073500 3324541089
```

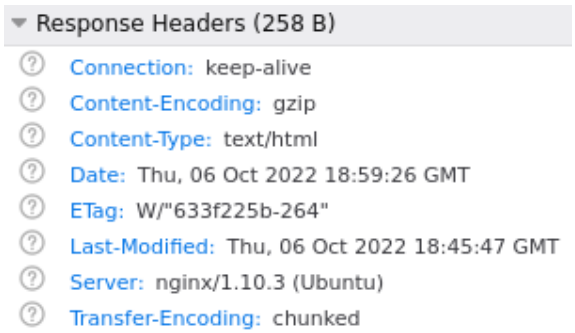
Figur 3: Output var `/var/log/snort/alert`, som viser de alarmer, der blev aktiveret.

Spm 3.4 Modtagelse af alarmer

Som det kan ses på figur 4, modtages der kun alarmer på regel 2 og 3. Regel 2 aktiveres to gange og regel 3 aktiveres kun én.

Regel 2 bliver aktiveret, fordi denne leder efter ordet "gzip" i pakke content. Når dette bliver kørt på en pakke indeholdende HTTP data, bliver kun HTTP headers tjekket med denne konfiguration. Alarmen bliver da aktiveret, da hjemmesiden i feltet "Content-Encoding" specificerer, at data, der sendes, er komprimeret med gzip. Grunden til, at reglen bliver aktiveret 2 gange, er fordi serveren har dette felt markeret i både svaret på den første GET-request, som besvares med status 200, og den anden GET-request, der besvares med en fejl 404. Figur 4 viser de headers, der var i svaret fra `GET /`-forespørgslen.

Regel 3 bliver kun aktiveret én gang, da denne leder efter ordet "Welcome". I modsætning til regel 1 leder regel 3 efter ordet i HTTP Response Body, fordi reglen er blevet bedt om at pege mod `file_data`. Dette forklarer også hvorfor regel 1 ikke blev aktiveret, da der ikke står "Welcome" i HTTP header. På grund af `metadata:service http` leder SNORT kun efter HTTP-pakker. Da passer kun én pakke til denne regel, nemlig den pakke, der indeholder websidens HTML, sendt med HTTP med status kode 200, hvor der står "Welcome" i den HTML, der vises, når siden tilgås.



Figur 4: Screenshot fra Firefox, der viser headers i modtaget pakke fra `GET /`.

Opgave 4: Firewall

Spm 4.1 IP Tables del 1

```
iptables -A OUTPUT -p tcp -j ACCEPT
```

Kommandoen betyder, at firewallen iptables vil acceptere alle udgående TCP-forbindelser. Da reglen blot kigger på al TCP-trafik, kan den anses som stateless. Der kigges altså ikke på TCP

stadier eller lignende.

```
iptables -A INPUT -p tcp -m state --state ESTABLISHED -j ACCEPT
```

Denne regel accepterer alle indkommende TCP-pakker, såfremt de er i en etableret tilstand. Denne regel er altså stateful.

Spm 4.2 IP Tables del 2

```
iptables -A OUTPUT -p tcp -j ACCEPT
```

Denne kommando er ækvivalent med den givet i tidligere spørgsmål. Den indsætter en række i bunden af firewallens outputkæde, som accepterer alle TCP-forbindelser. Denne er da også stateless.

```
iptables -A INPUT -p tcp -j ACCEPT
```

Denne kommando indsætter en række i bunden af firewallens inputkæde, som accepterer indkommende TCP-forbindelser. Denne er, af samme årsag som ovenstående, stateless.

```
iptables -A INPUT -p tcp --syn -j DROP
```

Denne kommando indsætter en række i bunden af firewallens inputkæde, som dropper alle pakker, der matcher at være TCP-pakker med SYN-flaget sat. Alle indkommende TCP-forbindelser vil da blive blokeret af firewallen. Dette holder heller ikke styr på de specifikke TCP-forbindelser og er derfor også stateless.

Spm 4.3 Implementering af IP Tables

Køres kommandoerne fra Spm 4.1 og Spm 4.2, sker der ikke noget med TCP-forbindelsen til serveren. Dette er eftersom den regel, som dropper indgående pakker med SYN-flaget sat, står nederst i firewallens tabel. Da vil reglen over, som accepterer alle TCP-forbindelser, aktivere før og pakken vil slippe igennem. Ligeledes vil alle TCP-pakker i output slippe igennem, da disse er bedt om at hoppe til `ACCEPT`.

Kører man de kommandoer, som er specificeret i øvelserne til uge 5, vil TCP-forbindelsen til at starte med ikke virke. Dette er eftersom den øverste regel i firewallens tabel for både input og output dropper alle pakker. Da kontakt til siden kun kører over TCP og ikke ICMP, vil man heller ikke få adgang til serveren ved at åbne for disse pakker i firewallen. Åbner man til gengæld for TCP på de porte, som HTTP plejer at køre på (port 80), kan man godt få adgang til siden. Dette ville dog til hverdagsbrug ikke være nok, da langt de fleste sider henviser til versionen med `https`, som plejer at køre på port 443 i stedet for 80. Da ville det også være nødvendigt at åbne for TCP-trafik på denne. Det ville også blive nødvendigt i de fleste sammenhæng at åbne port 53, som DNS-trafik kører over.