34334 - Advanced Data Networks and Cyber Security

Daniel Brasholt s214675

F22

Contents

Week 02: HTTP Effektivitet

Date: Monday 05.09.22

Lecture slides 02A

Lecture slides 02B

Lecture slides 02C

HTTP Effektivitet

HTTP/1.0 og TCP

Hver request blev igangsat af en 3-way handshake med TCP. Hver forbindelse blev derefter termineret og klient/server skulle instantiere endnu en forbindelse.

HTTP pipelining

Mest i HTTP/1.0, hvor forbindelserne dog bliver oprettet på samme tid i takt med, at browseren kan se, at flere bliver nødvendige. Forbindelse #1 bruges til html, #2 til første billede, #3 til andet billede og så videre. Bliver også lidt brugt i HTTP/1.1.

HTTP/1.1

Forbindelsen bliver ikke lukket mellem requests - i stedet bliver den samme TCP-forbindelse genbrugt mellem requests.

Forbindelsen bliver i stedet lukket, når den har været inaktiv i et givent stykke tid. Skal siden så genindlæses eller information hentes igen, må forbindelsen så bare blive genoprettet.

Head-of-line (HOL) blocking

Uden pipelining: den første request er stor, så den blokerer for resten og der kommer noget lignende congestion.

HTTP/2

Pipelining inden for enkelt TCP-forbindelse. Hver chink i svar angiver hvilket GET-request, de tilhører. Serveren kan så levere alle billeder i parallel.

Cookies

HTTP er stateless, så serveren husker ikke information om klienter. For at mindske load på serveren, gemmer den cookies på klient-PC, fx session cookies og persistent cookies.

Cookies bliver sat af serveren i GET-reply og bliver så sendt med fremtidige GET-requests af klienten.

Ved fx at sætte et billede på 1x1 pixel, vil klienten sende en “trackerserver” den cookie med et tracker ID, så trackerserveren nu ved, at samme bruger har besøgt forskellige sites (lecture slide 02A 13).

Webcaching

Browsere kan konfigureres med webcache som proxy og kan så sende alle requests gennem denne. Hvis cache-serveren ser flere ens requests, har denne nu en gemt kopi fra en tidligere forespørgsel. Det ekstra led i kæden bør ikke være et problem, fordi LAN bør være hurtigt nok til at det ikke opdages.

Mest brugt af ISP'er, så CDN'er kan have cache-servere liggende hos ISP'er. YouTube har fx en cache i Albertslund.

- Første forespørgsel fra en klient til cache:
  - Cache => Webserver: GET
  - Gemmer svar fra webserver samt indhold af 'Last-modified:' header (timestamp)

- Senere forespørgsler fra klienter:
  - Cache => Webserver: GET med 'If-modified-since:' header med timestamp
  - Webserver: Tjekker om lokal information er nyere end timestamp
    - Hvis ja: Svarer med data og opdateret 'Last-modified:' header
      - Cache returnerer modtaget data til klient
    - Hvis nej: Svarer med (kort) "304 Not modified" reply
      - Cache returnerer gemt data til klient

Figure 1: Conditional GET

HTTP Sikkerhed (TLS)

Liste af WWW-trusler på lecture slide 02B 2.

Sikkerhed bliver oftest implementeret i transportlaget - IPSec er ikke særligt udbredt og at bygge det ind i applikationslaget giver ekstra krav til alle de forskellige applikationer, der skal forsøge at implementere dette.

TLS bliver implementeret mellem applikationslaget og transportlaget - data bliver så sendt i TCP-pakke.

Historik: SSL 1.0 (ikke udgivet), 2.0, 3.0, TLS 1.0, 1.1, 1.2, 1.3. Kun 1.2 og 1.3 bør være i brug.

Skema over TLS Record på lecture slide 02B 6.

Sessioner & forbindelser

TLS Session: association mellem to parter, fx server og klient. Specificerer relevante sikkerhedsparametre (algoritmer, ciphers osv.)

TLS Forbindelse: dataflow mellem server og klient. Udregning af nøgler osv.

Man kan så genbruge parametre fra tidligere session i ny forbindelse.

Change Cipher & Alert protokoller

Change Cipher: en protokol med 1 besked: change cipher.

Alert: Benyttes til at angive fejl i udveksling af TLS-meddelelser. Giver ofte en terminering af TLS-forbindelse.

Handshake

Handshake aftaler TLS-version, algoritmer, ciphers og til sidst nøgler.

Fase 1: Først oprettes TCP-forbindelse. Inde i denne bliver der sendt **client_hello** - seneste TLS-version understøttet, tilfældig information til nøgler, forslag til algoritmer. **server_hello** angiver TLS-version, tilfældig information og valg af algoritmer.

Fase 2: Indeholder **certificate** underskrevet af CA, **server_key_exchange** (parametre til Diffie-Hellman) og måske **certificate_request** hvis server vil have klient certificeret.

Fase 3: Indeholder **certificate**, hvis serveren bad om det; **client_key_exchange**, fx klientens Diffie-Hellman-parametre; og **certificate_verify**, som er hash udregnet over alle tidligere beskeder. Denne underskrives med private key.

Heartbeat: Sener data frem og tilbage en gang imellem for at tjekke, at forbindelsen stadig er aktiv og i live.

TLS 1.3

Mindre sikre algoritmer og options udeladt. Hurtigere "1-RTT" handshake kan i nogle tilfælde laves, hvor klient sender key-data i forbindelse med **client_hello**.

Nogle gange mulighed for "0-RTT" handshake, hvor klient og server har gemt tidligere nøgle, som genbruges. Dog mindre sikkert, hvis nøgle er kompromitteret.

HTTPS - HTTP over TLS

Efter 3-way handshake og TLS handshake, kan man så begynde at sende HTTP-trafik over forbindelsen.

Wireshark-øvelser - del 1

1. HTTP

- Spm 1: Version HTTP/1.1
- Spm 2: Bør returnere 200
- Spm 3: 04. September kl 13:58:01
- Spm 4: 222 bytes

1.1

- Spm 5: Der er tilføjet en If-Modified-Since header og en If-None-Match header
- Spm 6: Serveren returnerer først content og så ikke
- Spm 7: Først 200 med content, så 304 uden

1.2

- Spm 8: Der blev afsendt 2 GET-requests: en til siden og en til Favicon.
- Spm 9: Status er 200 i begge svar
- Spm 10: Til tekst-stykket er det nødvendigt med 3 TCP segmenter; 2 med svar og 1 med statuskode. Favicon sendes i enkelt pakke

2. TLS

2.2

- Spm 11: Content Type: handshake
- Spm 12: Den indeholder ikke en nonce
- Spm 13: Første Cipher Suite: TLS_AES_128_GCM_SHA256 og Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
- Spm 14: Browser understøtter 1.3 og 1.2.

2.3

- Spm 15: Cipher Suite: TLS_AES_256_GCM_SHA384
- Spm 16: Ingen nonce
- Spm 17: Ja, meningen er at genbruge TLS-sessionen til flere forskellige forbindelser eller forespørgsler
- Spm 18: Man kan ikke se et server-certifikat, da dette bliver sendt i et krypteret felt efter key exchange.

2.4

- Spm 19: Nej, men den er efter sigende ikke med i en GCM-krypteret besked
- Spm 20: Nej

DNS Sikkerhed

Mulige løsninger: DNS over TLS, DNS over DTLS, DNS over HTTPS (‘DoH’). Disse er dog ikke bredt anvendt, fx fordi der er ekstra overhead.

DNS Cache Poisoning: Se slides for proces. Handler om at give en forkert IP-adresse til autoritativ navneserver, så man spørger en forkert ved DNS-opslag.

DNSSEC

Sikrer korrekthed af records, ikke mod aflytning eller DoS. Hver record er tilknyttet en digital signatur (hash krypteret med private key). Denne signatur er genereret af autoritativ entitet for relevant DNS-zone.

Hver zone har to sæt {private, public}-keys: zone-signing keys og key-signing keys.

Privat ZSK-nøgle benyttes til digital signatur af alle traditionelle DNS records. Signaturen kan verificeres ved offentlig ZSK-nøgle.

Ved opslag: ønsket record og tilhørende signatur returneres. Kan verificeres ved at resolver også sender query for DNSKEY record for offentlig ZSK-nøgle, så resolver kan verificere signatur på oprindelig record. Offentlig ZSK er så signeret af privat KSK-nøgle, så den kan verificeres med offentlig KSK-nøgle.

Hash af offentlig KSK-nøgle er gemt hos parent zone, kaldet *delegation signer*. Hash af offentlig KSK-nøgle er signeret med privat ZSK-nøgle for *parent zone*. Tillid til offentlig KSK-nøgle = tillid til *parent zone*.

DNSKEY for *root zone* antages kendt og *trusted*.

De øverste nøgler erstattes ca. 4 gange om året.

Wireshark-øvelser - del 2

Week 03: Wireless Sikkerhed

Date: Monday 12.09.22

Lecture slides

802.11 wireless sikkerhed

802.11 ramme

- Addr1: MAC på modtager på trådløs, fx AP
- Addr2: MAC på afsender af ramme
- Addr3: MAC på router interface / 802.3-modtager

Associering med AP

- Scanner kanaler og lytter efter beacon rammer
- Vælger AP
- Typisk: authentication
- Associering og aftale parameter
- Sikkerheds-authentication
- IP via DHCP

Passiv scanning:

- Scanning efter beacon frames
- Association Request til ønsket AP (type 0)
- Association Response til enhed

Aktiv scanning:

- Probe Request fra enhed
- Probe Response fra AP

- Association Request/Response

WEP

- Afsender beregner Integrity Check Value
- 104 bit shared key
- 24 bit initialization vector
- 128-bit key i PRNG
- Datarammer krypteres

Autentifikation og kryptering

AS = Authentication Server

1) identifikation af hvad der understøttes
2) Gensidig autentifikation og udledning af "shared symmetric key"
    - AS og klient har kendt hemmelighed (password)
    - Nonce bruges til at forebygge replay attack og signatur (integritet)
3) Shared symmetric key (fx AES) til kryptering
    - Samme nøgle udledt af klient og AS

EAP

Extensible Authentication Protocol (EAP): definerer end-to-end request/response protokol mellem klient og AS (EAPoL (EAP over LAN) i Wireshark)

4-way handshake

- To formål:
    - Gensidig autentifikation
    - Forhandle nye PTK (Pairwise Temporal Key)
- Forekommer sikker:
    - Ingen angreb i mange år udover password-gætteri
    - Matematisk bevist at PTK er hemmelig
    - Selve krypteringsprotokollen er sikker

Msg1(r,ANonce), Msg2(r,SNonce) - begge parter udleder PTK

Msg3(r+1, GTK) fra AP (krypteret)

Msg4(r+1) fra enhed (krypteret)

Enhed har nu PTK & GTK, AP har PTK

Key Reinstallation Attack

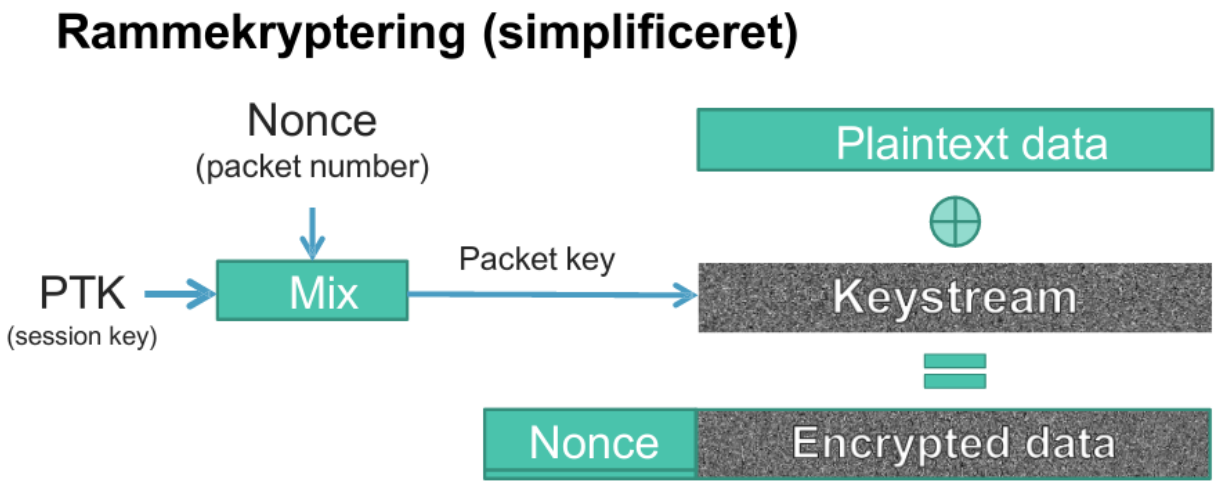

Figure 2: Rammekryptering (simplificeret)