Tampereen yliopisto

Mika Roisko

HTTP STRICT TRANSPORT SECURITY

Käytännön vaikutus selainpohjaisen automaation tietoturvaan

Kandidaatintyö
Tekniikan ja luonnontieteiden tiedekunta
Huhtikuu 2022

TIIVISTELMÄ

Mika Roisko: HTTP Strict Transport Security
Kandidaatintyö
Tampereen yliopisto
Teknisten tieteiden kandidaatin tutkinto-ohjelma
Huhtikuu 2022

Http Strict Transport Security (HSTS) -käytäntö on internetin tietoturvaa parantava Hypertext Transfer Protocol (HTTP) -tiedonsiirtoprotokollan viestien ylätunniste, josta tehtiin RFC-standardi vuonna 2012. Tämän tutkimuksen tarkoituksena on vastata kysymykseen, miten HSTS vaikuttaa selainpohjaisen automaatiosovelluksen tietoturvaan. Työ on toteutettu kirjallisuusselvityksenä, jonka lähteinä on käytetty ensisijaisesti internet-standardeja, konferenssijulkaisuja ja artikkeleita.

Tutkimuksessa perehdytään TCP/IP-protokollaperheen mukaisen internetyhteyden muodostamiseen uudelleenohjauksen avulla ja sen jättämään tietoturvariskiin, joka ratkaistaan HSTS-käytännöllä. Uudelleenohjaus on yleinen tapa muodostaa suojattu verkkoyhteys, mutta osa liikenteestä tapahtuu suojaamattoman verkon ylitse. Suojaamaton yhteys jättää riskin mies välissä -hyökkäykselle, jossa kolmas osapuoli tunkeutuu verkossa keskustelevien osapuolten väliin. Tutkimuksessa käsitelty HSTS-käytäntö on toteutettu minimoimaan tämän hyökkäyksen riski pakottamalla osapuolten välinen yhteys suojatuksi. Käytännön positiivinen vaikutus perustuu siihen, että suojaamatonta liikennettä ei sallita, jolloin osapuolten välistä yhteyttä ei voida riisua suojaamattomaksi hyökkääjän toimesta. HSTS pienentää myös riskiä evästeiden kaappauksiin, sillä osapuolten välillä siirretty data välitetään symmetrisen salauksen avulla suojattuna.

Automaatiossa HSTS-käytännön käyttökohteena on loppukäyttäjän ja palvelimen välisen tietoturvan parantaminen. Tutkimuskysymyksessä esitelty automaatioverkko rajattiin esineiden internetiin (engl. Internet of Things, IoT). Esineiden internetissä laitteita on kytketty internetiin, jonka kautta niitä voidaan lukea tai ohjata. IoT-verkon yli siirretty datamäärä on suuri, ja data sisältää paljon arkaluontoista tietoa. Siksi tiedon luottamuksellisuus, eheys ja käytettävyys ovat merkittävässä roolissa. Tutkimuksesta ilmenee, että käytäntö on hyvä lisä kattavaan tietoturvaan, mutta automaatiossa sen käyttökohteet osoittautuvat pieniksi. Automaatiossa haasteita aiheuttavat itseallekirjoitetut varmenteet ja verkkojen käyttämät protokollat, jotka eroavat TCP/IP-perheestä. Kaikkea dataa ei voida eikä haluta salata, sillä esimerkiksi laitteiden vikatilanteissa salattujen viestien purkaminen on työlästä.

Avainsanat: HSTS, HTTPS, Man-in-The-Middle, tietoturva, verkkohyökkäys

SISÄLLYSLUETTELO

1. Johdanto 1
2. Suojatun internetyhteyden muodostaminen ja sen riskit 3
2.1 Internetyhteyden muodostaminen protokollien avulla 4
2.1.1 OSI-mallin alempi kerros internetissä 4
2.1.2 OSI-mallin ylempi kerros internetissä 6
2.2 Palvelimen uudelleenohjaus 10
2.3 Verkkohyökkäykset 11
2.4 Mies välissä -hyökkäys 12
3. Strict Transport Security 14
4. HSTS:n merkitys esineiden internetissä. 18
4.1 Automaatioverkon turvallisuusvaatimukset 18
4.2 Käyttökohteet automaatiossa 20
4.3 HSTS-käytännön vaikutukset automaation tietoturvaan 21
4.4 Haasteet automaatioverkoissa 22
5. Yhteenveto 24
Lähteet 26

LYHENTEET JA MERKINNÄT

CA	Certificate authority, luotettava sertifikaatin myöntäjä
DNS	Domain Name System
Domain	Palvelimen verkkotunnus
Header	HTTP:n otsikkokenttä/ylätunniste
HSTS	HTTP Strict Transport Security
HTTP	Hyper Text Transport Protocol
HTTPS	Hyper Text Transport Protocol Secure
IoT	Internet of Things, esineiden internet
IP	Internet Protocol
MITM	Man-In-The-Middle (mies välissä) -verkkohyökkäys
OSI-malli	Open Systems Interconnection, tiedonsiirtoprotokollien kuvaus pinona
PKI	Public Key Infrastructure, julkisen avaimen infrastruktuuri
SSL	Secure Socket Layer
Subdomain	Palvelimen aliverkkotunnus
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UA	User Agent, käyttäjäagentti
UDP	User Datagram Protocol
URL	Uniform Resource Locator

1. JOHDANTO

Esineiden internet (engl. Internet of Things, IoT) on nopeasti yleistyvä tapa tehdä esimerkiksi rakennusten, tehtaiden tai asuntojen ominaisuuksista etäluettavia tai -ohjattavia. Smart home (älykoti) -ratkaisut tuovat helpotusta arkeen, kun kodin lukitus, valaistus tai laitteet ovat yhdistettynä internetiin, jonka kautta laitteita voidaan ohjata käyttöliittymän avulla mistä päin maailmaa tahansa. Tehtaissa esimerkiksi prosessien tilaa tai anturien dataa voidaan lukea tai ohjata etänä, mikä pienentää tarvetta manuaaliselle työlle. Rakennuksissa lukitusta, kulutusmittareita tai vaikkapa kulunvalvontaa voidaan siirtää verkkoon. Tätä konseptia kutsutaan nimellä esineiden internet, jossa laitteiden tiedot jaetaan palveluntarjoajalle, joka tarjoaa käyttöliittymän niiden ohjaamiseen. Esineiden internet tuo helppoutta elämään ja työhön, mutta riski joutua aktiivisen verkkohyökkäyksen kohteeksi kasvaa, kun internetin yli siirretty data sisältää yhä enemmän arkaluontoista tietoa. (Varga et al. 2017) Eräs suuri riski syntyy mies välissä (engl. Man-in-The-Middle, MiTM) -hyökkäyksestä, jossa kolmas ei-toivottu osapuoli tunkeutuu osapuolten väliin lukemaan ja manipuloimaan internetin yli siirrettyä dataa.

Esineiden internet vaatii toimiakseen internetyhteyden, joka perustuu yhteistietokäytänteisiin eli protokolliin. Internet Protocol (IP) ja Transmission Control Protocol (TCP) luovat yhdessä internetin tärkeimmän protokollapinon TCP/IP, jossa IP huolehtii datapakettien toimittamisesta osapuolten välillä ja TCP huolehtii pakettien saapumisesta perille. Hypertext Transfer Protocol (HTTP) ja Transport Layer Security (TLS) yhdessä muodostavat suojattuna protokollana tunnistetun Hypertext Transfer Protocol Securen (HTTPS), jossa HTTP:n yli siirretty liikenne on salattu TLS-protokollan mukaisesti vaikeuttaen MiTM-hyökkäysten toteutusta. Näiden protokollien avulla muodostetaan suojattu internetyhteys. HTTPS itsessään ei kuitenkaan estä mies välissä -hyökkäyksen mahdollisuutta.

Http Strict Transport Security (HSTS) on HTTPS-viestien mukana lähetettävä ylätunniste (engl. header), jonka tarkoitus on pakottaa osapuolet keskustelemaan HTTPS-protokollan mukaisesti suojaamattoman HTTP:n sijaan. TLS:n mukainen salaus pienentää riskiä joutua etenkin mies välissä -hyökkäyksen kohteeksi ja pakotettuna HSTS:n mukaisesti, osapuolet sopivat pitäytyvänsä suojatussa yhteydessä niin kauan, kun yhteys on auki. Pakotettu HTTPS pienentää hyökkäysten riskiä entisestään. HSTS-käytännöstä tehtiin standardi Internet Engineering Task Forcen (IETF) toimesta vuonna 2012, ja sen ensisijainen tarkoitus on ratkaista riskit, joita HTTPS ei ratkaise. (Hodges et al. 2012)

Kandidaatintyön tutkimuskysymyksenä on, *miten HSTS-käytäntö vaikuttaa selainpohjaisen automaatiosovelluksen tietoturvaan.* Tutkimus on toteutettu kirjallisuusselvityksenä, jonka lähteinä on käytetty ensisijaisesti IETF:n määrittämiä standardeja, konferenssijulkaisuja sekä erinäisiä teoksia, joissa on tutkittu sekä verkkohyökkäyksiä että niiden torjumista eri keinoin. Tämän tutkimuksen alussa tutustutaan yleisellä tasolla internetin yli keskustelevien osapuolten käyttämiin protokolliin, kuten TCP/IP-protokollaperheeseen sekä HTTP- ja HTTPS-protokolliin sekä niiden eroihin. Tärkeässä osassa on suojatun yhteyden muodostaminen TLS:ää käyttäen. Tämän jälkeen esitellään verkkohyökkäyksiä, jotka kohdistuvat edellä mainittuihin protokolliin.

Kolmannessa luvussa tarkastellaan HSTS-käytäntöä, jonka ensisijainen käyttötarkoitus on estää mies välissä -hyökkäyksiä. Luvussa esitellään HSTS:n standardia, sen käytännön periaatteita, käyttöönottoa, hyötyjä ja ratkaisu toisessa luvussa esitettyyn tietoturvariskiin. HSTS-käytäntöä voidaan käyttää missä tahansa verkossa, joka kykenee keskustelemaan HTTPS-protokollan ylitse. Tällainen verkko voi olla esimerkiksi esineiden internetin käyttämä verkko, joka aiheuttaa omat turvallisuusvaatimuksensa. Neljännessä luvussa syvennytään tarkemmin automaation kontekstiin ja pohditaan, mitä vaatimuksia automaatioverkolla on ja missä automaatiossa voidaan soveltaa HSTS-käytäntöä. Kaikkea liikennettä ei voida eikä haluta suojata, mikä aiheuttaa ongelmia HSTS:n kanssa, joista on kerrottu tarkemmin neljännessä luvussa. Viimeisessä luvussa esitellään yhteenvetona tutkimuksen tulokset sekä pohditaan työn onnistumista.

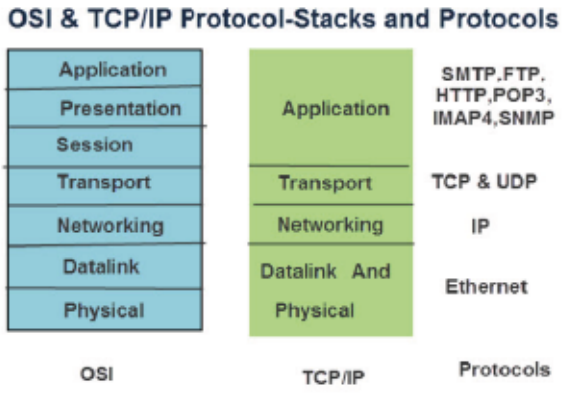
2. SUOJATUN INTERNETYHTEYDEN MUODOSTAMINEN JA SEN RISKIT

Internet ja automaatio mahdollistavat monen arkipäiväisen toimen siirtämisen digitaaliseksi. Erilaiset esineiden internet (engl. Internet of Things, IoT) -ratkaisut, kuten älykoti (engl. smart home) tai tehdasautomaatio (engl. Industrial IoT), tuovat helpotusta arkeen ja työhön, kun esimerkiksi lukitusta, kulunvalvontaa, valaistusta tai laitteita voidaan ohjata tai lukea mistä päin maailmaa tahansa internetin välityksellä. Kaikki tämä on rakennettu internetin varaan. Internet on toistensa päällä toimivia yhteistietokäytänteitä eli protokollia (Nielsen et al. 1999). Protokollien yhdistelmistä voidaan toteuttaa monenlaisia verkkoja, mutta tutkimuksessa keskitytään ensisijaisesti internetiin ja tutkimuksen kannalta oleellisiin protokolliin.

Internetissä keskustelevat osapuolet voidaan jakaa osiin, käyttäjän puolesta toimivaan käyttäjäagenttiin (user agent, UA) ja palvelimeen. Käyttäjäagentti voi olla esimerkiksi selain, editori tai muu loppukäyttäjän työkalu, kuten puhelinsovellus. Palvelimet ovat palveluntarjoajan tietokoneita tai virtuaalikoneita, jotka suorittavat palvelinohjelmistoa ja vastaavat käyttäjäagentin pyyntöihin. Asiakkaan ja palvelimen väliseen internetin ylitse tapahtuvaan keskusteluun sisältyy paljon erilaisia protokollia. (Nielsen et al. 1999) Tässä luvussa tarkastellaan yleisesti tutkimuksen kannalta tärkeitä internetiin liittyviä protokollia, niiden tarkoitusta ja internetyhteyteen kohdistuvia hyökkäyksiä, joita tutkimuksessa käsitelty Http Strict Transport Security (HSTS) pyrkii ratkaisemaan.

2.1 Internetyhteyden muodostaminen protokollien avulla

Protokollia voidaan jakaa perheisiin tai kerroksiin niiden käyttötarkoituksen tai osallistumisjärjestyksen mukaan. Tällä tavoin protokollia voidaan kuvata pinona, jossa alempien protokollien päälle pinotaan ylempien kerroksien protokollia. Pinoamisen edellytyksenä on alemman kerroksen protokollan onnistunut suoritus. Kuvan 2.1 vasemmassa laidassa on esitettynä OSI-mallin mukaiset kerrokset.



Kuva 2.1. *OSI-mallin sekä TCP/IP-protokollaperheen mukaiset kerrokset (Cope 2019).*

OSI-mallissa on seitsemän kerrosta, joista jokainen kerros käyttää vähintään yhtä alemman kerroksen protokollaa ja on yhteydessä ylempiin kerroksiin, pois lukien tietysti alin ja ylin kerros eli fyysinen ja sovelluskerros. Alemman kerroksen protokollat ovat alkeellisempia, ja ylemmän kerroksen suorittaminen vaatii alemman kerroksen protokollan onnistuneen suorituksen. (Ala-Mutka et al. 2002) Tutkimuksen kannalta OSI-mallista erotetaan TCP/IP-protokollaperheen mukaiset protokollat, joista käsitellään ensisijaisesti Ethernet, Internet Protocol (IP), Transmission Control Protocol (TCP), Hypertext Transfer Protocol (HTTP) ja Transport Layer Security (TLS). Nämä protokollat muodostavat kuvan 2.1 oikean laidan mukaisen TCP/IP-protokollaperheen, joka on edellytys internetyhteydelle ja HSTS:n toiminnalle.

2.1.1 OSI-mallin alempi kerros internetissä

Kaikki internetiin kytketyt laitteet pohjautuvat OSI-mallin alemman kerroksen protokollien onnistuneeseen suoritukseen. IoT:n kannalta laitteet vaativat yhteyden lähiverkkoon, ja kaikki laitteiden tuottama data siirretään näiden protokollien avulla. Tässä alaluvussa tarkastellaan internetyhteyden kivijalka eli alemman kerroksen protokollat.

Kuvassa 2.1 esitetyn OSI-mallin alemman eli neljän ensimmäisen kerroksen protokolliin kuuluvat muun muassa Ethernet, IP ja TCP, jotka ovat internetin tärkeimpiä protokollia. Ethernet on fyysisen kerroksen protokolla, joka on lähiverkkoon tarkoitettu pakettipohjainen ratkaisu ja joka toteuttaa sekä fyysisen että siirtoyhteyskerroksen. Ethernet-yhteys

saadaan muodostettua kahden tai useamman laitteen välille käyttäen fyysistä Ethernet-kaapelia (fyysinen kerros), jolloin dataa voidaan lähettää datapaketteina osapuolelta toiselle (siirtokerros). (Postel ja Reynolds 1988) Lähetettävä tiedosto jaetaan pienempiin osiin eli paketteihin, jotka lähetetään tietyssä järjestyksessä vastaanottajalle, joista vastaanottaja kokoaa tiedoston uudestaan luettavaan muotoon Ethernet-protokollan mukaisesti. Internetyhteyden edellytys on kytkettävän tietokoneen, kytkimen tai reitittimen liittäminen lähiverkkoon käyttäen Ethernetiä.

Kun laite on kytkettynä Ethernet-protokollan mukaisesti lähiverkkoon, IP huolehtii pakettien lähetyksestä ja toimituksesta oikeaan osoitteeseen. IP on verkkokerroksen protokolla, mutta se ei takaa datapakettien saapumista, vaan datapaketit voivat kadota, viivästyä, saapua väärässä järjestyksessä tai niitä voi tulla useampi yhden sijaan. Laitteet, jotka ovat yhteydessä internetiin, yksilöidään IP-osoitteiden avulla, jolloin IP-protokolla huolehtii, että osapuolet lähettävät paketit oikealle vastaanottajalle. (*Internet Protocol* 1981)

TCP on kuljetuskerroksen protokolla, jonka avulla paketteja voidaan lähettää luotettavasti. Protokollan tarkoitus on kertoa paketin kohtalosta, johon IP ei ota kantaa. Jotta TCP toimii, se tarvitsee sekä lähettäjän että vastaanottajan IP-osoitteen. Protokollassa on kolme vaihetta: yhteyden muodostaminen, tiedonsiirto ja yhteyden katkaisu. Ensimmäinen vaihe, yhteyden muodostaminen eli TCP-kättely etenee seuraavasti:

1. Yhteyden aloittava osapuoli lähettää SYN-paketin (synchronization), joka sisältää satunnaisen sekvenssinumeron A.
2. Vastaanottaja vastaa SYN/ACK-paketilla (acknowledgement), joka sisältää yhden suuremman sekvenssinumeron A+1 kuittauksena paketin saapumisesta. SYN/ACK sisältää myös toisen satunnaisen sekvenssinumeron B.
3. Viimeisessä yhteyden muodostamisen vaiheessa yhteyden aloittaja kuittaa vastaanottaneensa SYN/ACK-paketin lähettämällä vastauksena ACK-paketin, joka sisältää sekvenssinumeron B+1.

Osapuolet tunnistavat toisensa, sillä vastauksessa on aina lähetettyä sekvenssinumeroa seuraava luku ja osapuolet osaavat odottaa vastausta oman pyyntönsä sekvenssinumeron perusteella. Näin osapuolet tietävät kommunikoivansa toistensa kanssa. TCP-datansiirto perustuu näihin SYN/ACK-viesteihin. Mikäli protokolla ei saa toiselta osapuolelta ACK-pakettia, joka kuittaa paketin vastaanotetuksi, datapaketti lähetetään uudelleen. Sekvenssinumeroiden perusteella yksilöidyt paketit on helppo koota takaisin alkuperäiseksi viestiksi. (*Transmission Control Protocol* 1981) Kun yhteyttä ei enää käytetä, osapuolet katkaisevat yhteyden.

TCP-protokolla täydentää alemman kerroksen IP-protokollan, jolloin yhdistelmästä saadaan TCP/IP. Tässä protokollassa IP huolehtii osapuolille osoitteet ja reitittää ne, minkä jälkeen TCP-protokolla huolehtii datapakettien siirrosta ja siirron onnistumisesta. (*Internet Protocol* 1981; *Transmission Control Protocol* 1981) TCP/IP luokitellaan protokollaperheeksi, joka on esitettynä kuvassa 2.1. TCP/IP-protokollaperhe on edellytys HSTS-käytännölle.

TCP/IP-protokollaperheeseen kuuluu nimen lisäksi muitakin alemman kerroksen protokollia, kuten esimerkiksi TCP:n kaltainen kuljetuskerroksen protokolla UDP (User Datagram Protocol) ja DNS (Domain Name System). DNS-protokolla kääntää käyttäjän syöttämät www-alkuiset verkkotunnukset IP-osoitteiksi, joiden avulla paketit reititetään oikeille vastaanottajilla. UDP on puolestaan yhteydetön tietoliikenneprotokolla, jota käytetään TCP:n sijasta silloin, kun pakettien saapumista ei tarvitse varmistaa ja verkon yli siirrettävä tietomäärä on suuri. (*User Datagram Protocol* 1980) UDP ei ole TCP:n tavoin luotettava protokolla, sillä se ei ota kantaa datapakettien kohtalosta.

2.1.2 OSI-mallin ylempi kerros internetissä

Ylemmän kerroksen protokollat jaetaan OSI-mallin mukaan istuntokerrokseen, esitystapakerrokseen ja sovelluskerrokseen. Kuvan 2.1 mukaisesti TCP/IP-perhe ei sisällä istunto- tai esitystapakerrosta, eivätkä ne sisällä tutkimuksen kannalta olennaisia protokollia, joten jätetään ne käsittelemättä. Sovelluskerros puolestaan on OSI-mallin ylin kerros, jota sovellukset käyttävät viestintään internetin yli. Sovelluskerrokseen sisältyy esimerkiksi protokollat TLS/SSL, HTTP ja HTTPS. HSTS:n kannalta näiden protokollien toiminnan ymmärtäminen on tärkeää, joten tutustutaan niihin tarkemmin tässä alaluvussa.

TLS on salausprotokolla, jota käytetään verkkoliikenteen salaamiseen IP-verkkojen yli, ja se perustuu varmenteisiin eli sertifikaatteihin (Dierks ja Rescorla 2006). Silloin kun sähköisen varmenteen myöntäjä on jokin yleisesti luotetuksi tunnustettu julkinen taho (engl. Certificate authority, CA), kuten pankki tai valtio, on kyse julkisen avaimen infrastruktuurista (public key infrastructure, PKI). Tämä infrastruktuuri perustuu varmenteiden lisäksi epäsymmetriseen salaukseen, jossa internetin yli siirrettyjen pakettien tiedot salataan kahden avaimen avulla: toisella avaimella tieto salataan ja toisella puretaan. Julkiset avaimet luodaan esimerkiksi RSA (Rivest–Shamir–Adleman), DSA (Digital Signature Algorithm) tai Diffie-Hellman -algoritmeilla. (Rescorla 2018)

Internetissä käytetyt sertifikaatit ovat yleensä tyypiltään X.509, joka on julkisen avaimen infrastruktuurin standardisoitu sertifikaatti (Boeyen et al. 2008). Se pitää sisällään tiedot esimerkiksi varmenteen versiosta, sarjanumerosta, allekirjoitusalgoritmin tyypistä, voimassaoloajasta ja käytetyn avaimen tyypistä sekä luontiin käytetystä algoritmista. Sertifikaatti määrittää yksityiskohdat palvelimen ja käyttäjäagentin välisen liikenteen salauksesta ja tarjoaa salaukseen tarvittavan julkisen avaimen molemmille osapuolille. (Rescor-

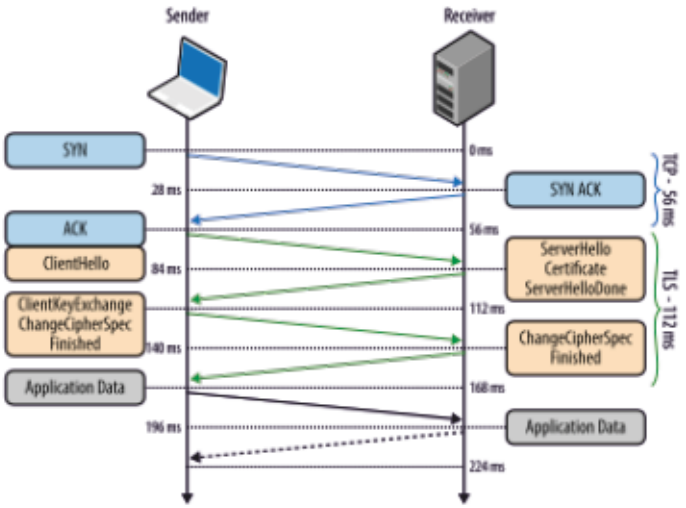
la 2018) Mikäli verkkosivun yhteys on suojattu TLS-protokollalla, palvelimelle myönnettyä sertifikaattia voi tutkia klikkaamalla selaimen URL:in (Uniform Recource Locator) vieressä olevaa lukkoa, jonka jälkeen aukeaa mahdollisuus tutkia sertifikaattia.

TLS sisältää TCP:n tavoin kättelyvaiheen (engl. TLS handshake), jossa osapuolet tunnistautuvat toisilleen ja vaihtavat salaisuuksia keskenään. Kättely on erilainen riippuen avaintenvaihtoalgoritmista. Yleisin on RSA-avaintenvaihtoalgoritmi, joka etenee seuraavasti (Rescorla 2018):

1. TLS-kättely alkaa selaimen lähettämästä 'Client hello' -viestistä, joka sisältää kryptografisia tietoja, kuten tuetun TLS/SSL-protokollan version, satunnaisen bittijonon (client random), jota käytetään laskennassa ja tuetut *Cipher suite*:t. Esimerkki uniikista *Cipher suitesta* $DHE\_RSA\_AES256\_SHA256$, josta ilmenee
    - Avaintenvaihtoalgoritmi DHE,
    - Autentikointialgoritmi RSA,
    - Datan salaukseen käytetty algoritmi AES256 ja
    - *Message Authentication Code* (MAC) -algoritmi SHA256 (Bisson 2019).
2. Palvelin vastaa 'Client hello' -viestiin 'Server hello' -viestillä, joka sisältää palvelimen sertifikaatin, palvelimen valitseman *Cipher suite*:n ja palvelimen luoman satunnaisen bittijonon (server random).
3. Asiakasselain todentaa palvelimen palauttaman sertifikaatin, joka varmistaa palvelimen aitouden.
4. Asiakas lähettää toisen satunnaisgeneroidun merkkijonon (premaster secret). Merkkijono on salattu sertifikaatista luetun julkisen avaimen avulla, ja salaus voidaan purkaa vain palvelimen yksityisen avaimen avulla.
5. Palvelin purkaa viestin, jolloin varmistetaan yksityisen avaimen toimivuus.
6. Asiakas ja palvelin luovat istuntoavaimet *(session key)* käyttäen *client- ja server random* sekä *premaster secret* -merkkijonoja. Molempien luotujen avainten tulisi nyt olla samat.
7. Asiakas lähettää 'finished'-viestin palvelimelle, joka sisältää salatun istuntoavaimen.
8. Palvelin lähettää 'finished'-viestin asiakkaalle, joka sisältää salatun istuntoavaimen.
9. Tästä eteenpäin on saavutettu symmetrinen salaus; kättely on valmis ja yhteys jatkuu käyttäen luotuja istuntoavaimia.

Kaikki TLS:n käytättämät algoritmit hyödyntävät epäsymmetristä salausta avainten muodostamiseen, minkä takia avaimia on lähes mahdotonta päätellä toisistaan. TLS-protokollan tarjoamaan suojaukseen riittää siis, että vain toinen avain pysyy osapuolten välisenä sa-

laisuutena. Kuvassa 2.2 on esitetty osapuolten välisen internetyhteyden muodostus vaiheittain sekä vaiheisiin kulunut aika.



***Kuva 2.2.** TCP- ja TLS-kättelyt yhteydenmuodostuksessa (Jackson 2017).*

HTTP on hypertekstin siirtoon tarkoitettu tiedonsiirtoprotokolla, ja sitä käytetään UA:n ja palvelimen välisessä tiedonsiirrossa. Protokollan suoritus alkaa siten, että UA avaa TCP-yhteyden selaimen ja palvelimen välille edellä kuvaillulla tavalla. Tämän jälkeen käyttäjäagentti lähettää http-pyynnön palvelimelle, joka vastaa pyyntöön lähettämällä vastauksen protokollan mukaisesti. Alla olevassa esimerkissä on selaimen tekemä pyyntö Netflixin palvelimelle.

```
:authority: www.netflix.com
:method: GET
:path: /WiHome
:scheme: https
accept-encoding: gzip, deflate, br
accept-language: fi-FI,fi;q=0.9,en-US;q=0.8,en;q=0.7
dnt: 1
sec-fetch-dest: document
sec-fetch-mode: navigate
sec-fetch-site: none
sec-fetch-user: ?1
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/86.0.4240.198 Safari/537.36
```

Pyynnössä esiintyvät otsikot (myös ylätunniste, engl. header) ovat HTTP-viesteissä käytettyjä tunnisteita, jotka osapuolet lukevat pyynnöistä aina. (Nielsen et al. 1999) Esimer-

kiksi selaimen pyynnössä esiintyvä *user-agent*-ylätunniste on palvelimelle välitetty tieto käyttäjän käyttämästä selaimesta ja sen versiosta. Tilan säästämiseksi pyynnöstä on poistettu muutama ylätunniste, kuten evästeet (engl. cookies).

Palvelin lukee pyynnön ylätunnisteet ja vastaa pyyntöön palvelimelle määritetyllä tavalla. Vastaukset pitävät sisällään HTTP-vastauskoodin, joka on ilmoitus käyttäjäagentille pyynnön tilasta. Alla oleva esimerkki on palvelimen vastaus aiemmin tehtyyn pyyntöön.

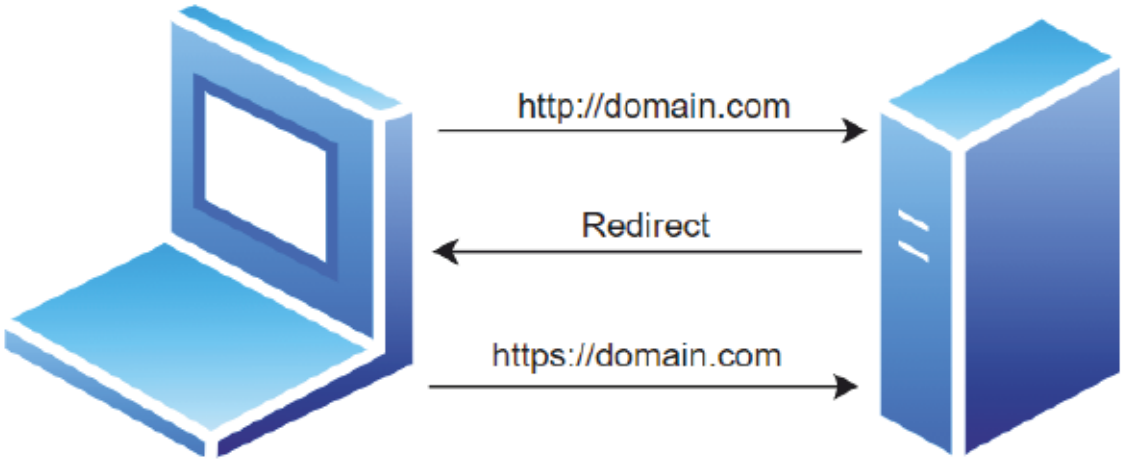
```
cache-control: no-cache, no-store
date: Sun, 22 Nov 2020 14:57:56 GMT
edge-control: no-cache, no-store
location: http://www.netflix.com/browse
status: 301
strict-transport-security: max-age=31536000
via: 2 i-0dea277fcd8881f1f (eu-west-1)
x-content-type-options: nosniff
x-frame-options: DENY
x-netflix.nfstatus: 1_1
x-netflix.proxy.execution-time: 34
x-originating-url: http://www.netflix.com/WiHome
x-xss-protection: 1; mode=block;
    report=https://www.netflix.com/ichnaea/log/freeform/xssreport
```

Yllä olevassa palvelimen vastauksessa on http:n mukainen vastauskoodi (status) *301*, johon palataan tarkemmin seuraavassa alaluvussa 2.2. Myös palvelimen vastaus pitää sisällään ylätunnisteita, jotka käyttäjän puolesta toimiva käyttäjäagentti lukee aina. Vastauksesta voidaan huomata myös tutkimuksen ydin, *strict-transport-security*-ylätunniste, joka aktivoi HSTS-käytännön palvelimen ja selaimen välille. Tämä tarkastellaan luvussa 3. Ensin tarkastellaan riski, jonka vastauskoodin 3xx mukainen uudelleenohjaus jättää jälkeensä.

2.2 Palvelimen uudelleenohjaus

Mikäli HTTP on yhdistetty TLS-protokollaan, näiden yhdistelmästä muodostuu suojattu tiedonsiirtoprotokolla HTTPS, jossa TLS on pinottu HTTP:n päälle. Tämä protokolla tunnistaa TLS:n avulla osapuolet julkisen avaimen infrastruktuurin avulla, salaa tiedot SHA (Secure Hash Algorithm) tiivistefunktion avulla ja siirtää tiedot HTTP-protokollaa käyttäen. Näin verkossa keskustelevien osapuolten välille saadaan muodostettua suojatuksi tunnistettu HTTPS:n mukainen internetyhteys. (Rescorla 2000)

HTTPS-protokollan mukainen yhteys voidaan avata kirjoittamalla selaimen kenttään https-alkuinen verkkotunnus (engl. domain). Mikäli verkkosivu pyydetään http-alkuisena tai kokonaan ilman etuliitettä, esimerkiksi *example.com*, on oletuksena suojaamaton HTTP-yhteys, sillä Lavrenovs ja Melón 2018 tekemän tutkimuksen mukaan suuri osa verkkoliikenteestä käyttää suojaamatonta yhteyttä tiedonsiirrossa. Palvelin voi ohjata selaimen käyttämään suojattua yhteyttä vastaamalla ensimmäiseen pyyntöön uudellenohjausta tarkoittavalla HTTP-koodilla 3xx. Uudelleenohjauksen mukana palvelin palauttaa https-alkuisen verkkotunnuksen, johon selaimen tulee ottaa yhteys ennen onnistunutta HTTPS-yhteydenmuodostusta. Kuvassa 2.3 on esitettynä suojatun yhteyden muodostaminen uudelleenohjausta käyttämällä.



***Kuva 2.3.** Selaimen ja palvelimen välinen uudelleenohjaus suojattuun internetyhteyteen.*

Uudelleenohjaus tarjoaa osapuolille suojatun HTTPS-protokollan mukaisen yhteyden, mutta samalla mahdollistaa verkkohyökkäyksille, sillä käyttäjäagentin ensimmäinen pyyntö ja palvelimen vastaus on suoritettu suojaamattoman HTTP-yhteyden yli. Tämä yhteydenmuodostuksen aikainen suojaamaton vaihe on aikaikkuna, jonka aikana suojattuun verkkoon on mahdollista toteuttaa verkkohyökkäys.

2.3 Verkkohyökkäykset

Palveluiden ja laitteiden siirtyessä internetiin yhä enemmän dataa siirretään internetin ylitse. Tämä tuo mukanaan riskejä verkkohyökkäysten muodossa. Verkkohyökkäyksellä tarkoitetaan verkossa tehtävää hyökkäystä, jonka tarkoituksena voi olla esimerkiksi tiedonhankinta, kiristys tai ilkivalta. Verkkohyökkäyksen tyyppejä on useita, ja ne voidaan jakaa kahteen ryhmään, passiivisiin ja aktiivisiin.

Hyökkäys on passiivinen, jos hyökkääjä kuuntelee tai seuraa verkkoliikennettä kuitenkaan manipuloimatta tai muuttamatta sitä (Ryck et al. 2014). Passiivisiin verkkohyökkäyksiin kuuluu muun muassa liikenteen kuuntelu (engl. eavesdropping). Verkkoliikennettä voidaan seurata kannettavan langattoman verkon (engl. hotspot) avulla, vaikka käytössä oleva langaton verkko (engl. Wi-Fi) olisi suojattu salasanalla (Hodges et al. 2012). Kuuntelemalla liikennettä hyökkääjä näkee kaikken http:n yli siirrettyn datan, sillä osapuolten välistä liikennettä ei ole salattu. Esimerkiksi hotspotin pystyttäminen julkiseen langattomaan verkkoon mahdollistaa hyökkääjälle saada tietoonsa kaikki, mitä verkon käyttäjät tekevät internetissä, mikäli yhteys ei ole suojattu TLS:llä. Hyökkäyksen avulla on mahdollista tarkastella esimerkiksi vierailtuja verkkosivuja tai syötteitä, kuten käyttäjätunnuksia ja salasanoja sekä evästeitä. Verkon kuuntelu on hyvin suoraviivainen hyökkäys. Erilaiset ohjelmistot, kuten Firesheep tai Ettercap tekevät hyökkäyksestä helpon toteuttaa. Hyökkäys vaatii käytettävän ohjelman lisäksi vastaanottimen, jolla voidaan kuunnella ympärillä tapahtuvaa verkkoliikennettä (Ryck et al. 2014). Toisin sanoen kaikki http:n yli siirretty data on luettavissa yhden ohjelman avulla, ja kenen tahansa on mahdollista sitä kuunnella.

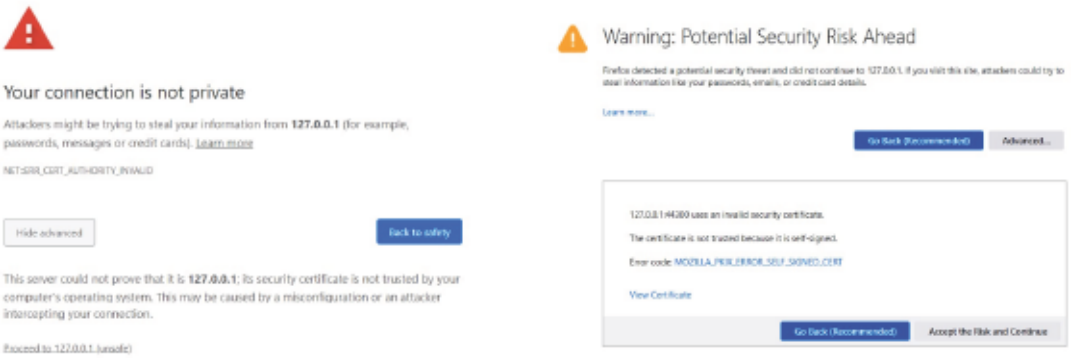
Aktiiviset hyökkääjät pyrkivät soluttautumaan verkkoliikenteeseen osapuolten väliin esittäytymällä toisena osapuolena esimerkiksi väärentämällä DNS-nimipalvelun osoitteen langattoman verkon sisältä (Hodges et al. 2012; Ryck et al. 2014). Aktiivisen hyökkäyksen tarkoituksena on tunkeutua osapuolten väliseen verkkoliikenteeseen ja manipuloida aktiivisesti osapuolten välillä siirrettyä dataa. Aktiivinen hyökkääjä pystyy lukemaan, hallitsemaan ja estämään kaikkea suojaamattoman verkon yli siirrettävää liikennettä. (Ryck et al. 2014) Aktiivisen hyökkäyksen toteuttaminen vaatii mies välissä (engl. Man-in-The-Middle, MiTM) -verkkohyökkäyksen, jossa kolmas osapuoli tunkeutuu verkossa keskustelevien osapuolten väliin.

2.4 Mies välissä -hyökkäys

Mies välissä (engl. Man-in-The-Middle, MiTM) -hyökkäyksen tarkoituksena on seurata ja/tai manipuloida verkkoliikennettä ja saada haltuunsa esimerkiksi salasanoja tai muita hyökkääjälle hyödyllisiä tietoja. Esineiden internetissä MiTM-hyökkäyksen avulla on myös mahdollista manipuloida laitteiden ohjauksia ja saada ne toimimaan odottamattomalla tavalla. Mies välissä alkaa yhteyden aikaisessa vaiheessa eli luvussa 2.1.1 kuvaillussa TCP:n kättelyvaiheessa, jossa osapuolet eivät vielä tunne toisiaan. Tässä vaiheessa kolmas osapuoli esittäytyy toisena osapuolena yhteyden molempiin suuntiin. Suojaamattoman http-yhteyden ylitse hyökkäys on helppo toteuttaa, sillä osapuolten välinen liikenne ei ole salattua. Alkuperäinen yhteys voidaan kaapata helposti, kun TLS-sertifikaatteja ei tarvitse väärentää. (Ryck et al. 2014)

TLS:n käyttö HTTP:n päällä vaikeuttaa tämän MiTM-hyökkäyksen toteutusta, kun data on salattu avaimilla, jotka ovat osapuolten välisiä salaisuuksia. Nämä salaisuudet määritetään TLS:n kättelyssä luvussa 2.1.2 kuvaillun TLS-sertifikaatin avulla. Yksinään salattu liikenne ja vaatimus sertifikaateista eivät kuitenkaan takaa, että MiTM-hyökkäys olisi mahdoton toteuttaa (Dolnák ja Litvik 2017). Edellisen luvun kuvassa 2.3 esitetyssä uudelleenohjauksessa yksi pyyntö ja vastaus lähetetään suojaamattoman verkon ylitse, jolloin hyökkääjän on mahdollista kaapata selaimen lähettämä pyyntö ja välittää se eteenpäin toiselle osapuolelle manipuloituna.

HTTPS:ään toteutettu hyökkäys vaatii, että käyttäjä hyväksyy virheellisen sertifikaatin, jolloin hyökkääjä saa käyttäjän käyttämään HTTP:tä HTTPS:n sijasta. Selain tunnistaa, että kyseessä ei ole luotettava sertifikaatti ja varoittaa käyttäjää. Varoitus voidaan ohittaa (esitetty kuvassa 2.4), jolloin verkkosivua päästään käyttämään HTTP:n kanssa ilman TLS:n mukaista suojausta. Callegati et al. 2009 mukaan ainoa tapa todistaa väärennetty varmenne tai verkkosivu on tarkistaa verkkotunnukselle myönnetty sertifikaatti selaimen osoitekentän reunasta, jota harva käyttäjä kuitenkaan tekee.



***Kuva 2.4.** Varoitus virheellisestä sertifikaatista. Vasemmalla Chromen ja oikealla Firefoxin varoitus.*

Tämän sertifikaatin ongelmasta ilmoittaman varoituksen ohittaminen on ainoa edellytys joutua SSL-stripping tyyppisen mies välissä -hyökkäyksen uhriksi, jossa yhteydestä ohitetaan TLS-protokolla. SSL-stripping hyökkäys suojattuun HTTPS-yhteyteen rakennetaan hyökkääjän näkökulmasta seuraavasti (Callegati et al. 2009):

1. Esitä reititintä (gateway) selaimelle LAN-verkossa (MiTM)
2. Välitä selaimen pyyntö palvelimelle ilman väliintuloa
3. Kaappaa palvelimen vastaus reitittimestä ennen selainta
4. Muodosta väärennetty, itse allekirjoitettu sertifikaatti alkuperäisen tilalle
5. Lähetä väärennetty sertifikaatti selaimelle
6. Kun käyttäjä hyväksyy väärennetyn sertifikaatin ohittamalla selaimen varoituksen, muodostetaan suojattu yhteys sekä selaimen ja hyökkääjän että hyökkääjän ja palvelimen välille.

Hyökkäyksen onnistuminen vaatii ainoastaan selaimen varoituksen ohittamisen, jonka jälkeen hyvin toteutetussa hyökkäyksessä yhteys on näennäisesti kunnossa, kun sertifikaatti on väärennetty, ja selain luulee keskustelevansa palvelimen kanssa. (Callegati et al. 2009) Hyökkäys on suoraviivainen, ja näiden vaiheiden jälkeen hyökkääjä voi purkaa kaikki suojatun verkon yli lähetettävät viestit, sillä hän on saanut käsiinsä molempien osapuolten TLS-protokollan käyttämät salaus- ja purkuavaimet.

3. STRICT TRANSPORT SECURITY

Edellisessä luvussa kuvailtiin yksityiskohtaisesti suojatun internetyhteyden muodostus käyttäen palvelimen puolen vastausta, joka uudelleenohjaa käyttäjän selaimen suojattuun https-yhteyteen. Lopputuloksena osapuolet ovat saavuttaneet TLS-protokollan mukaisen suojatun yhteyden. Yhteys on kuitenkin muodostettu käyttäen ensin suojaamatonta http-yhteyttä, mikä mahdollistaa mies välissä -hyökkäyksen. Tällä tavoin muodostettu suojattu yhteys muodostetaan joka kerta, kun verkkotunnusta pyydetään ilman https-etuliitettä. Näitä hyökkäyksen mahdollistavia aikaikkunoita syntyy useissa tilanteissa, esimerkiksi hakemalla verkkosivua ilman https-etuliitettä tai seuraamalla verkkosivun sisäisiä linkkejä, jotka ovat kokonaan ilman etuliitettä. Http Strict Transport Security -käytäntö on suunnattu ratkaisemaan tämä edellisessä luvussa esitelty riski.

HTTP Security Headers ovat nimensä mukaisesti verkkoliikenteeseen tarkoitettuja turvallisuutta parantavia HTTP-ylätunnisteita, joita käytetään osapuolten välisen tietoturvan parantamiseen. HTTP Strict Transport Security (HSTS) on yksi tähän ryhmään kuuluva ylätunniste, jonka Internet Engineering Task Force (IETF) määritti standardiksi RFC6797 lokakuussa 2012 (Hodges et al. 2012). HTTP Strict Transport Security ratkaisee uudelleenohjauksien tuoman MiTM-hyökkäyksen riskin pakottamalla osapuolet keskustelemaan suojatulla HTTPS-protokollalla. IETF:n standardin mukaan HSTS-käytännön ollessa käytössä, suojaamaton HTTP-liikenne on kokonaan kielletty osapuolten välillä. HSTS-ylätunniste tallentuu UA:n välimuistiin, ja sen voimassaoloaikana pyynnöt palvelimelle tulee tehdä suojattuna https-alkuisena. Vastuu suojatun yhteyden muodostamisesta siirretään ylätunnisteen mukana UA:lle. (Hodges et al. 2012)

HSTS:n ensisijainen tarkoitus on vähentää luvussa 2.4 esitetyn MiTM-hyökkäyksen riskiä, joka on toteutettu kuvailulla *SSL-stripping*-tekniikalla. HSTS on myös hyvä käytäntö torjumaan evästeiden kaappauksia (engl. cookie hijacking), kun kaikki osapuolten välinen liikenne on salattu. (Kiyani 2017) Evästeet saattavat sisältää esimerkiksi autentikointiin käytettyjä käyttäjätunnuksia ja salasanoja, jotka HTTPS-yhteydessä salataan TLS:n varmenteen määräämällä tavalla. HSTS on palvelimen HTTP-protokollan konfiguraatio ja se voidaan ottaa käyttöön määrittämällä ylätunniste seuraavasti:

```
Strict-Transport-Security: max-age=31536000 .
```

Palvelin lähettää yhteyden ensimmäisen HTTPS-vastauksen mukana HSTS-ylätunnisteen,

joka kertoo käyttäjäagentille ohjeet muodostaa yhteys tästä eteenpäin aina suojattua HTTPS-protokollaa käyttäen, *max-age*-parametrin ilmoittaman ajan (sekunteina).

HSTS-tunniste tulee lähettää suojatun HTTPS-protokollan ylitse. Suojaamattoman HTTP-yhteyden yli lähetettynä ylätunniste tulee hylätä käyttäjäagentin toimesta, eikä standardin mukaan sitä saa lukea (Hodges et al. 2012). Ylätunnisteen lukemisen jälkeen tieto on tallennettu käyttäjäagentin välimuistiin (engl. cache), ja suojaamattoman HTTP-yhteyden muodostaminen tähän osoitteeseen on kielletty, kunnes

- HSTS-otsikko tuhotaan palvelimen toimesta asettamalla *max-age*-parametri nollaan, joka yli kirjoittaa vanhan ylätunnisteen tai
- HSTS-otsikko vanhenee, eli edellisestä vierailusta verkkosivulla on kulunut yli *max-age*:n ilmoittama aika,

ja jos käyttäjäagentti ei onnistu muodostamaan suojattua HTTPS-yhteyttä palvelimeen, yhteys tulee katkaista. (Hodges et al. 2012; Nidecki 2019)

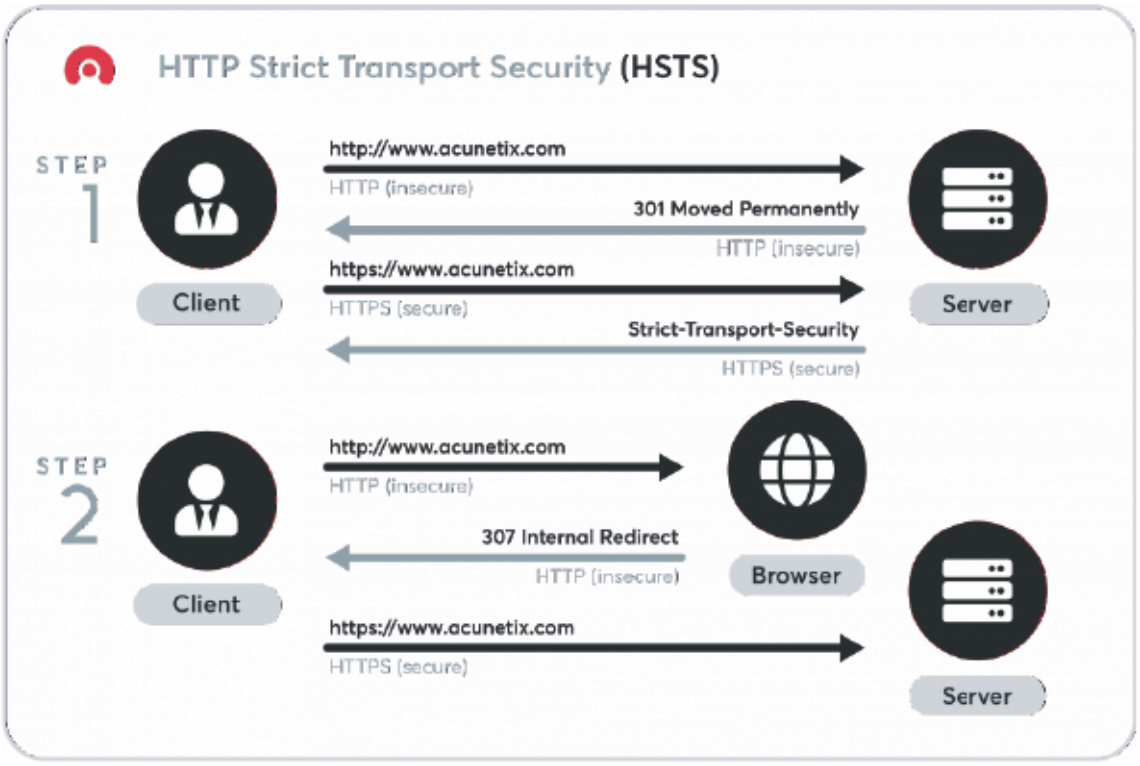
Palvelimella voi olla määritettynä useita eri verkkotunnuksia (engl. domain), jossa yhdellä verkkotunnuksella (*example.com*) voi olla useita aliverkkotunnuksia (engl. subdomain, kuten
foo.example.com). HSTS-käytäntö voidaan ottaa käyttöön jokaisen verkkotunnuksen aliverkkotunnuksella lisäämällä ylätunnisteen määritelmään parametri *includeSubDomains* seuraavalla tavalla:

```
Strict-Transport-Security: max-age=31536000; includeSubDomains ,
```

joka kertoo käyttäjäagentille ohjeet muodostaa yhteys tästä eteenpäin ainoastaan suojattuna myös verkkotunnuksen jokaiseen aliverkkotunnukseen. (Hodges et al. 2012) HSTS määritetään aina tietylle verkkotunnukselle. Esimerkiksi määrittämällä HSTS käyttöön verkkotunnukselle *www.example.com* ilman *includeSubDomains* parametria, verkkotunnus *example.com* ei käytä HSTS-käytäntöä, vaan ainoastaan sen www-alkuinen tunnus. (Nidecki 2019) Kattavan suojauksen saavuttamiseen usean verkkotunnuksen sisällyttäminen HSTS-käytäntöön on ehdottoman tärkeää (Hodges et al. 2012).

Luvussa 2 kerrottiin sertifikaattien tarkoituksesta TLS-protokollan yhteydessä. Selain varoittaa käyttäjää, mikäli sertifikaatissa on jokin tunnistettu ongelma. Näitä ovat muun muassa vanhentuneet, itse allekirjoitetut tai tuntemattoman CA:n myöntämät sertifikaatit. Mikäli HSTS on käytössä, varoituksen ohittaminen ei ole mahdollista, sillä virheellisen sertifikaatin ohittaminen johtaa suojaamattomaan HTTP-yhteyteen, ja HSTS-standardin mukaan HTTP ei ole sallittu. (Hodges et al. 2012; Nidecki 2019)

HSTS-ylätunniste tallentuu käyttäjän selaimen välimuistiin ensimmäisellä yhteydenotto-kerralla. HTTP-ylätunnisteet luetaan viesteistä aina, ja uudempi ylikirjoittaa vanhan, jolloin tunniste pysyy voimassa. Jos käyttäjäagentti ei ole tietoinen palvelimen HSTS-käytännöstä (ei aiempia yhteydenottopyyntöjä), ensimmäinen pyyntö verkkotunnukseen sekä palvelimen vastaus tehdään suojaamattomana, kuvan 3.1 vaiheen yksi mukaisesti. Selain tunnistaa palvelimen vastauksen uudelleenohjauksesta, jonka jälkeen selain pyytää https-alkuista verkkosivua. Palvelin vastaa palauttamalla pyydetyn tiedon HTTPS-yhteyden yli, jonka mukana palvelin lähettää HSTS-tunnisteen. (Hodges et al. 2012) HSTS-ylätunnisteen lukemisen jälkeen kaikki pyynnöt suojaamattomaan verkkotunnukseen ohjataan käyttäjäagentin puolesta suoraan https-muotoisena, kuvan 3.1 vaiheen 2 mukaisesti, eikä tästä eteenpäin vaadita uudelleenohjauksia tunnisteen voimassaoloaikana.



***Kuva 3.1.** HSTS:n toiminta (Nidecki 2019).*

Ensimmäinen yhteydenmuodostus on edelleen suojaamaton, kun UA ei ole tietoinen HSTS-käytännöstä, mikä jättää riskin verkkohyökkäyksille. Palvelimella on kuitenkin mahdollisuus poistaa tämä riski käytöstä ottamalla käyttöön parametri *preload*:

```
Strict-Transport-Security: max-age=xxx; includeSubDomains; preload,
```

mikä tarkoittaa, että verkkotunnus on lisätty *HSTS preload* -listaan, jonka selain tarkistaa aina, ennen kuin se lähettää pyynnön palvelimelle. Mikäli palvelimen verkkotunnus löytyy preload-listasta, yhteys muodostetaan aina HTTPS-protokollan yli, myös ensimmäisellä yhteydenmuodostuksella. (De los Santos ja Torres 2018) Preload ei ole osa IETF:n määrittämää standardia, vaan osa avoimen lähdekoodin Chromium projektia (Chromium

2021). Preload-käytäntö on käytössä suuressa osassa selaimista (Chrome, Firefox, Safari, Opera, IE11 ja Edge) (Nidecki 2019).

HSTS:n käyttöönotolla saavutetaan myös yllättävä vaikutus verkkosivun latausajoissa. Kun käytäntö pakottaa käyttämään HTTPS-protokollaa, vältytään selaimen ja palvelimen välisiltä uudelleenohjauksilta. Näiden uudelleenohjauksien välttäminen nopeuttaa selaimen pyyntöjen odotusaikaa. Saavutettu hyöty saadaan kuitenkin vain silloin, kun palvelimen osoitetta pyydetään http-alkuisena tai kokonaan ilman https-etuliitettä. Hyöty latausajan nopeutumiseen sekä turvallisuuteen saavutetaan myös, mikäli verkkosivun palvelimen kehittäjä on asettanut verkkotunnuksen sisäisten linkkien referenssejä http-alkuisena. (Hodges et al. 2012)

Käyttäjäagentilla ei ole juurikaan merkitystä käytännön kannalta, sillä kaikki modernit selaimet tukevat HSTS-käytäntöä. Yhteensopivuusongelmia voi kuitenkin esiintyä selainten vanhemmissa versioissa. Eniten käytetyt modernit selaimet, kuten Chrome, Edge, Firefox, tukevat kaikki sekä HSTS-käytäntöä että Chromium projektin preload-parametrin käyttöä. HSTS-tuki löytyy myös Android- ja iOS-selaimista. (MDN 2021)

4. HSTS:N MERKITYS ESINEIDEN INTERNETISSÄ

Luvussa 3 todettiin HSTS-käytännön toimivan TCP/IP-protokollaperheen päällä, HTTP-protokollan ylätunnisteena. Automaatiossa tämän vaatimuksen täyttää esimerkiksi esineiden internet (engl. Internet of Things, IoT), johon tutkimuskysymyksen selainpohjainen automaatio rajattiin. Esineiden internet on järjestelmä, joka kytkee antureita, sensoreita ja ohjelmistoja omaavia laitteita internetiin, jossa ne jakavat tietoa toistensa sekä niiden käyttäjien kanssa. Internetin yli siirretyn datan määrä kasvaa, mitä enemmän laitteita siihen kytketään. Samalla korostuu tietoturvan merkitys, sillä data pitää sisällään yhä enemmän yksilöille ja yrityksille tärkeää tietoa, käyttäjätunnuksista ja salasanoista aina kulunvalvonnan tai tuotantoprosessin tietoihin ja salaisuuksiin. Tässä luvussa tarkastellaan turvallisuusvaatimuksia ja HSTS-käytännön käyttökohteita ja -tarkoituksia esineiden internetissä.

4.1 Automaatioverkon turvallisuusvaatimukset

Automaatioverkot ja datan arkaluontoisuus itsessään korostavat verkon turvallisuusvaatimuksia. Yleinen tapa kuvata järjestelmän turvallisuutta on RAMS-akronyymi, jonka kirjaimet tulevat termeistä Realiability (luotettavuus), Availability (saatavuus/käytettävyys), Maintanability (ylläpidettävyys) ja Safety (turvallisuus). Järjestelmän oikeellisen toiminnan kannalta jokainen termi on tärkeä. Luotettavuudella tarkoitetaan järjestelmän kykyä toimia oikein ja juuri niin kuin sen pitäisi sille annetuissa olosuhteissa. Esimerkiksi oikeellinen virheiden käsittely on tärkeä osa luotettavuutta, sillä usein järjestelmien tulee toimia pitkiä aikoja ilman mahdollisuuksia uudelleenkäynnistyksiin. (Gill et al. 2013) Automaatioverkot toimivat usein ympäristöissä, joissa turvallisuus on taattava ja järjestelmiltä odotetaan korkeaa luotettavuutta (Varga et al. 2017).

Saatavuus kuvaa järjestelmän, sen laitteen, ohjelman tai muun palvelun saatavuutta olla käytössä, eli sen tulee toimia sille määritetyssä käyttöympäristössä. Hyvällä ylläpidettävyydellä taataan järjestelmän onnistunut korjaaminen tai päivittäminen tarpeen vaatiessa. RAMS-termien turvallisuus kattaa tietoturvan lisäksi myös fyysisen turvallisuuden. (Gill et al. 2013)

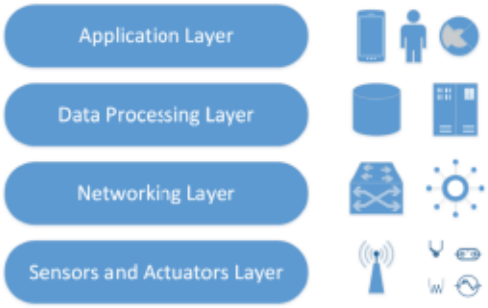
Tietoturva aiheuttaa oman vaatimuksensa, ja sillä on suuri merkitys etenkin automaatioverkoissa. IoT-sovelluksissa loppukäyttäjän ja palvelimen välillä siirretty data sisältää paljon henkilökohtaista ja arkaluontoista tietoa, ja laitteiden lisääntyessä sen määrä kasvaa jatkuvasti. Hyökkääjän kannalta data on myös yhä merkittävämpää. Tietoturvan merkitys on siis suuri, ja se kattaa

- tiedon luottamuksellisuuden, eli tieto on vain sen käyttöön oikeutettujen tahojen saatavilla,
- eheyden, eli tietoja voi muokata vain siihen oikeutetut tahot tai henkilöt ja
- käytettävyyden eli sen, että tiedot ja järjestelmät ovat käyttöön oikeutettujen tahojen hyödynnettävissä. (Kyberturvallisuuskeskus 2020)

Nämä vaatimukset ja datan luonne edellyttävät järjestelmiltä tiukkoja tietoturvatoimenpiteitä (Varga et al. 2017). Edellä esitetyistä turvallisuuden vaatimuksista HSTS-käytäntö vaikuttaa kaikista eniten tiedon luottamuksellisuuteen, mutta sillä on vaikutuksia myös muihin vaatimuksiin. Mies välissä -hyökkäys vaarantaa jokaisen edellä mainitun kohdan, kun kolmas osapuoli on tunkeutunut liikenteeseen. Hyökkäys vaarantaa myös järjestelmän saatavuutta sekä luotettavuutta.

4.2 Käyttökohteet automaatiossa

Tutkimuksessa selainpohjainen automaatio rajattiin esineiden internetiin, mutta HSTS-käytäntöä voidaan hyödyntää muissakin verkoissa. Vaatimuksena on TCP/IP-verkko, joka käyttää HTTP:tä tiedonsiirtoon. Verkon on myös kyettävä siirtämään kaikki data HTTPS-protokollan ylitse. Esineiden internetissä toimivat sovellukset täyttävät tämän vaatimuksen, mikäli ne keskustelevat HTTP:n avulla internetin ylitse. Kuvassa 4.1 on esitettynä eräs arkkitehtuurinen kerrostus IoT-verkosta, jossa kerrokset keskustelevat viereisten kerrosten kanssa.



***Kuva 4.1.** IoT-järjestelmän arkkitehtuuriset tasot. (Varga et al. 2017 s. 2)*

Kuvan mukaisen arkkitehtuurin alin kerros pitää sisällään järjestelmän laitteet. Laitteet ovat kytkettyinä lähiverkon reitittimiin tai kytkimiin jonkin lähikenttäratkaisun, kuten Ethernet, Bluetooth, NFC tai Zigbee protokollien avulla. (Varga et al. 2017) Tietoa siirretään edelleen datan käsittelykerrokseen esimerkiksi TCP- tai UDP-protokollaa käyttäen. Dataa voidaan käsitellä palvelimilla tai hyödyntää suoraan laitteiden välisessä (engl. Machine-to-Machine, M2M) kommunikaatiossa. (Varga et al. 2017) HSTS-käytäntö tarvitsee kuitenkin TCP/IP-protokollaperheen protokollia, jotta sitä voidaan hyödyntää. Tämän vaatimuksen myötä HSTS-käytäntöä ei voida soveltaa esimerkin mukaisen IoT-verkon alimmissa kerroksissa.

Esimerkin arkkitehtuurin ylin kerros Application layer on käyttöliittymä eli ihmisen ja koneen rajapinta (engl. Human-Machine Interface, HMI). Palvelimet tarjoavat esineiden internetin tuottamaa dataa ja ohjausmahdollisuuksia näihin käyttöliittymiin. Tiedonsiirto palvelimen ja käyttöliittymän välillä voidaan toteuttaa esimerkiksi HTTP- tai MQTT-protokollalla (Yokotani ja Sasaki 2016). MQTT on sovelluskerroksen protokolla, joka on suunniteltu etenkin esineiden internetin käyttämiin verkkoihin. Sen etuna on publish/subscribe (pub/sub) -tyylinen yksisuuntainen kommunikaatio, joka on vähemmän verkkoa rasittava protokolla kuin HTTP. HSTS-käytäntöä voidaan kuitenkin soveltaa vain TCP/IP-verkoissa, jotka käyttävät HTTP:tä tiedonsiirtoon, jolloin MQTT-protokollaa käyttävät verkot ovat käytännön ulkopuolella (Hodges et al. 2012; Yokotani ja Sasaki 2016). Selainpohjaisessa automaatiossa datan siirtoprotokollana käytetään usein HTTP:tä.

4.3 HSTS-käytännön vaikutukset automaation tietoturvaan

HSTS auttaa torjumaan mies välissä -hyökkäyksiä kieltämällä HTTP-liikenteen verkon tyypistä riippumatta. Se ei kuitenkaan ota kantaa muihin kuin HTTP:n yli siirrettyyn liikenteeseen. Esimerkiksi lähikenttäratkaisut ja datan siirto kytkimiltä tai reitittimiltä palvelimille voivat olla käytännön ulkopuolella, mikäli nämä osapuolet ovat eri verkossa tai ne eivät keskustele HTTP-protokollan mukaisesti. Esineiden internetissä alemmissa kerroksissa käytetään myös muita protokollia HTTP:n sijaan, joihin käytäntöä ei voida soveltaa (Cirani 2019).

HSTS-käytäntö ei tarjoa päästä-päähän (end-to-end) -salausta, vaan ainoastaan pakottaa suojauksen asiakas-palvelin (client-server) -yhteyteen. TLS-suojaus aktivoidaan ja puretaan palvelimella, minkä takia salaus on aktiivinen vain palvelimen ja asiakaslaitteen välillä (Rescorla 2018). Mikäli verkko ja sen laitteet kykenevät keskustelemaan HTTPS-yhteyden ylitse, ei HSTS-käytännön käytölle ole esteitä. Internetin turvallisuuteen perustava avoin verkkoyhteisö Open Web Application Security Project (OWASP) tarjoaa verkkosivuillaan kymmenen yleisintä palvelinten haavoittuvuutta, joista vuonna 2021 toiseksi yleisin on *Cryptocraphic Failures*. Riskin pienentämiseen OWASP ehdottaa TLS-protokollan käyttöä ja sen vahvistamista HSTS-käytännöllä. (Open Web Application Security Project 2022)

OWASP:n suositusta tukee vuonna 2019 toteutettu penetraatiotestaus kuluttajille suunnattuihin IoT-sovelluksiin (Shakdher et al. 2019). Testauksessa oli mukana laaja skaala esineiden internetin sovelluksia, joista tutkimuksessa on mainittuna älykoti- ja turvallisuusratkaisut sekä hyvinvointiapplikaatiot ja internet-yhteydessä olevat autot. Penetraatiotestauksessa käytettiin hyödyksi OWASP:n määrittämiä merkittävimpiä haavoittuvuuksia, joihin kohdennettiin erilaisia mies välissä -hyökkäyksiä. Tutkimuksen tulokset osoittavat, että SSL stripping -tekniikalla suoritettu TLS-protokollan ohittaminen onnistui 13:sta testissä testatuista 28:sta sovelluksesta. (Shakdher et al. 2019) Näissä 13:sta palvelin saatiin keskustelemaan HTTP:n ylitse, jolloin käyttäjätunnukset voitiin lukea suoraan selkotekstinä.

Penetraatiotestauksen toinen hyökkäysvektori hyödynsi TLS-sertifikaatin väärennystä, jolloin hyökkäjä esittää, että yhteys on luotettava. Tässä hyökkäysvektorissa tutkijoiden hyökkäys onnistui 27:ään kaikista 28:sta sovelluksesta. Tutkimuksen tuloksissa todetaan, että suuressa osassa tutkituista sovelluksista tekijät ovat yrittäneet käyttää HTTPS:ää, mutta monessa näistä sovelluksista virhe löytyi konfiguraatiosta. Tutkimuksessa päädytään samaan lopputulokseen, jota OWASP suosittelee: sovellusten tietoturvaa voidaan parantaa oikeaoppisella HSTS-käytännöllä mies välissä -hyökkäyksiä ja evästeiden kaappausta vastaan. (Shakdher et al. 2019)

HSTS-käytännön käyttökohteet ovat siis kohtalaisen tarkasti rajattuja, mutta datan arkaluontoisuudesta sekä korkeasta luotettavuus- ja saatavuusvaatimuksista johtuen HSTS:n käyttö on osa kattavaa tietoturvaa. Turvallisuusparannusten lisäksi palvelimen uudelleenohjauksien vähenemisen myötä selaimessa käytetty odotusaika sekä verkon kuormitus vähenevät. Tällä voi olla vaikutuksia hitaissa verkoissa tai reaaliaikavaatimuksen omaavissa automaatiosovelluksissa. Verkon kuormituksen ja odotusaikojen väheneminen on kuitenkin vähäistä ja vain sivuhuomio HSTS-käytännössä.

4.4 Haasteet automaatioverkoissa

Vaikka HSTS-käytännöllä saavutetaan hyötyjä loppukäyttäjän ja palvelimen välisen liikenteen suojaamisessa, automaatioverkoissa toimiva Strict Transport Security saattaa tuottaa haasteita. Ensimmäisenä on pakotettu HTTPS, jolloin HTTP:tä sallita. Kaikki osapuolten välinen liikenne pitää pystyä toteuttamaan HTTPS-protokollan yli. Etenkin vanhemmissa verkoissa kaikki laitteet eivät välttämättä tue TLS-protokollaa, ja esimerkiksi vikatilanteissa datapakettien sisältö on salattu ja virheilmoitusten selvitys on haastavaa datan kryptauksesta johtuen. Salauksesta johtuen vikatilanteet voivat vaikuttaa järjestelmän saatavuuteen ja ylläpidettävyyteen, kun järjestelmän korjaus voi kestää pidempään.

HTTPS-protokollan yli siirretty data rasittaa verkkoa hieman HTTP:tä enemmän, mutta sen vaikutus on kuitenkin pieni. Suurempi vaikutus syntyy kuvassa 2.2 esitetyssä TLS-protokollan kättelyvaiheessa. HTTP-yhteyden muodostus (TCP-kättely) suoritetaan yhdellä viestiparilla ja avainten vaihdon vuoksi TLS-kättely vaatii kaksi. HTTP-yhteys suljetaan oletuksena heti, kun pyyntöön on vastattu, ja TLS-kättelyn aikana asiakkaan on lähetettävä sertifikaatti palvelimelle aina. (Rescorla 2018) Verkkoa rasittavia haasteita voi syntyä mikäli yhteyksiä avataan usein, kun yhteyden muodostus vaatii kolminkertaisen määrän liikennettä. Tämä rasitus ylittää HSTS-käytännön hyödyn uudelleenohjausten vähentymisestä ja sillä voi olla vaikutusta järjestelmän toimintaan tai reaaliaikaisuuteen.

Automaatioverkot voivat olla eriytettyjä ja ulkopuolisilta suljettuja, jolloin sillä ei ole julkista verkkotunnusta. Eriytetyn verkon TLS-sertifikaatit voivat olla itse allekirjoitettuja. Mikäli sertifikaatti on itse allekirjoitettu, selain varoittaa käyttäjää mahdollisesti väärennetystä sertifikaatista, kuvan 2.4 mukaisesti. Itse allekirjoitetun varmenteen tilanteessa verkon käyttäjä on usein tottunut ohittamaan varoituksen, eikä varoituksesta voida erottaa, onko sertifikaatti itse allekirjoitettu vai onko se varoitus alkaneesta MiTM-hyökkäyksestä. Mikäli HSTS otetaan käyttöön tässä yhteydessä, yhteyden muodostus epäonnistuu HSTS-käytännön mukaisesti, kun HTTP-liikenne on kielletty. (Hodges et al. 2012)

Itse allekirjoitettujen sertifikaattien tapauksissa HSTS voidaan kuitenkin ottaa käyttöön lisäämällä sertifikaatin myöntäjä (CA) suoraan käyttäjän selaimen tai käyttöjärjestelmän luotettujen sertifikaattien myöntäjien hakemistoon (engl. trusted root certification store). Virheellisen sertifikaatin tarkastelu tarjoaa vihjeen tähän, joka on esitettynä kuvassa 4.2. Myöntäjän tallentaminen suoraan käyttöjärjestelmään vaatii CA:n tietojen ja sertifikaattien siirtämistä ja asentamista suoraan käyttäjän laitteeseen. (Hodges et al. 2012)

Kuva 4.2. *Virheellisen sertifikaatin tarkastelu osoittaa, että luottamus sertifikaatin myöntäjään saadaan lisäämällä se käyttöjärjestelmän luotettujen tahojen varastoon.*

Kuluttajille suunnatuissa sovelluksissa loppukäyttäjiä ei tulisi opastaa varmenteiden lisäämiseen manuaalisesti, sillä sertifikaatit ovat TLS-salauksen perusta, joten käyttöjärjestelmään asennettu väärennetty sertifikaatti tuottaisi riskejä. Eriytetyissä verkoissa käytetyt laitteet ovat usein rajattuja, jolloin vastuu myöntäjän lisäämisestä olisi järjestelmän ylläpitäjällä. HSTS on siis mahdollista saadaa käyttöön myös itse allekirjoitettujen sertifikaattien käyttämillä verkkotunnuksilla. Verkkotunnus voidaan avata vain laitteilla, joihin sertifikaatin myöntäjän tiedot ovat lisätty manuaalisesti. Tämä voi tuottaa haasteita avoimen verkkotunnuksen omaavissa sovelluksissa, mutta suljetuissa ympäristöissä kaikki ulkopuoliset laitteet ovat suljettu pois, eikä niillä voida muodostaa yhteyttä palvelimeen. Laitteiden tarkka rajaaminen voi parantaa tietoturvaa, kun käyttäjälaitteet on määritetty tapauskohtaisesti.

5. YHTEENVETO

HTTP Strict Transport Security (HSTS) on palvelimen ja selaimen välille muodostetun internetyhteyden HTTP-ylätunniste. HSTS-käytäntö luokiteltiin RFC-standardiksi vuonna 2012, ja sen tarkoitus on pakottaa osapuolet pitäytymään TLS-protokollan mukaisesti suojatussa HTTPS-yhteydessä. Standardin mukaan suojaamaton HTTP-yhteys on kielletty kaikissa tapauksissa. Käytännön tarkoitus on ratkaista verkkohyökkäyksien riskit, joita syntyvät, kun osapuolet siirtävät tietoa suojaamattoman HTTP-protokollan ylitse. Ratkaistaviin riskeihin sisältyy erityisesti mies välissä (MiTM) -hyökkäys sekä evästeiden kaappaus.

Automaatiossa esineiden internet on eräs käyttökohde HSTS-käytännölle. Antureilla, sensoreilla ja ohjelmistoilla varustetut laitteet kytketään lähiverkon kytkimiin, jotka välittävät laitteilta saadun datan palvelimille. Palvelimella dataa voidaan prosessoida esitettävään muotoon, josta loppukäyttäjä voi selaimella tai mobiilisovelluksella pyytää dataa internetin ylitse HTTP:tä käyttäen. Näiden osapuolten välinen liikenne voidaan pakottaa suojattuun HTTPS:n mukaiseen yhteyteen HSTS-käytännön avulla.

Kuluttajille suunnatuissa IoT-sovelluksissa siirretty data sisältää paljon yksityistä tietoa, kuten terveys- tai kulunvalvontatietoja. Tehtaisiin suunnatuissa Industrial IoT -ratkaisuissa data sisältää tietoja prosessista ja voi mahdollistaa laitteiden ohjausta, mikä vaarantaa järjestelmän oikeellista toimintaa. IoT-verkoissa datan määrä on suuri, ja sen arkaluontoisuus korostaa tietoturvan merkitystä. HSTS-käytännöllä suojataan tietoturvaa, joka sisältää etenkin tiedon luottamuksellisuuden, eheyden ja käytettävyyden. Hyöty saavutetaan, kun HTTP-yhteyttä ei sallita, joka vähentää riskiä mies välissä -hyökkäykselle ja evästeiden kaappaukselle. Automaatioverkoissa haasteita voi aiheuttaa itse allekirjoitetut sertifikaatit, joiden vuoksi HSTS-käytäntö ei mahdollista yhteyden muodostusta. Tämä voidaan ratkaista lisäämällä sertifikaatin myöntäjä käyttöjärjestelmän sertifikaattihakemistoon, mikä rajaa tarkasti verkossa toimivat laitteet. Pakotettu HTTPS ja datan salaus voivat aiheuttaa ongelmia myös vikatilanteissa datan salauksesta johtuen, mikä voi vaikeuttaa järjestelmän ylläpitoa. HTTPS rasittaa myös verkkoa HTTP:tä enemmän, johtuen TLS:n kättelyvaiheesta.

HSTS:n käyttö on yleisesti erittäin suositeltavaa, mikäli voidaan varmistua, että osapuolet kykenevät siirtämään kaiken liikenteen suojattuna HTTPS:n ylitse. Open Web Applica-

tion Security Project (OWASP) kerää tietoa internetin suurimmista turvallisuusriskeistä, ja HSTS-käytännön käyttämättömyys kuuluu kymmenen suurimman riskin joukkoon. Tätä tukee myös tutkimuksessa käsitelty kuluttajille suunnattujen IoT-järjestelmien penetraatiotestaus, jossa onnistuttiin pääsemään 27:ään 28:sta testatusta järjestelmästä. Suurin osa testattujen järjestelmien turvallisuusaukoista johtui virheellisestä TLS-protokollan konfiguraatioista. HSTS-käytännöllä voidaan saavuttaa myös latausaikojen nopeutumista ja liikenteen vähenemistä, kun uudelleenohjauksien tarve pienenee. HSTS on hyvä käytäntö suojaamaan osapuolten välistä liikennettä, koska sen periaate on pakottaa osapuolet käyttämään suojattua yhteyttä. Jos suojattu yhteys ei ole mahdollinen, yhteys tulee sulkea välittömästi. HSTS ei kuitenkaan ole yksinään riittävä käytäntö suojaamaan liikennettä, vaan se on hyvä lisä parantamaan tietoturvaa.

Työn tutkimuskysymyksenä oli selvittää, miten HSTS-käytäntö parantaa selainpohjaisten automaatiosovellusten tietoturvaa. Tutkielmassa käsiteltiin melko paljon internetyhteyttä ja siihen vaadittuja protokollia, etenkin HTTPS-yhteyttä, johon tutkittu Strict Transport Security vaikuttaa. Ongelman asettelu ja ratkaisu onnistuivat mielestäni hyvin. Tutkimus tehtiin kuitenkin automaation näkökulmasta, joka osoittautui melko haastavaksi, sillä useassa lähteessä IoT-verkot oli toteutettu muilla kuin TCP/IP-perheen ja HTTP-protokollien avulla, ja siitä johtuen automaation osuus jäi hieman suppeaksi. Käytännön perimmäinen käyttötarkoitus on turvata loppukäyttäjän ja palvelimen välinen tiedonsiirto, joka on esineiden internetissä vain pieni osa suurta kokonaisuutta. Käytännön vaikutus automaatiossa osoittautui kohtalaisen pieneksi, mutta tärkeäksi osaksi kattavaa tietoturvaa.

LÄHTEET

Ala-Mutka, K., Rintala, M., Savikko, V. ja Palviainen, J. (2002). *OSI-malli*. Saatavissa: http://www.cs.tut.fi/etaopetus/titepk/luku19/OSI.html.

Bisson, D. (2019). *What Are Cipher Suites?* Venafi. Verkkosivu. Saatavissa (viitattu 7.4.2021): https://www.venafi.com/blog/what-are-cipher-suites.

Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S. ja Cooper, D. (2008). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC5280, IETF. Saatavissa: https://rfc-editor.org/rfc/rfc5280.txt.

Callegati, F., Cerroni, W. ja Ramilli, M. (2009). "Man-in-the-Middle Attack to the HTTPS Protocol". *IEEE security & privacy*. pp. 78–81.

Chromium (2021). *Chromium Projects*. Verkkosivu. Saatavissa (viitattu 7.4.2021): https://www.chromium.org/chromium-projects.

Cirani, S. (2019). *Internet of things : architectures, protocols and standards*. First edition. pp. 22–76. Hoboken, NJ: Wiley.

Cope, S. (2019). *The TCP/IP Model and Protocol Suite Explained for Beginners*. Verkkosivu. Saatavissa (viitattu 11.11.2021): http://www.steves-internet-guide.com/internet-protocol-suite-explained/.

De los Santos, S. ja Torres, J. (2018). "Analysing HSTS and HPKP implementation in both browsers and servers". *IET Information Security*. pp. 275–284.

Dierks, T. ja Rescorla, E. (2006). *The Transport Layer Security (TLS) Protocol Version 1.1*. RFC4346, IETF. 87 p. Saatavissa: https://rfc-editor.org/rfc/rfc4346.txt.

Dolnák, I. ja Litvik, J. (2017). "Introduction to HTTP security headers and implementation of HTTP strict transport security (HSTS) header for HTTPS enforcing". Teoksessa: *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. pp. 1–4. IEEE.

Gill, K., Yang, S.-H. ja Wang, W.-L. (2013). "Secure remote access to home automation networks". *IET information security* 7.2. pp.118–125. Saatavissa: https://ietresearch-onlinelibrary-wiley-com.libproxy.tuni.fi/doi/epdf/10.1049/iet-ifs.2011.0303.

Hodges, J., Jackson, C. ja Barth, A. (2012). *HTTP Strict Transport Security (HSTS)*. RFC6797, IETF. 46 p. Saatavissa: http://tools.ietf.org/html/rfc6797.

Internet Protocol (1981). RFC791, IETF. 45 p. Saatavissa: http://tools.ietf.org/html/rfc791.

Jackson, B. (2017). *Analyzing HTTPS Performance Overhead*. Verkkosivu. Saatavissa: (viitattu 9.4.2021): https://www.keycdn.com/blog/https-performance-overhead.

Kiyani, S. (2017). "TorSNIP Hidden Service Proxy with End to End Security". Tampere University of Technology. 47 p. Saatavissa: https://trepo.tuni.fi//handle/123456789/25394.

Kyberturvallisuuskeskus (2020). *Tietoturva*. Saatavissa (viitattu 6.4.2021): https://www.kyberturvallisuuskeskus.fi/fi/toimintamme/saantely-ja-valvonta/tietoturva.

Lavrenovs, A. ja Melón, F. J. R. (2018). "HTTP security headers analysis of top one million websites". Teoksessa: *2018 10th International Conference on Cyber Conflict (CyCon)*. pp. 345–370. IEEE.

MDN (2021). *Strict-Transport-Security*. Saatavissa (viitattu 30.10.2021): https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security.

Nidecki, T. (2019). "What Is HSTS and Why Should I Use It?" Verkkosivu. Saatavissa (viitattu 5.4.2021): https://securityboulevard.com/2019/05/what-is-hsts-and-why-should-i-use-it/.

Nielsen, H., Mogul, J., Masinter, L. M., Fielding, R. T., Gettys, J., Leach, P. J. ja Berners-Lee, T. (1999). *Hypertext Transfer Protocol – HTTP/1.1*. Saatavissa: https://tools.ietf.org/html/rfc2616.

Open Web Application Security Project (2022). *OWASP Top Ten*. Verkkosivu. Saatavissa (viitattu 27.3.2022): https://owasp.org/www-project-top-ten.

Postel, J. ja Reynolds, J. (1988). *A Standard for the Transmission of IP Datagrams over IEEE 802 Networks*. RFC948, IETF. URL: https://www.ietf.org/rfc/rfc1042.txt.

Rescorla, E. (2000). *HTTP Over TLS*. RFC2818, IETF. Saatavissa: https://rfc-editor.org/rfc/rfc2818.txt.

Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC8446, IETF. Saatavissa: https://rfc-editor.org/rfc/rfc8446.txt.

Ryck, P. D., Desmet, L., Piessens, F. ja Johns, M. (2014). "Attacks on the Network". Teoksessa: *Primer on Client-Side Web Security*. pp. 33–55. Saatavissa: https://doi.org/10.1007/978-3-319-12226-7_5. Cham: Springer International Publishing.

Shakdher, A., Agrawal, S. ja Yang, B. (2019). "Security Vulnerabilities in Consumer IoT Applications". Teoksessa: *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. 6 p. Saatavissa: https://ieeexplore.ieee.org/abstract/document/8819463.

Transmission Control Protocol (1981). Saatavissa: https://tools.ietf.org/html/rfc793.

User Datagram Protocol (1980). RFC 768. URL: https://rfc-editor.org/rfc/rfc768.txt.

Varga, P., Plosz, S., Soos, G. ja Hegedus, C. (2017). "Security threats and issues in automation IoT". Teoksessa: *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*. 6 p. Saatavissa: https://ieeexplore.ieee.org/document/7991968.

Yokotani, T. ja Sasaki, Y. (2016). "Comparison with HTTP and MQTT on required network resources for IoT". Teoksessa: *2016 International Conference on Control, Electronics,*

Renewable Energy and Communications (ICCEREC). 6 p. Saatavissa: https://ieeexplore.ieee.org/document/7814989.