Ville Ihalainen

CoAP-protokollan tietoturva

Tietotekniikan
pro gradu -tutkielma
8. tammikuuta 2024

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Kokkolan yliopistokeskus Chydenius

Tekijä: Ville Ihalainen
Yhteystiedot: -
Puhelinnumero: -
Ohjaaja: Risto T. Honkanen
Työn nimi: CoAP-protokollan tietoturva
Title in English: Information Security of CoAP protocol
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 77+2
Tiivistelmä: Tutkielman tarkoituksena on selvittää, onko CoAP-protokolla tietoturvallinen. Tutkielmassa tarkastellaan CoAP-protokollan toimintaa, tietoturvaominaisuuksia, protokollaan kohdistuvia tietoturvauhkia, sekä CoAP:n suojauksen toteuttavaa DTLS-protokollaa. Lisäksi rakennetaan testausasetelma CoAP-protokollan tietoturvaominaisuuksien ja siihen kohdistuvien uhkien testaamiseksi. Testausasetelmassa CoAP-protokollaan kohdistetaan IP-osoitteen spoofing ja yksinkertainen vahvistushyökkäys, sekä kaapataan ja tarkastellaan DTLS-suojattua liikennettä. Tutkimusaiheen taustoittamiseksi tarkastellaan CoAP:n lisäksi lyhyesti IoT:ia, sen protokollapinoa ja siihen eri kerroksilla kohdistuvia tietoturvauhkia, sekä CoAP:n kuljetuskerroksella käyttämää UDP-protokollaa.

Tutkimusmenetelminä tutkielmassa sovelletaan kirjallisuuskatsausta ja konstruktiivista tutkimusmenetelmää. Kirjallisuuskatsauksen menetelmin tarkastellaan aiemman tutkimuksen pohjalta CoAP-protokollan ominaisuuksia, tietoturvaominaisuuksia sekä tietoturvauhkia. Testausosiossa käytettävän CoAP-protokollan tietoturvallisuuden testausasetelman luomisessa sovelletaan konstruktiivista tutkimusmenetelmää. Tutkimuskysymykseen vastataan kirjallisuuskatsauksen ja testauksen tulokset yhdistävän analyysin perusteella.

Analyysin perusteella CoAP-protokolla on tietoturvallinen silloin, kun sen turvallisuustiloja sekä DTLS-suojausta käytetään oikein uhkatasoon nähden ja ohjelmistot sekä kirjastot ovat päivitettyjä siten, etteivät ne sisällä tunnettuja haavoittuvuuksia. Lisäksi salauksessa käytettävien avainten generointi ja jakelu sekä bootstrapping-prosessi laitteiden tai järjestelmien käyttöönottojen yhteydessä tulee toteuttaa turvallisesti. Jos DTLS-suojausta ei käytetä, on CoAP-protokolla altis useille eri hyökkäyksille, kuten spoofing, vahvistushyökkäys ja protokollien väliset hyökkäykset.
Avainsanat: CoAP, DTLS, IoT, tietoturvallisuus, spoofing, vahvistushyökkäys, sovellusprotokolla

Abstract: Thesis aims to find out whether the CoAP protocol is secure in terms of information security. The thesis examines the operation and information security features of the CoAP protocol, information security threats to the protocol and the DTLS protocol which implements CoAP protocol's protection. In addition, a testing setup will be built for testing the security features of the CoAP protocol and potential threats against it. In the testing setup, IP-spoofing and simple amplification attack are tested against the CoAP protocol, as well as capturing and examination of DTLS-protected traffic. For background, in addition to CoAP, IoT and its protocol stack and the threats against it at different layers are briefly reviewed, as well as the UDP protocol used by CoAP in the transport layer.

Research methods used in the thesis are literature review and constructive research method. Based on previous research, characteristics, information security features and security threats of the CoAP protocol are examined using the methods of the literature review. A constructive research method is applied in creating the security testing setup for the CoAP protocol used in the testing section. The research question is answered based on an analysis that combines the results of the literature review and testing.

Based on the analysis, the CoAP protocol is secure when security modes and DTLS protection are used correctly in relation to the threat level. Used CoAP implementations and libraries must also be updated so that they do not contain known vulnerabilities. In addition, the generation and distribution of the keys used in encryption, as well as the bootstrapping process in connection with the commissioning of the devices or the system, must be carried out securely. If DTLS protection is not used, the CoAP protocol is vulnerable to several different attacks, such as spoofing, amplification attacks and cross-protocol attacks.

Keywords: CoAP, DTLS, IoT, information security, spoofing, amplification attack, application protocol

Sanasto

AMQP	Advanced Message Queuing Protocol
ARP	Address Resolution Protocol
CoAP	Constrained Application Protocol
DDS	Data Distribution Service
DoS	Denial of Service, palvelunesto
DDoS	Distributed Denial of Service, hajautettu palvelunesto
DTLS	Datagram Transport Layer Security
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things, esineiden internet
IPsec	Internet Protocol Security
Kbit	Kilobitti
Kt	Kilotavu
Mbit	Megabitti
MTU	Maximum Transmission Unit
MQTT	Message Queue Telemetry Transport
REST	Representational State Trasfer
Spoofing	Lähdeosoitteen huijaaminen/väärentäminen
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
XMPP	Extensible Messaging and Presence Protocol

Sisällys

1 Johdanto

Esineiden internetin laitteiden eli IoT-laitteiden (Internet of Things) määrä ja merkitys kasvaa koko ajan. Nämä laitteet voivat olla esimerkiksi jääkaappeja, ovikelloja tai biometrisiä tietoja keräämiä älykkäitä laitteita [18]. IoT-laitteiden määrän nopea kasvu ja kehitys, sekä niiden rajoittunut luonne aiheuttavat haasteita niiden tietoturvallisuudelle. Pieni virrankulutus on tärkeä ominaisuus IoT-laitteissa, sillä laitteet toimivat usein paristoilla, joiden kapasiteetti on rajallinen ja joiden vaihtaminen voi olla haastavaa. Kaikki laskenta ja langaton viestintä kuluttavat laitteen paristoa, joten niiden pitäminen mahdollisimman vähäisenä on toivottavaa. Tietoturvaominaisuuksien lisääminen ja käyttöönottaminen laitteessa kasvattaa virrankulutusta, sekä kasvaneen laskennan osalta, että lisääntyneenä viestintänä pakettien kasvaessa salatun liikenteen myötä.

Tässä tutkielmassa tarkastellaan IoT-laitteissa käytettävän sovellusprotokollan, CoAP:n (Constrained Application Protocol), tietoturvallisuutta. CoAP-protokollan tietoturvallisuutta on käsitelty myös aiemmassa tutkimuksessa. Esimerkiksi Almeghlef et al. [3] tarkastelevat CoAP:hen kohdistuvia vahvistushyökkäyksiä ja Rahman et al. [33] käsittelevät DTLS-suojatun CoAP-protokollan turvallisuusongelmia. Olemassa oleva tutkimus keskittyy kuitenkin joko yksittäisiin uhkiin ja niiden ratkaisuihin kuten [3], tai käsittelevät CoAP-protokollan tietoturvallisuutta lyhyesti, ilman testauksen tukemaa analyysia, kuten [33]. CoAP:n tietoturvaominaisuuksia tarkkaan kartoittavaa, laajasti eri uhkia käsittelevää ja analyysissään toteutetun testauksen tuloksia hyödyntävää tutkimusta ei ole toteutettu aiemmin.

Tutkielman tavoitteena on vastata seuraavaan tutkimuskysymykseen, sekä sen tueksi asetettuihin apukysymyksiin:

- Onko CoAP-protokolla tietoturvallinen?
  - Mistä muodostuu CoAP-protokollan tietoturvallisuus?
  - Mitkä ovat tärkeimmät CoAP-protokollaan kohdistuvat tietoturvauhkat?
  - Miten protokollan tietoturvallisuutta voidaan testata?

Tutkimuskysymykseen pyritään vastaamaan protokollaa ja sen tietoturvallisuutta tarkastelevan kirjallisuuskatsauksen, sekä protokollan tietoturvatestauksen tu-

loksia yhdistävän analyysin perusteella. Tietoturvatestauksessa käytettävän testausasetelman luomisessa sovelletaan konstruktiivista tutkimusmenetelmää.

Kirjallisuuskatsauksessa tarkastellaan CoAP-protokollaa aiemman tutkimuksen pohjalta. Tarkastelussa ovat protokollan toiminta, ominaisuudet, tietoturvaominaisuudet, sekä protokollaan kohdistuvaksi tunnistetut tietoturvauhkat. Konstruktiivista metodia hyödyntämällä luodaan CoAP-protokollan tietoturvallisuuden testausasetelma, joka mahdollistaa protokollan tietoturvaominaisuuksien ja siihen kohdistuvien tietoturvauhkien testaamisen.

Tietoturvatestauksessa toteutetaan spoofing ja yksinkertainen vahvistushyökkäys. Lisäksi tuotetaan ja tarkastellaan DTLS-suojattua CoAP-liikennettä. Testauksen tulokset yhdistetään kirjallisuuskatsauksen tulosten kanssa analyysivaiheessa, jossa selvitetään vastaus tutkimuskysymykseen.

Tutkielman päätuloksena on, että CoAP-protokolla on tietoturvallinen, mikäli sen turvallisuustiloja käytetään oikein uhkataso huomioiden ja avainten jakelu sekä bootstrapping-prosessi on toteutettu turvallisesti. Protokolla on altis useille eri hyökkäyksille, kuten tietoturvatestauksessa testatuille spoofingille sekä vahvistushyökkäyksille. DTLS-suojauksen käyttämättä jättäminen ilman alempien kerrosten suojausta mahdollistaa näiden hyökkäysten toteuttamisen, eikä protokolla tällaisessa tilanteessa ole tietoturvallinen.

Tutkielman rakenne koostuu tämä johdantoluku mukaan lukien seitsemästä luvusta. Luvussa 2 esitellään tutkielmassa käytetyt tutkimusmenetelmät. Luvussa 3 taustoitetaan tutkielmaa IoT:n ja sen tietoturvan osalta. Luvussa 4 tarkastellaan CoAP-protokollan ominaisuuksia, sekä sen tietoturvaominaisuuksia ja -uhkia. Luvussa 5 rakennetaan testausasetelma ja toteutetaan protokollan testaaminen. Luvussa 6 vastataan tutkimuskysymykseen kirjallisuuskatsauksen ja testauksen tulokset yhdistävän analyysin pohjalta. Luvussa 7 esitetään tutkielman yhteenveto.

2 Menetelmät ja aineisto

Tutkielman tutkimusmenetelmiä ovat kirjallisuuskatsaus ja konstruktiivinen tutkimusmenetelmä. Kirjallisuuskatsauksessa tarkastellaan CoAP-protokollan toimintaa, ominaisuuksia, tietoturvaominaisuuksia sekä protokollaan kohdistuvia tietoturvauhkia. Lisäksi tarkastellaan IoT:ia ja sen tietoturvallisuutta sekä CoAP:n sijoittumista IoT:n protokollapinossa. Konstruktiivista menetelmää sovelletaan CoAP-protokollan tietoturvallisuuden testausta varten rakennettavan testausasetelman luomisessa.

Kirjallisuuskatsauksen ja konstruktiivista menetelmää soveltaen rakennetussa testausasetelmassa toteutetun tietoturvatestauksen tulokset yhdistetään tutkielman loppuvaiheen analyysissä, jolloin saadaan vastattua tutkielman tutkimuskysymykseen. Konstruktiivisen menetelmän ja sen soveltamisen lisäksi luvussa käsitellään myös tutkielmassa käytetty aineisto.

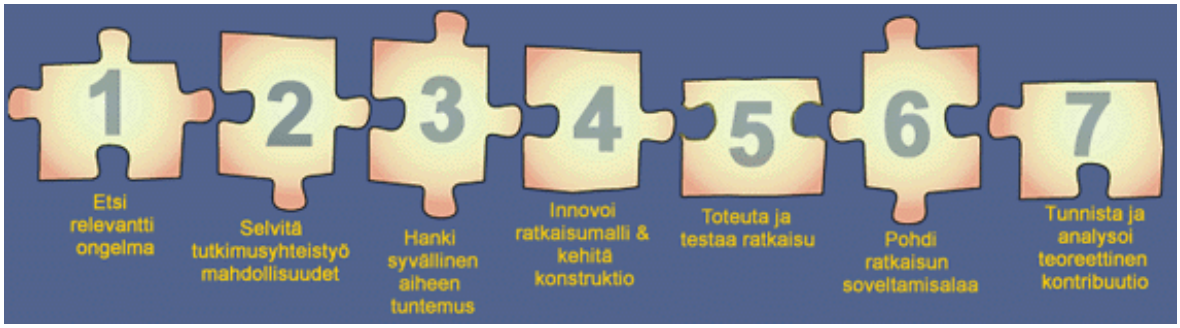
2.1 Konstruktiivinen menetelmä

Konstruktiivisessa tutkimusotteessa tuotetaan innovatiivisia konstruktioita, joilla pyritään reaalimaailman ongelmien ratkaisuun [19]. Menetelmä on kehitetty liiketaloustieteen alueella ja sitä on sovellettu laajasti, esimerkiksi tietojärjestelmätieteiden ja lääketieteen alalla.

Konstruktiivisen metodin ydinkäsitteenä on konstruktio, joka voi olla esimerkiksi malli, diagrammi, suunnitelma, organisaatiorakenne, kaupallinen tuote tai tietojärjestelmämalli [19]. Konstruktio ei ole löydetty vaan se on keksitty ja kehitetty. Esimerkkejä konstruktion kehittämisestä ovat keinotekoiset kielet, kuten Morse-aakkoset tai ohjelmointikielet, sekä lääketieteessä lääkkeiden tai hoitomuotojen kehittäminen.

2.1.1 Konstruktiivinen tutkimusprosessi

Tutkimusprosessi esitellään Lukan [19] esittämän konstruktiiviselle tutkimukselle ominaisen tutkimusprosessin mukaisesti. Lukan esittämä konstruktiivinen tutkimusprosessi on esitetty kuvassa 2.1.



Kuva 2.1: Konstruktiivisen tutkimuksen tutkimusprosessi [19]

Relevantti ongelma

Prosessin ensimmäisessä vaiheessa etsitään relevantti ongelma. Asiaa tulisi pohtia sekä käytännön, että teorian kannalta. Ideaalisti aiheella tulisi olla käytännöllistä merkitystä ja samalla sen tulisi olla paradoksaalinen tai alianalysoitu. Tutkimusaiheita voi löytyä esimerkiksi käytännön edustajien ajatuksista, toiminnasta ja vaikeiksi kokemista asioista.

Tutkimusyhteistyön mahdollisuudet

Toisessa vaiheessa selvitetään tutkimusyhteistyömahdollisuudet. Osapuolten tulee sitoutua projektiin ja merkittäviinkin panostuksiin projektissa. Tyypillisesti tutkija tulee jäseneksi ongelman ratkaisemiseksi muodostettavaan työryhmään. Tutkija voi olla myös kyseisen työryhmän vetäjä. Yksin toimiva tutkija tulee kokemuksen perusteella lähes väistämättä epäonnistumaan eikä todelliseen toteutusvaiheeseen todennäköisesti päästä.

Syvällisen tuntemuksen hankkiminen

Kolmannessa vaiheessa hankitaan syvällinen aiheen tuntemus. Tutkimusaiheen tuntemuksen hankinnassa voidaan käyttää etnografisia metodeja kuten havainnointia, haastatteluja ja kirjallisten aineistojen analysointia. Tutkija perehtyy kohdeorganisaatioon pyrkien saavuttamaan näkemyksen sen lähtötilanteesta ja paljastamaan sen eksplisiittiset sekä implisiittiset ongelmat ja tavoitteet. Lisäksi pyritään ongelma-alueen käsitteellistämiseen sekä varmistamaan tietoisuus alan aiemmista teorioista.

Ratkaisumallin innovointi ja konstruktion kehittäminen
Neljännessä vaiheessa innovoidaan ratkaisumalli ja kehitetään konstruktio. Vaihe on kriittinen, koska jos siinä ei onnistuta, ei tutkimusta voida jatkaa. Vaiheen luonne on luova ja heuristinen, eikä siihen siksi ole yleispätevää metodologista ohjeistusta. Pelkästään aiempien konstruktioiden tai ratkaisujen siirtäminen ja soveltaminen uuteen ympäristöön ei ole konstruktiivisen tutkimusotteen soveltamista.

Toteuttaminen ja testaus
Viidennessä vaiheessa toteutetaan ja testataan ratkaisu. Käytännön testaaminen on yksi tärkeimpiä konstruktiivisen tutkimuksen ominaispiirteitä, joka erottaa sen analyyttisestä mallinnuksesta, jossa konstruktiot yleensä vain rakennetaan, mutta toteuttamiskelpoisuutta ei testata. Konstruktion testaaminen ei ole pelkästään teknistä, vaan myös tutkimusprosessin toimivuutta testataan kokonaisuudessaan.

Soveltamisalan pohdinta
Kuudennessa vaiheessa pohditaan ratkaisun soveltamisalaa. Tässä vaiheessa tutkijan tulisi ottaa etäisyyttä empiiriseen työhönsä ja pohtia yhdessä kohdeorganisaation kanssa läpikäymäänsä oppimisprosessia. Prosessin tulosten ja sen ennakkoehtojen analysoiminen on tässä vaiheessa tärkeintä. Mikäli testaus on onnistunut, voidaan pohtia, kuinka laajasti ja millä muunnoksilla konstruktio voitaisiin siirtää muihin organisaatioihin. Myös testauksen epäonnistuessa tarvitaan teoreettista analyysia, jolloin voidaan analysoida esimerkiksi, voidaanko epäonnistumista välttää muissa organisaatioissa.

Kontribuution tunnistaminen ja analysointi
Seitsemännessä vaiheessa tunnistetaan ja analysoidaan teoreettinen kontribuutio. Tämä vaihe on ratkaiseva vaihe projektissa, jossa tutkijan tulisi pystyä eksplikoimaan projektin teoreettinen kontribuutio. Tämä voi tapahtua esimerkiksi havaintojen reflektoinnilla aiempaan (mahdollisesti jo olemassa olevaan) teoriaan.

2.1.2 Menetelmän soveltaminen tutkielmassa

Konstruktiivista metodia hyödynnetään tässä tutkielmassa soveltavasti yhdessä kirjallisuuskatsauksen kanssa. Edellä esitelty Lukan [19] näkökulma konstruktiivisen metodin tutkimusprosessista ei sellaisenaan toteudu tutkielmassa, sillä konstruktiivista metodia hyödynnetään vain osana tutkielmaa, eikä tutkielman tavoitteena ole

pelkkä kehitettävä konstruktio. Lisäksi esitellyn prosessin näkökulma on vahvasti organisaatiolle tehtävässä työssä ja tutkimusryhmässä. Tutkielmaa ei tehdä yhteistyössä jonkin organisaation kanssa, eikä olemassa ole työryhmää, jonka jäsen tutkielman tekijä olisi. Metodin hyödyntäminen on kuitenkin merkittävässä osassa testauksen mahdollistamisessa ja siten myös testaustulosten tuottamisessa lopun analyysiä varten.

Koska konstruktiivinen menetelmä ja sen prosessi ovat merkittävässä roolissa tutkielmassa kuvataan tässä aliluvussa kuitenkin koko tutkielman tutkimusprosessi kuvatun konstruktiivisen tutkimusprosessin avulla. Tätä varten prosessia täydennetty kolmannen vaiheen osalta kuvaamalla myös teoriaosuudessa tehtyä kirjallisuuskatsausta. Lisäksi viimeisen vaiheen osalta on kuvattu kirjallisuuskatsauksen, sekä konstruktion avulla toteutetun testausten tulosten yhdistävän analyysin tekeminen.

Relevantti ongelma

Relevanttia ongelmaa lähdettiin etsimään IoT:n, tietoturvan ja tietoturvan testaamisen yhdistelmästä. Tässä vaiheessa käytettäväksi metodiksi ei vielä ollut määritelty konstruktiivista metodia vaan menetelmä valikoitui käytettäväksi vasta tutkimusongelman määrittämisen jälkeen.

CoAP-protokolla tietoturvallisuuden arvioiminen on relevantti aihe IoT-sovellusten kehittäjille, jotka harkitsevat CoAP-protokollan käyttöä ja toteutusta sovelluksissaan. Protokollan tietoturvallisuutta on tarkasteltu aiemmissa tutkimuksissa, mutta tarkastelu on yleensä keskittynyt jonkin tietyn tietoturvaongelman arviointiin ja mahdollisesti sen ratkaisuun. Yhteen vetävää ja havainnollistavaa tutkimusta, joka käsittelisi tietoturvaa kokonaisuutena, ei ole toteutettu.

Konstruktiivinen menetelmä tarjoaa ongelman ratkaisun tueksi mahdollisuuden kehittää asetelma ja ympäristö, jossa tietoturvaa voidaan testata, testausten tulosten tarjotessa mahdollisuuden syvemmälle analyysille protokollan tietoturvallisuudesta, kuin pelkkä kirjallisuuskatsaus.

Tutkimusyhteistyön mahdollisuudet

Tutkielman lähtökohtana ei ollut tuottaa ratkaisua yksittäisen organisaation liiketoimintaongelmaan. Tästä lähtökohdasta ongelma on muotoutunut siten, ettei prosessissa tarkoitettu tutkimusyhteistyö ole tutkielman kannalta relevantti, eikä mahdollisuuksia kartoitettu tarkemmin.

Syvällisen tuntemuksen hankkiminen

Syvällinen aiheen tuntemus hankittiin teoriaosan kirjallisuuskatsauksen, sekä testausasetelman kehittämisen ja rakentamisen aikana. Kirjallisuuskatsauksen aikana selvitettiin, minkä tyyppistä testausta protokollalle on aiemmin tehty, jotta konstruktio olisi uniikki. Konstruktion kannalta kirjallisuuskatsaus vastasi siis siihen, *mitä* konstruktiota hyödyntäen tulisi testata. Kirjallisuuskatsauksen materiaaleihin perehtyessä kasvatettiin myös riittävä osaaminen toimivan konstruktion kehittämiseksi.

Ratkaisumallin innovointi ja konstruktion kehittäminen

Koska konstruktio on luonteeltaan hyvin tekninen, kuten protokollan tietoturvallisuuden tarkastelu kokonaisuudessaankin, on kirjallisuuskatsauksen osana tuotettu tutkielman teoriaosaan kattava selvitys protokollan toiminnasta, mukaan lukien alempien tasojen protokollat riittävässä tarkkuudessa. Kattava selvitys on tarpeellinen, jotta valmista konstruktiota ja sen tuloksia voidaan analysoida yhdessä teoriaosan kanssa. Lisäksi konstruktion rakentamisessa tehdyt valinnat ja yksityiskohdat ovat perusteltavissa kattavan teoriaosuuden pohjalta.

Itse konstruktion rakentamisen, eli testausasetelman ja -ympäristön luominen koostui aiempien toteutettujen testausten kartoittamisesta, soveltuvien työkalujen ja teknologioiden selvittämisestä, sekä varsinaisesta ympäristön rakentamisesta ja testaamisesta. Keskeisin vaihe oli käytettävien työkalujen ja teknologioiden valinta, sillä toimivan ympäristön ja asetelman rakentaminen vaati useiden eri ratkaisujen käyttöä, joiden tuli olla keskenään yhteensopivia. Työkalujen ja teknologioiden soveltuvuuden testaaminen vaati useisiin eri vaihtoehtoihin perehtymistä, sekä myös niiden testaamista.

Toteuttaminen ja testaus

Testausympäristön ja -asetelman rakentamiseen liittyi jo itsessään paljon testausta ja valmiin konstruktion kohdalla voitiin olla varmoja, että ainakin tekninen toteutus toimisi ja jonkinlaisia tuloksia olisi konstruktiota hyödyntämällä mahdollista saada. Toteutusvaiheessa konstruktiota hyödynnettiin suorittamalla protokollan tietoturvan testaus määritettyjen ominaisuuksien ja uhkien osalta. Konstruktio voitiin todeta onnistuneeksi, sillä testaus saatiin toteutettua halutulla tavalla ja testaus tuotti tuloksia, joita voitiin hyödyntää loppuvaiheen analyysissä.

Soveltamisalan pohdinta

Onnistuneen konstruktion testauksen myötä voidaan todeta, että konstruktio on toimiva. Konstruktio mahdollisti tutkielmassa suoritetun testauksen toteuttamisen, mutta tarjoaa myös mahdollisuuden muiden ominaisuuksien ja uhkien testaamiseen. Näiden ominaisuuksien ja uhkien testaaminen voi kuitenkin vaatia konstruktion jatkokehittämistä siten, että siihen tuodaan uusia elementtejä esimerkiksi jonkin toteutetussa konstruktiossa käyttämättömän protokollan osalta. Riippuen käyttötapauksesta voi jatkokehityksen tarve olla hyvin kevyttä tai mahdollisesti työläämpää.

Kontribuution tunnistaminen ja analysointi

Konstruktion rakentamisessa onnistuttiin tuottamaan CoAP-protokollan tietoturvan testausympäristö ja -asetelma. Vastaavaa konstruktiota ei ole tunnistettu aiemmasta tutkimuksesta. Testausympäristöä ja -asetelmaa voidaan hyödyntää sellaisenaan tai jatkokehitettävänä CoAP-protokollan tietoturvallisuuden testaamisessa.

Konstruktion avulla tuotettujen testaustulosten ja kirjallisuuskatsauksen yhdistävän analyysin avulla voitiin arvioida CoAP-protokollan tietoturvaa. CoAP-protokollan tietoturvaa kokonaisuutena ei ole vastaavasti tutkittu aiemmin.

2.2 Aineisto

IoT:sta, sen tietoturvallisuudesta ja CoAP-protokollasta löytyy paljon aiempaa tutkimusta. Tutkielman aineistona käytettiin pääasiassa aiempaa aihealueen tutkimusta edustavia tieteellisiä artikkeleita ja konferenssijulkaisuja, sekä RFC-dokumentteja. Muita käytettyjä lähteitä olivat työkalujen ja teknologioiden verkosta saatavilla olevat dokumentit, NVD:n (National Vulnerability Database) haavoittuvuustiedot, tekniset raportit/whitepaperit sekä yksi väitöskirja. Aiemman tutkimuksen osalta lähdeaineiston valinnassa pyrittiin tunnistamaan keskeisimpiä, yleisesti käytettyjä artikkeleita ja konferenssijulkaisuja.

Aiemman tutkimuksen osalta keskeisiä lähteitä olivat Almeghlefin et al. [3], Rahmanin et al. [33] ja Zamfirin et al. [46] konferenssijulkaisut sekä Al-Fuqahan et al. [1] artikkeli. Almeghlef et al. tarkastelevat vahvistushyökkäyksiä CoAP-protokollaa vastaan ja esittelevät koneoppimiseen perustuvan tunnistusjärjestelmän näille hyökkäyksille. Rahman et al. tarkastelevat DTLS-suojatun CoAP:n turvallisuuden ongel-

mia, sekä niiden ratkaisuja ja avoimia haasteita. Zamfir et al. tarkastelevat IoT:n protokollien, mukaan lukien CoAP:n, turvallisuuden haasteita. Al-Fuqaha et al. tarjoavat yleiskatsauksen IoT:sta painottaen sen mahdollistavia teknologioita, protokollia ja sovelluksia.

Keskeinen lähde oli myös RFC 7252 -dokumentti [39], joka määrittelee CoAP-protokollan. Dokumenttia käytettiin CoAP-protokollan toiminnan ja ominaisuuksien tarkassa selvityksessä, mikä oli tarpeellista konstruktion, eli protokollan tietoturvallisuuden testausasetelman luomiseksi, sekä testauksen tulosten ymmärtämiseksi. Dokumentti ottaa lisäksi kantaa myös protokollan tunnistettuihin turvallisuushaasteisiin. Myös muita RFC-dokumentteja kuten 6347 (DTLS 1.2) [36] käytettiin lähdemateriaalina tarpeellisessa laajuudessa.

Erityisenä lähdeaineistoon liittyvänä huomiona DTLS-protokolla määritellään nykyisin RFC 9147 -dokumentissa [37] (versio 1.3). Huomioitavaa on, että kyseinen dokumentti on tutkielman kirjoitushetkellä tuore (2022) ja muu lähdeaineisto on DTLS:ää käsitteleviltä osin vanhempaa, joten ne käsittelevät protokollan vanhempaa versiota 1.2, joka määritellään RFC 6347 -dokumentissa [36]. Näin ollen lähdeaineisto ei huomioi uuden version uusia ominaisuuksia ja parannuksia vanhempaan versioon, joilla voi mahdollisesti olla merkitystä sen käyttöön CoAP:n yhteydessä. Tässä tutkielmassa käsitellään siis DTLS-protokollan versiota 1.2. Version 1.3 osalta on tuotu esiin keskeiset muutokset vanhaan versioon ja analysoitu niiden vaikutusta CoAP-protokollan näkökulmasta, mutta sen toimintaa ei käsitellä yksityiskohtaisesti.

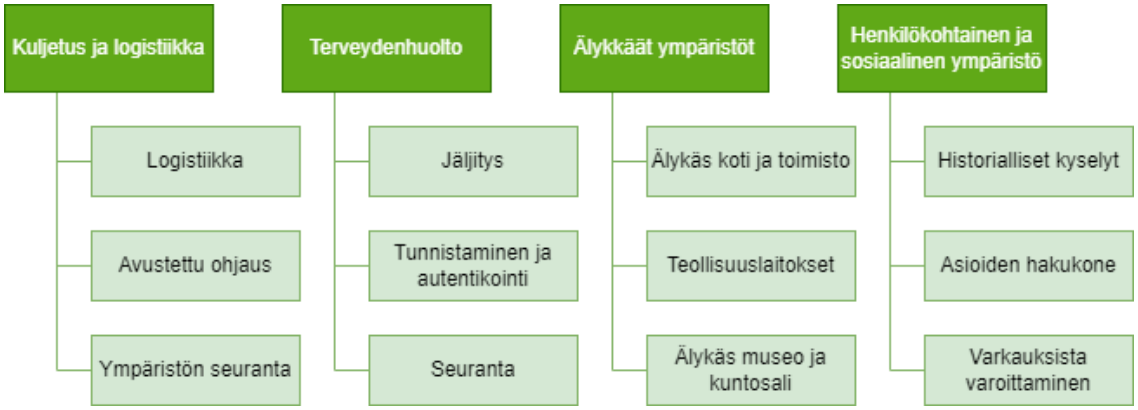
3 IoT ja tietoturva

Esineiden internet (Internet of Things, IoT) muuttaa fyysisiä objekteja ja perinteisiä laitteita älykkäiksi hyödyntämällä erilaisia teknologioita, kuten kaikkialle ulottuvaa laskentaa, sulautettuja laitteita, kommunikaatioteknologioita, sensoriverkkoja, internet-protokollaa ja sovelluksia [1]. IoT mahdollistaa näiden älykkäiden laitteiden nähdä, kuulla, ajatella sekä suorittaa tehtäviä yhdessä ”keskustelemalla” jakaakseen tietoa tai koordinoidakseen päätöksiä. Tulevaisuudessa IoT:n piirissä odotetaan olevan merkittäviä koti- ja liiketoimintasovelluksia, jotka vaikuttavat elämänlaatuun ja kasvattavat taloutta. Esimerkiksi älykodit mahdollistavat asukkailleen automaattisen autotallin ovien avaamisen kotiin tullessa, sekä automaattisen tai etänä tapahtuvan kahvin valmistuksen, ilmastoinnin, television ja muiden sovellusten ohjaamisen.

IoT-laitteiden liittäminen Internetiin nostaa kuitenkin esiin useita turvallisuushaasteita, sillä pääosaa verkon teknologioista ja protokollista ei ole suunniteltu tukemaan IoT:ia [22]. Lisäksi IoT:n kaupallistaminen on johtanut esimerkiksi yksityisyyden ongelmiin, kyberhyökkäysten uhkaan ja järjestäytyneeseen rikollisuuteen. IoT:hen liittyviä turvallisuusuhkia ja yksityisyyden ongelmia ei vielä tunneta hyvin, minkä takia niitä on tutkittava ja käsiteltävä perusteellisesti.

3.1 IoT:n sovelluskohteet

IoT-laitteille on olemassa monia eri sovelluskohteita sekä käyttöympäristöjä ja IoT-laitteiden jaottelua näihin kohteisiin sekä ympäristöihin voidaan toteuttaa useilla eri tavoilla. Tässä aliluvussa esitellään muutamia IoT:n sovelluskohteita ja käyttöympäristöjä Atzorin et al. [5] määrittelemän jaon mukaisesti. Atzorin et al. esittelemä jako neljään sovelluskohteeseen tai käyttöympäristöön on esitetty kuvassa 3.1.



Kuva 3.1: IoT:n sovelluskohteita ja käyttöympäristöjä [5]

3.1.1 Liikenne ja logistiikka

Kehittyneet autot, junat, linja-autot ja polkupyörät sisältävät yhä enemmän erilaisia sensoreita, toimilaitteita ja prosessointitehoa [5]. Myös tiet, kiskot ja kuljetettavat tavarat on varustettu tunnisteilla ja sensoreilla, jotka välittävät tärkeitä tietoja esimerkiksi liikenteenohjauspaikoille ja kuljetusajoneuvoihin liikenteen ohjaamiseksi paremmin. Lisäksi ne auttavat varikkojen hallinnassa, tarjoavat matkailijalle asianmukaista liikennetietoa ja valvovat kuljetettujen tuotteiden tilaa.

Tärkeimpiä kuljetuksen ja logistiikan sovelluksia on RFID- ja NFC-teknologioihin perustuva reaaliaikainen tiedonkäsittely jokaisessa toimitusketjun vaiheessa alkaen hyödykkeen suunnittelusta, raaka-aineiden hankintaan, tuotantoon, kuljetukseen, varastointiin, jakeluun, sekä puolituotteiden ja tuotteiden myyntiin, palautusten käsittelyyn sekä myynnin jälkeiseen palveluun [5]. Tuotteisiin liittyvää tietoa on myös mahdollista saada nopeasti, oikea-aikaisesti ja tarkasti. Tämä mahdollistaa yritysten tai jopa koko toimitusketjun vastaamisen monimutkaisiin ja muuttuviin markkinoihin mahdollisimman nopeasti.

Muita esimerkkejä tärkeimmistä sovelluksista ovat esimerkiksi avustettu ajaminen ja ympäristömuuttujien valvonta kuljetuksissa [5]. Avustetussa ajamisessa liikennevälineiden kuljettajat sekä matkustajat saavat sensoreiden ja toimilaitteiden välityksellä tärkeää tietoa, joka mahdollistaa paremman navigoinnin ja turvallisuuden esimerkiksi törmäyksenestojärjestelmien ja vaarallisten aineiden kuljetusten seurannalla. Ympäristömuuttujia, kuten lämpötilaa, kosteutta ja mahdollisia iskuja on tärkeää seurata helposti pilaantuvien tuotteiden, kuten hedelmien, tuoretuotteiden, lihan ja maidon osalta niiden säilymisen varmistamiseksi [17]. Tällainen

valvonta on mahdollista erilaisiin sensoriratkaisuihin tukeutuen.

3.1.2 Terveydenhuolto

IoT tarjoaa useita hyötyjä myös terveydenhuollon alalle ja alan sovelluksia voidaan pääsääntöisesti jakaa esineiden ja ihmisten (henkilöstö ja potilaat) jäljittämiseen, tunnistamiseen ja ihmisten autentikointiin, sekä automaattiseen tiedonkeruuseen ja seurantaan [42].

Jäljittämisen tarkoituksena on tunnistaa henkilöiden ja esineiden liikettä [42]. Jäljittäminen voi olla reaaliaikaista sijainnin seurantaa, kuten potilasvirran seuraamista työnkulun parantamiseksi sairaaloissa tai liikkumisen seurantaa erityisten pisteiden kohdalla tietyille alueille pääsemisen hallitsemiseksi. Omaisuuden osalta seurantaa käytetään yleisimmin jatkuvaan varaston sijainnin seurantaan (esimerkiksi ylläpitoon, saatavuuteen tarkastamiseen ja käytön seurantaan liittyen) sekä materiaaliseurantaan, jolla estetään sisään jäämiset leikkauksissa esimerkiksi näytteiden ja verituotteiden osalta.

Tunnistamisella ja autentikoinnilla voidaan välttää esimerkiksi vääriä lääkkeitä, annoksia, aikoja ja toimenpiteitä [42]. Lisäksi se mahdollistaa kattavat ja ajantasaiset potilastiedot sekä vauvojen tunnistamisen sairaaloissa sekoittumisen välttämiseksi. Henkilöstön osalta tunnistusta ja autentikointia käytetään useimmiten pääsynhallinnassa. Omaisuuden osalta niitä voidaan käyttää esimerkiksi turvatoimien vaatimusten täyttämiseen, varkauksien välttämiseen sekä tärkeiden tuotteiden ja instrumenttien katoamisen estämiseen.

Sensorilaitteita hyödyntävä seuranta mahdollistaa potilaskeskeisen toiminnan, potilaan sairauksien diagnosoinnin ja tarjoaa reaaliaikaista tietoa potilaan terveydentilasta [5]. Sovellusalueita ovat erilaiset telelääketieteen ratkaisut, potilaan lääkemääräysten noudattamisen seuranta ja varoitukset potilaan hyvinvointiin liittyen. Näillä sovellusalueilla sensoreita voidaan soveltaa sekä laitos-, että kotihoidossa. Heterogeenisiä langattomiin teknologioihin perustuvia potilasvalvontajärjestelmiä voidaan käyttää potilaan seurantaan kaikkialla [25]. Näissä järjestelmissä integroidaan useita langattomia teknologioita biosignaalien keskeytymättömän seurannan varmistamiseksi myös potilaiden ollessa liikkeellä.

3.1.3 Älykkäät ympäristöt

Koteihin ja toimistoille sijoitettavat sensorit ja toimilaitteet voivat tehdä elämästä mukavampaa monella tapaa [5]. Lämmitystä voidaan säätää mieltymysten ja sään mukaisesti, huoneiden valaistus voi muuttua vuorokaudenajan mukaan, kotien onnettomuuksia voidaan välttää asianmukaisilla valvonta- sekä hälytysjärjestelmillä ja energiaa voidaan säästää sammuttamalla sähkölaitteita automaattisesti, kun niitä ei tarvita. Energian kulutusta voidaan seurata ja automaatiolla voidaan optimoida virrankulutusta koko päivän ajan tarkkailemalla, milloin sähkön pörssihinnat ovat halpoja [7]. Lisäksi voidaan ottaa huomioon kunkin kodin laitteen, kuten esimerkiksi akkulaturin, jääkaapin sekä uunin erityisvaatimukset.

Älykkäät ympäristöt auttavat myös parantamaan teollisuuslaitosten automaatiota esimerkiksi tuotettaviin osiin liitettävien RFID-tunnisteiden massiivisella käyttöönotolla [5]. Spiess et al. kuvaavat konferenssijulkaisussaan [40] yleisen skenaarion, jossa RFID-lukija lukee tunnisteen, kun tuotettavat osat saavuttavat käsittelypisteen. Lukija generoi tapahtuman, joka sisältää kaikki tarvittavat tiedot, kuten RFID-numeron, ja tallentaa sen verkkoon. Kone/robotti saa ilmoituksen tästä tapahtumasta ja noutaa tuotettavan osan. Yhdistämällä yrityksen järjestelmän ja RFID-tunnisteen tietoja se tietää, miten osaa tulee käsitellä jatkossa. Samanaikaisesti koneeseen asennettu langaton sensori tarkkailee tärinää ja jos se ylittää tietyn kynnyksen, nostetaan tapahtuma prosessin välittömästi pysäyttämiseksi. Kun tällainen hätätapahtuma levitetään, sen vastaanottavat laitteet reagoivat siihen. Robotti vastaanottaa hätäpysäytystapahtuman ja lopettaa toimintansa välittömästi. Tehtaan päällikkö näkee myös välittömästi toiminnanohjausjärjestelmästä (ERP, Enterprise Resource Planning) tilausten tilan, tuotannon edistymisen, laitteen tilan sekä globaalin näkymän kaikista elementeistä ja mahdollisista sivuvaikutuksista, jotka johtuvat tuotantolinjan viivästymisestä laitteiden toimintahäiriöiden vuoksi.

Älykäs museo ja kuntosali ovat kaksi edustavaa esimerkkiä älykkäistä vapaa-ajan ympäristöistä, joissa IoT-teknologiat voivat auttaa tilojen hyödyntämisessä parhaalla mahdollisella tavalla [5]. Esimerkiksi museossa rakennuksen näyttelyt voivat tuoda esille erilaisia historiallisia ajanjaksoja (esimerkiksi Egyptin kausi tai jääkausi), joissa ilmasto-olot vaihtelevat suuresti. Rakennus mukautuu paikallisesti näihin olosuhteisiin huomioiden myös olosuhteet ulkona. Kuntosalilla personal trainer voi ladata harjoitusprofiilin harjoituslaitteisiin jokaiselle harjoittelijalle, jotka laite sitten tunnistaa automaattisesti RFID-tunnisteiden avulla. Terveysparametreja seurataan koko harjoituksen ajan ja raportoiduista arvoista tarkistetaan, onko harjoitte-

lija ylikunnossa tai onko hän liian rentoutunut harjoituksia tehdessään.

3.1.4 Henkilökohtainen ja sosiaalinen ympäristö

Henkilökohtaiseen ja sosiaaliseen ympäristöön kuuluvat sovellukset, joiden avulla käyttäjä voi olla vuorovaikutuksessa muiden ihmisten kanssa sosiaalisten suhteiden ylläpitämiseksi ja rakentamiseksi [5]. Esimerkkejä sovelluksista voivat olla asioiden hakukone, varkaudesta varoittaminen ja historialliset kyselyt.

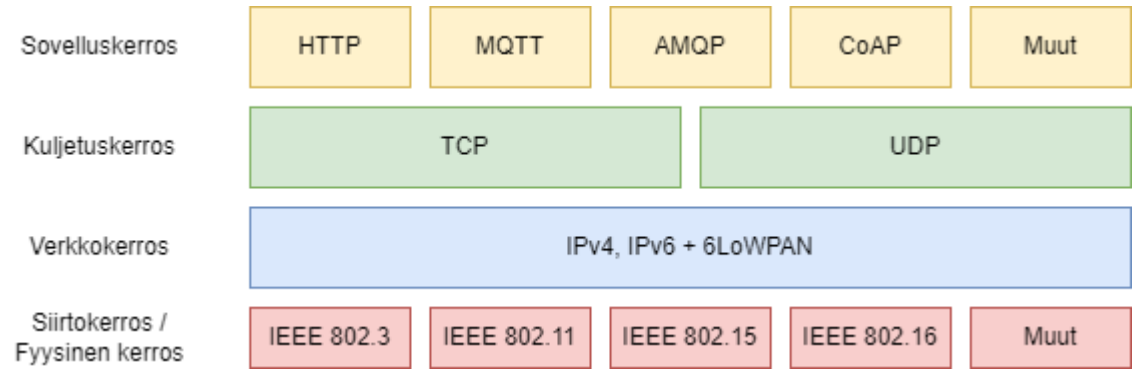
Asioiden hakukone on työkalu, joka auttaa löytämään esineitä, joista emme muista, mihin olemme ne jättäneet [5]. Yksinkertaisen verkkopohjaisen RFID-sovelluksen avulla käyttäjät voivat tarkastella merkittyjen (tagged) kohteidensa viimeisintä tallennettua sijaintia tai etsiä tietyn kohteen sijaintia. Tämän sovelluksen ennakoivampi laajennus hyödyntää käyttäjän määrittämiä tapahtumia ilmoittaakseen käyttäjille, kun viimeisin tallennettu objektin sijainti vastaa joitain tiettyjä ehtoja.

Edellisen kaltainen sovellus voi antaa käyttäjälle tiedon siitä, onko joitain esineitä siirretty rajoitetulta alueelta (omistajan asunto tai toimisto), mikä osoittaisi, että esine on varastettu [5]. Tässä tapauksessa tapahtumasta on ilmoitettava välittömästi omistajalle ja/tai vartijoille. Sovellus voi esimerkiksi lähettää käyttäjille tekstiviestin, kun varastettu esine kuten kannettava tietokone, lompakko tai koriste, poistuu rakennuksesta ilman lupaa.

Esineisiin ja tapahtumatietoihin kohdistuvilla historiallisilla kyselyillä käyttäjät voivat tutkia toimintansa trendejä ajan mittaan [5]. Tämä voi olla erittäin hyödyllistä sovelluksissa, jotka liittyvät pitkäaikaiseen toimintaan kuten yritysprojekteihin ja yhteistyöhön. Esimerkkinä voitaisiin rakentaa digitaalinen päiväkirjasovellus, joka tallentaa ja näyttää tapahtumia esimerkiksi Google-kalenterissa myöhempää tarkastelua varten. Tällä tavalla käyttäjät voivat katsoa päiväkirjojaan nähdäkseen, miten ja kenen kanssa he ovat viettäneet aikaansa.

3.2 IoT:n protokollapino

IoT-järjestelmien protokollapinossa on useita samoja protokollia kuin yleisesti käytetyssä TCP/IP-protokollapinossa, mutta näihin järjestelmiin liittyy myös useita vain niille ominaisia protokollia ja teknologioita. Nämä protokollat ja teknologiat pyrkivät huomioimaan IoT-järjestelmien vaatimuksia keveyden ja virrankulutuksen suhteen. IoT-järjestelmien protokollapino on esitetty kuvassa 3.2.



Kuva 3.2: IoT-järjestelmien protokollapino [23]

3.2.1 Sovelluskerros

Sovelluskerros on suorassa vuorovaikutuksessa käyttäjän kanssa ja koostuu sovelluksista, joilla kullakin on omat sovelluskerroksen protokollansa [45]. Se vastaa palveluiden tarjoamisesta ja määrittää joukon protokollia viestien välittämiseksi sovellusten tasolla.

Perinteiset sovellusprotokollat soveltuvat huonosti IoT-laitteille niiden rajoittuneen luonteen vuoksi. Näiden tilalle on kehitelty kevyempiä protokollia, joita ovat CoAP:n lisäksi esimerkiksi MQTT (Message Queue Telemetry Transport), AMQP (Advanced Message Queuing Protocol) XMPP (Extensible Messaging and Presence Protocol) ja DDS (Data Distribution Service) [1].

IoT-laitteet sijoittuvat erilaisiin ympäristöihin ja käyttötapauksiin, joissa on käytössä erilaisia sovelluksia ja eriäviä laskentatehoja laitteiden välillä [45]. Sovellusprotokollan valitseminen riippuu näistä ympäristöistä ja käyttötapauksista. Eri sovelluksissa käytetään siis erilaisia sovellusprotokollia. Naik on artikkelissaan [23] vertaillut neljää laajasti hyväksyttyä ja nousevaa IoT-protokollaa. Näitä protokollia ja niiden keskeisiä eroavaisuuksia on esitetty taulukossa 3.1.

Sovellusprotokollien toiminnan periaate voi perustua julkaisija/tilaaja tai pyyntö/vastaus -malleihin. Kuten taulukosta 3.1 voidaan nähdä, tukee osa protokollista, kuten esimerkiksi CoAP, näitä molempia. Julkaisija/tilaaja -mallissa asiakas julkaisee viestejä välittäjälle (broker) ja muut asiakkaat voivat tilata näitä julkaisuja tai ne voidaan säilyttää tulevia tilauksia varten [23]. Pyyntö/vastaus -mallissa puolestaan asiakas tekee pyynnön palvelimelle ja palvelin vastaa tähän pyyntöön.

Sovellusprotokollien välillä on myös eroja niiden raskaudessa. Eroavaisuudet voivat johtua esimerkiksi taulukossa 3.1 vertailluista otsikkojen koosta, kuljetusker-

Taulukko 3.1: IoT-laitteiden sovellusprotokollia [23]

Ominaisuus	**CoAP**	**MQTT**	**AMQP**	**HTTP**
Julkaisija/tilaaja	x	x	x	
Pyyntö/vastaus	x		x	x
Otsikon koko	4 tavua	2 tavua	8 tavua	Ei määritetty
QoS	QoS 0/1/2	CON/NON	Settle/Unsettle Format	Kuljetuskerros (TCP)
Kuljetuskerros	UDP	TCP	TCP	TCP
Salaus	DTLS, IPSec	TLS/SSL	TLS/SSL, IPSec, SASL	TLS/SSL
Koodaus	Binääri	Binääri	Binääri	Teksti
Portti	5683/5684	1883/8883	5671/5672	80/443

roksen protokollasta tai palvelun laadun (QoS, Quality of Service) toteutustavasta.

Salauksen ja turvallisuuden osalta protokollat eroavat siten, että TCP:tä kuljetuskerroksella käyttävät protokollat tukevat TLS/SSL-salausta (Transport Layer Security, Secure Sockets Layer). AMQP tukee lisäksi IPSec:iä, joka toimii verkkokerroksella, sekä SASL:ia (Simple Authentication and Secure Layer). Koska CoAP:ssa käyttää kuljetuskerroksella UDP:ta, se tukee SSL/TLS:n sijasta DTLS-salausta (Datagram Transport Layer Security), sekä verkkokerroksella IPSec:iä.

3.2.2 Muut kerrokset

Kuljetuskerroksella IoT-laitteissa voidaan käyttää TCP:tä, kuten esimerkiksi taulukossa 3.1 käsitellyissä HTTP:ssä, MQTT:ssä ja AMQP:ssa, tai UDP:ta, kuten CoAP:n tapauksessa. TCP:n suurimpia etuja on luotettava pakettien toimitus [23]. UDP ei tarjoa tätä luotettavuutta, jolloin pakettien toimittamisen varmistaminen täytyy tarvittaessa toteuttaa muulla tavalla. Toisaalta UDP:n etuna on sen keveys yhteydenmuodostuksen puuttuessa, mikä on edullista IoT-laitteiden yhteydessä.

Verkkokerroksella voidaan käyttää IP-protokollan versioita IPv4 ja IPv6, sekä 6LoWPAN:ia (IPv6 over Low power Wireless Personal Area Networks). IPv4:n ongelmana on osoiteavaruuden rajallisuus, joka on ongelma IoT-laitteiden suuren määrän ja sen kasvun takia. IPv6:ssa osoitteiden määrä ei ole ongelma. Rajoittunei-

den IoT-laitteiden, kuten vähävirtaisten sensoreiden näkökulmasta IP-protokollan kehysten suuri otsikon koko on ongelmallinen. Tähän osaltaan tarjoaa ratkaisua 6LoWPAN, joka toimii sovituskerroksena IPv6-paketeille sovittaen ne IEEE 802.15.4 määrittelyyn, jolloin ne sopivat paremmin vähävirtaisiin langattomiin verkkoihin [1].

Siirtokerroksella ja fyysisellä kerroksella voidaan käyttää useita eri teknologioita kuten Ethernet (IEEE 802.3), WLAN (IEEE 802.11), Bluetooth (IEEE 802.15) ja WiMAX (IEEE 802.16). IEEE 802.15.4 on standardi, joka määrittelee sekä fyysisen kerroksen ja median pääsynhallinnan vähävirtaisille langattomille verkoille [1]. Teknisten ominaisuuksiensa, kuten alhaisen virrankulutuksen, tiedonsiirtonopeuden ja kustannusten, sekä suuren viestinnällisen suorituskyvyn ansiosta sitä käytetään IoT-, M2M- (Machine-to-Machine) ja WSN-sovellutuksissa (Wireless Sensor Networks). Standardiin perustuu esimerkiksi ZigBee-protokolla.

3.3 IoT:n tietoturvallisuus

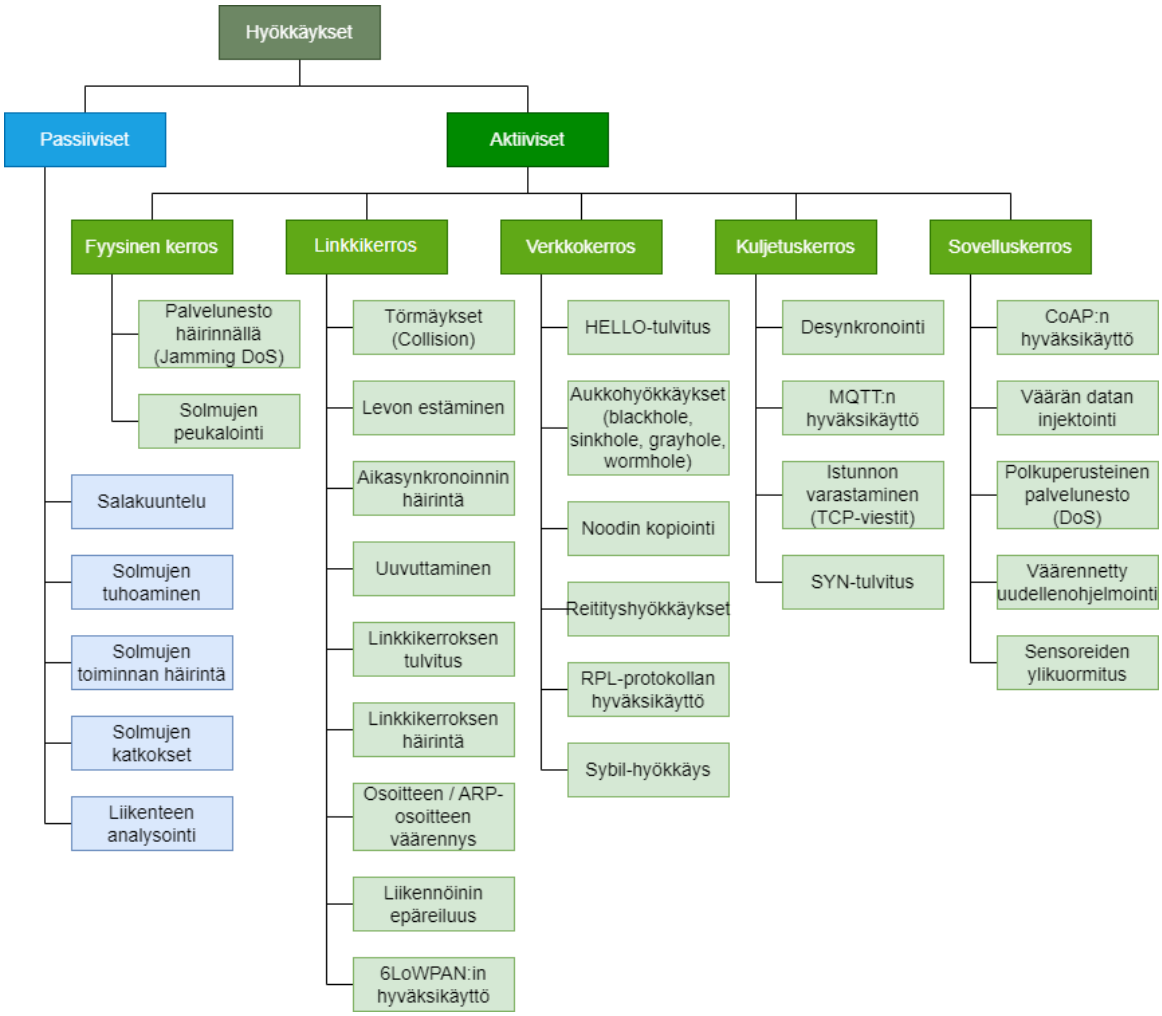
Tietoturvallisuus on merkittävä haaste IoT-toteutuksissa, IoT:n turvallisuuden puuttuvien yhteisten standardien ja arkkitehtuurien vuoksi [1]. IoT:n ydintoiminnallisuus perustuu tiedonvaihtoon miljoonien ja biljoonien internetyhteydellisten laitteiden välillä. Heterogeenisissä verkoissa, kuten IoT:n tapauksessa, käyttäjien turvallisuuden ja yksityisyyden takaaminen ei ole helppoa.

Yksi avoin ongelma laitteiden tietoturvassa on esimerkiksi avainten jakelu [1]. Toisaalta yksityisyyden ongelmat ja pääsynhallinta IoT-laitteiden välillä ilman häiriöitä ovat myös erittäin kriittisiä. Tiedonvaihdon turvaaminen on kuitenkin välttämätöntä yksityisyyden menettämisen tai vaarantumisen välttämiseksi. Ympärillämme lisääntynyt älykkäiden laitteiden määrä, jotka sisältävät arkaluontoisia tietoja, edellyttää läpinäkyvää ja helppoa pääsynhallintaa siten, että esimerkiksi yksi toimittaja voi vain lukea tietoja, kun taas toinen saa ohjata laitetta.

IoT-laitteet ovat suurimman osan ajasta valvomattomia, jolloin fyysinen hyökkääminen on helppoa fyysisesti hyökätä niihin [5]. Suurin osa yhteyksistä on langattomia, mikä tekee salakuuntelusta yksinkertaista, mikäli viestiliikennettä ei salata. Keskeinen tekijä on myös IoT-laitteiden rajoittuneisuus energian saatavuuden ja laskentaresurssien osalta, mikä korostuu erityisesti passiivisissa komponenteissa. Tästä syystä IoT-laitteissa ei yleensä pystytä toteuttamaan monimutkaisia turvallisuutta tukevia järjestelmiä.

IoT-laitteet ovat haavoittuvaisia hyökkäyksille ja erilaisille uhille useista eri syistä [5]. Suurimmat uhat liittyvät autentikointiin ja tietojen eheyteen. Autentikointi on vaikeaa, koska se vaatii yleensä asianmukaisia autentikointi-infrastruktuureja ja -palvelimia, jotka vaativat asianmukaisten viestien vaihtoa muiden solmujen kanssa. IoT:ssa tällaiset lähestymistavat eivät yleensä ole mahdollisia, koska esimerkiksi passiiviset RFID-tunnisteet eivät voi vaihtaa useita viestejä autentikointipalvelimien kanssa. Sama ongelma koskee myös erilaisia sensorinoodeja, mutta vähemmissä määrin.

Butun et al. [9] listaavat erilaisia IoT- ja WSN-järjestelmiin (Wireless Sensor Networks) kohdistuvia hyökkäyksiä jaotellen ne passiivisiksi tai aktiivisiksi. Näitä hyökkäyksiä on esitetty kuvassa 3.3.

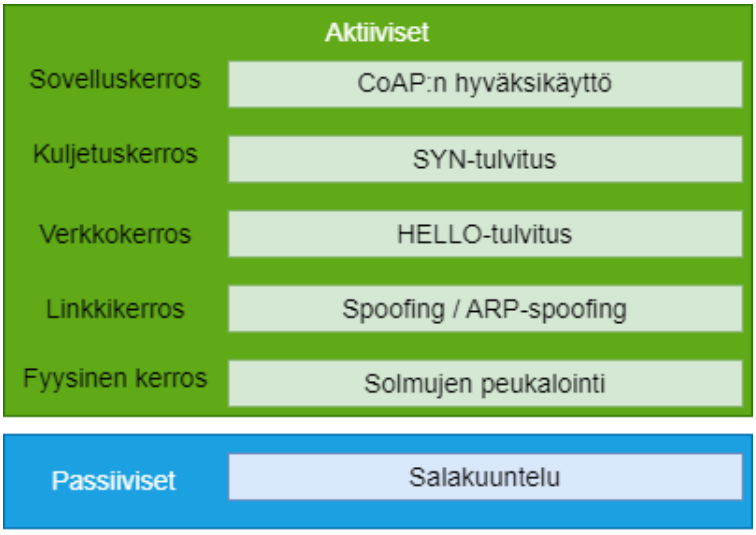


Kuva 3.3: Hyökkäyksiä IoT-järjestelmiä vastaan [9]

Passiivisia hyökkäyksiä ei voida mitata tai tunnistaa, sillä hyökkääjät eivät tuota radiosäteilyä tai liikennettä verkossa [9]. Langattomat verkot ovat alttiimpia passiivisille hyökkäyksille, sillä niiden kuuntelu on helpompaa. Passiiviset uhkat ovat pääsääntöisesti tietojen luottamuksellisuuteen liittyviä.

Aktiivisissa hyökkäyksissä hyökkääjä puolestaan tuottaa radiolähetteen tai tekee jonkin muun toimenpiteen, joka on havaittavissa [9]. Aktiiviset hyökkäykset kohdistuvat luottamuksellisuuden lisäksi myös eheyteen ja ne voivat myös tähdätä luvattomaan resursseihin pääsyyn tai niiden käyttöön viestinnän häiritsemiseksi.

Kuvasta 3.3 voidaan huomata, että IoT:hen voidaan kohdistaa useita erilaisia hyökkäyksiä, kaikilla eri OSI-mallin kerroksilla. Kaikkia hyökkäyksiä ei käydä tämän tutkielman puitteissa läpi, mutta kuvan 3.4 mukaisesti kultakin kerrokselta käydään läpi yksittäinen esimerkki kontekstin luomiseksi sovelluskerrokselle sijoittuvan CoAP-protokollan tietoturvallisuuden tarkastelua varten.



Kuva 3.4: Hyökkäysesimerkit kerroksittain

Salakuuntelu (passiiviset hyökkäykset)

Langattomat verkot, joita IoT-laitteissa useimmiten hyödynnetään ovat herkempiä passiivisille hyökkäyksille, kuten salakuuntelulle [9]. Viestinnän salakuuntelu ja viestien sieppaaminen saattaa paljastaa hyökkääjälle useita hyödyllisiä tietoja. Näitä tietoja voivat olla esimerkiksi tiettyjen solmujen sijainnit (verkossa ja fyysisesti). Kiinnostavia solmuja voivat olla avainten jaossa käytetyt solmut. Lisäksi viesteistä voidaan saada tietoon esimerkiksi niiden ID-numero, aikaleima, muita kenttiä ja oi-

keastaan kaikki, mikä ei ole salattua.

Solmujen peukalointi (fyysinen kerros)
Solmujen peukaloinnissa hyökkääjä ottaa laitteen haltuunsa esimerkiksi liittämällä sen piirilevyyn kaapeleita ja lukien sen dataa tai meneillään olevaa liikennettä [8]. Muistin sisältö vaarantuu hyökkääjän päästessä siihen käsiksi.

Lisäksi peukaloimalla voidaan muuttaa piirin alkuperäistä johdotusta tai muuttaa muistin sisältöä ja käyttää kaapattua laitetta hyökkääjän haluamalla tavalla [8]. Tämä mahdollistaa sen hyödyntämisen muissa hyökkäyksissä, kuten esimerkiksi palvelunestohyökkäyksissä tai väärän datan levittämisessä (integriteetin vaarantuminen).

Spoofing / ARP-spoofing (linkkikerros)
Termille ”spoofing” ei ole vakiintunutta suomenkielistä käännöstä. Tässä tutkielmassa spoofingilla tarkoitetaan osoitteen (esimerkiksi IP- tai MAC-osoite) huijaamista tai väärentämistä.

Linkkikerroksen spoofingissa hyökkääjän hallitsema laite väärentää itselleen kohdelaitteen MAC-osoitteen ja esiintyy tällä MAC-osoitteella muualla verkossa [9]. ARP-spoofingissa hyökkääjä lähettää verkkoon väärennettyjä ARP-viestejä (Address Resolution Protocol). Näiden viestien tavoitteena on yleensä liittää hyökkääjän MAC-osoite korkeamman tason solmun, kuten oletusyhdyskäytävän, IP-osoitteeseen, jolloin kaikki tälle IP-osoitteelle tarkoitettu liikenne lähetettäisiin hyökkääjälle.

HELLO-tulvitus (verkkokerros)
HELLO-tulvituksessa (flooding) hyökkääjä lähettää HELLO-mainosviestejä koko verkkoon korkeammalla lähetysteholla kuin tavalliset solmut, kertoen sijaitsevansa niiden lähellä [9]. Hyökkääjän laite ei kuitenkaan välttämättä ole tavallisten laitteiden kuuluvuusalueella ja kohdelaitteiden vastatessa mainosviesteihin, ei kukaan ota niitä vastaan. Näin hyökkääjä saa useita eri laitteita vastaamaan HELLO-viestiinsä, eikä sen tarvitse reagoida vastauksiin, mikäli se on kantavuusalueen ulkopuolella.

SYN-tulvitus (kuljetuskerros)
SYN-tulvituksessa hyökkääjä pyrkii kuluttamaan solmun energiaa ja/tai muistia täyttämällä sen väärillä SYN-viesteillä [9]. Tämä saavutetaan lähettämällä useita TCP-protokollan SYN-pyyntöjä kohteelle, jolloin kohde joutuu varaamaan resurs-

seja yhteyden tilan ylläpitämiseksi [44]. Kohdelaite voi myös rajoittaa yhteyksien määrää, mutta tämä vaikuttaa myös asiallisiin yhteyspyyntöihin.

CoAP:n hyväksikäyttö (sovelluskerros)
CoAP-protokolla on käytössä useissa IoT-sovellutuksissa ja sen turvallisuudessa on mainittu olevan useita haasteita [9]. Protokollan turvallisuuspuutteita voidaan hyväksikäyttää hyökkäyksessä sitä hyödyntäviä IoT-laiteita tai järjestelmiä kohtaan.

4 CoAP-protokolla

CoAP-protokolla (Constrained Application Protocol) on erityisesti rajoittuneille noodeille sekä verkoille suunniteltu sovelluskerroksen protokolla [39]. Sovellusprotokollat vastaavat tiedon välittämisestä sovellusten välillä sovelluskerroksella.

CoAP on yksinkertaistettu ja optimoitu versio HTTP-protokollasta (Hypertext Transfer Protocol), jonka toiminnallisuuksia on muokattu sopimaan paremmin rajoitettuun IoT-ympäristöön [34]. Näitä ominaisuuksia on esimerkiksi tuki multicast-lähetykselle, erityisen pieni kustannus (overhead) ja yksinkertaisuus. CoAP on myös yhteensopiva HTTP-protokollan kanssa välityspalvelimen (proxy) avulla ja sen pyynnöt ovatkin hyvin samantapaisia HTTP:n vastaavat. Protokollan keskeiset ominaisuudet ovat sen määrittelevän RFC 7252 -dokumentin [39] mukaisesti:

- Web-protokolla, joka täyttää M2M-vaatimukset rajoitetuissa ympäristöissä
- UDP:n käyttäminen valinnaisella luotettavuudella, sekä tuki unicast- ja multicast-pyynnöille
- Asynkroninen viestien vaihto
- Pieni otsikon koko ja jäsennyksen (parsing) yksinkertaisuus
- Tuki URI:lle (Uniform Resource Identifier) ja sisältötyypeille
- Yksinkertaiset välityspalvelin- ja välimuistiominaisuudet
- Tilaton HTTP-kartoitus välityspalvelinten rakentamisen mahdollistamiseksi, joilla voidaan tarjota pääsy CoAP-resursseihin HTTP:n kautta yhtenäisellä tavalla tai vaihtoehtoisesti yksinkertaisten HTTP-rajapintojen toteuttamiseksi CoAP:n kautta.
- DTLS:n (Datagram Transport Layer Security) käyttäminen suojauksen tuottamiseksi

4.1 CoAP:n toiminta

CoAP:n vuorovaikutusmalli on samanlainen kuin HTTP:n asiakas/palvelin -mallissa [39]. Koneiden välinen vuorovaikutus johtaa kuitenkin yleensä CoAP-toteutukseen, joka toimii sekä asiakas- että palvelinrooleissa. Viestinnässä asiakas lähettää HTTP-pyyntöä vastaavan CoAP-pyynnön palvelimelle pyytääkseen toimintoa palvelimella olevalle resurssille. Pyynnössä käytetään halutun toiminnon määrittämiseksi metodikoodia (Method Code) ja resurssin määrittämisessä URI:a (Uniform Resource Identifier). Palvelimen vastauksessa on vastauskoodi (Response Code) ja se voi sisältää myös esityksen kyseisestä resurssista.

Toisin kuin HTTP, CoAP käsittelee tätä viestinvaihtoa asynkronisesti datagrammityyppisen viestin kuljetuksen, kuten UDP:n kautta [39]. Tämä toteutetaan käyttämällä viestikerrosta, joka tukee valinnaista luotettavuutta (eksponentiaalisella odotusajalla yritysten välissä) viestityypin mukaisesti. CoAP:ssa on määritelty neljä viestityyppiä: vahvistettava (Confirmable, CON), ei-vahvistettava (Non-confirmable, NON), kuittaus (Acknowledgement, ACK) ja nollaus (Reset, RST).

4.1.1 CoAP-viesti

CoAP:ssa on käytössä kiinteä nelitavuinen otsikko, joka koostuu pakollisista sekä valinnaisista osista [39]. CoAP-viestin rakenne on esitetty kuvassa 4.1. Kuvassa on merkitty sinisellä viestin pakolliset osat, jotka täytyy olla jokaisessa viestissä ja valkoisella osat, jotka ovat valinnaisia.

Kuva 4.1: CoAP-viestin rakenne[39]

Versio (Ver)
Versio kertoo CoAP:n versionumeron ja on 2-bittinen etumerkitön kokonaisluku

[39]. RFC 7252 -määrittelyn mukaisissa sovellutuksissa käytetään arvoa 1 (binäärisenä 01). Tuntemattoman versionumeron omaavat viesti hylätään. Tulevaisuudessa voidaan käyttää myös muita numeroita, mikäli uusia versioita kehitetään.

Tyyppi (T)
Tyyppi on 2-bittinen etumerkitön kokonaisluku, joka määrittää viestin tyypin, joka voi olla Confirmable (0), Non-Confirmable (1), Acknowledgement (2) tai Reset (3) [39].

Tokenin pituus (TKL)
Tokenin pituus on 4-bittinen etumerkitön kokonaisluku, joka määrittää vaihtelevan koon Token-kentälle [39]. Arvo voi olla 0-8 tavua. Pituudet 9-15 ovat varattuja, eikä niitä tule käyttää.

Koodi
Koodi on 8-bittinen etumerkitön kokonaisluku, joka on jaettu 3-bittiseen luokkaan (suurimmat merkitsevät bitit) ja 5-bittiseen yksityiskohtaan (vähiten merkitsevät bitit) [39]. Koodin luokan arvoja voivat olla:

- 0: Pyyntö (request)
- 2: Onnistui -vastaus (success response)
- 4: Asiakasvirhe -vastaus (client error response)
- 5: Palvelinvirhe -vastaus (server error response)

Muut luokkien arvot ovat varattuja. Pyynnön tapauksessa koodi-kenttä osoittaa pyynnön metodin ja vastauksen tapauksessa vastauskoodin. RFC 7252 -mukaiset pyyntökoodit on esitetty taulukossa 4.1. Vastauskoodit on esitetty liitteessä A.

Taulukko 4.1: CoAP-viestin pyyntökoodit [39]

0.01	GET
0.02	POST
0.03	PUT
0.04	DELETE

Viestin ID

Viestin ID-tunnus (Message ID) on 16-bittinen etumerkitön kokonaisluku, joka on Big endian -muodossa [39]. Tunnusta käytetään havaitsemaan viestien duplikaatteja sekä yhdistämään yhteensovittamaan ACK- ja RST-viestit vastaavien CON- ja NON-viestien kanssa.

Token

Token-kenttä on valinnainen ja voi olla pituudeltaan 0-8 bittiä TKL-kentän mukaisesti [39]. Kenttää käytetään pyyntöjen sekä vastausten korrelointiin ja sen tarkoitus on erotella samanaikaisia pyyntöjä asiakkaalle. Kaikki pyynnöt sisältävät asiakkaan generoiman tokenin, joka palvelimen täytyy sisällyttää muuttamattomana kaikkiin vastauksiin. Arvo voi olla myös nolla, jos sen käytölle ei ole tarvetta. Turvallisuuden kannalta token-kentässä tulisi käyttää ei-triviaalia satunnaista arvoa, jolloin palvelimen vastausten väärentäminen olisi hyökkääjälle vaikeampaa.

Valinnat

Valinnat-kenttä (Options) voi olla tyhjä tai sisältää yhden tai useampia valintoja. Valinnat sisältävät valintanumeron (Option Number), valinta-arvon (Option Value) pituuden sekä valinta-arvon [39].

CoAP:ssa on määritelty useita erilaisia valintoja ja niillä voidaan määritellä esimerkiksi pyynnössä haluttu resurssi ja sen polku (Uri-Path). Valinnat esiintyvät valintanumeron mukaisessa järjestyksessä ja valintanumerot esitetään muutoksena edelliseen valintanumeroon. Keskeisimmät mahdolliset valinnat on esitetty taulukossa 4.2. Kaikki protokollalle RFC 7252 -dokumentissa [39] määritellyt valinnat on esitetty liitteessä A.

Hyötykuorma

Hyötykuorma on valinnainen ja jos viesti sisältää hyötykuorman, jonka pituus on nollaa suurempi, sen etuliitteenä on kiinteä, yksitavuinen hyötykuormamerkki (heksadesimaalimerkintänä 0xFF)[39]. Merkintä kertoo valintojen päättymisestä ja hyötykuorman sisältävän viestin osan alkamisesta. Hyötykuorma jatkuu merkistä UDP-datagrammin loppuun asti, joten sen koko lasketaan datagrammin koosta. Merkin puuttuminen kertoo, että hyötykuorman pituus on nolla.

Taulukko 4.2: CoAP-viestin keskeiset valinnat [39]

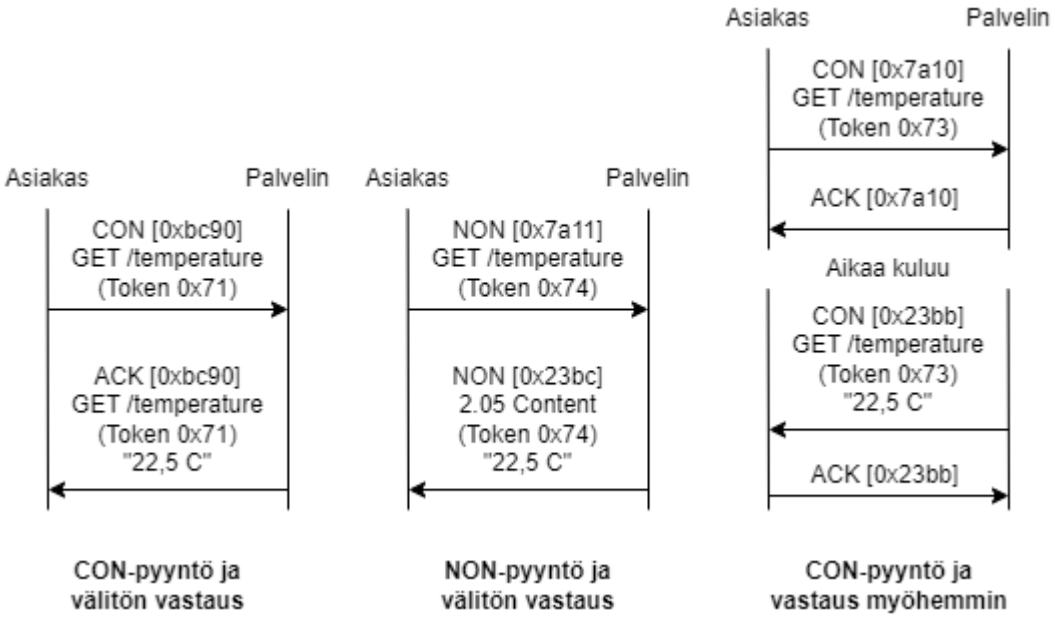
Numero	Valinta	Selite
3	Uri-Host	Resurssin isännän määrittäminen (Internet)
7	Uri-Port	Resurssin porttinumero kuljetuskerroksella
11	Uri-Path	Yksi segmentti resurssin absoluuttisesta polusta (kohde)
12	Content-Format	Hyötykuorman esitystavan määrittäminen
15	Uri-Query	Yksi argumentti resurssin parametrien asettamiseksi (kohde)

4.1.2 Viestintämalli

CoAP:n viestintämalli perustuu viestinvaihtoon päätelaitteiden välillä UDP:n yli [39]. Luotettavuus tuotetaan merkitsemällä viesti vahvistettavaksi (CON). Tällöin käytetään oletusaikakatkaisua ja eksponentiaalista odotusaikaa (back-off) uudelleenlähetysten välillä, kunnes vastaanottaja lähettää kuittauksen (ACK) samalla viestin ID:llä. Jos luotettavuutta ei tarvita, voidaan viesti lähettää vahvistamattomana (NON), jolloin viestiä ei kuitata. NON-viesti voi saada vastauksena nollausviestin (RST), mikäli vastaanottaja ei pysty käsittelemään viestiä.

CoAP:n pyyntö-/vastausmalli perustuu aliluvussa 4.1.1 käsiteltyihin metodi- ja vastauskoodeihin. Pyyntö välitetään NON- tai CON-tyyppisenä viestinä [39]. Jos pyynnön mukainen vastaus on heti saatavilla, se kuljetetaan kuittauksen (ACK) mukana pyynnön lähettäjälle (piggypacking). Mikäli vastausta ei ole heti saatavilla, pyyntö kuitataan tyhjällä kuittauksella, jolloin pyyntöä ei uudelleenlähetetä turhaan ja vastaus lähetetään myöhemmin CON-pyynnön mukana, jonka alkuperäinen lähettäjä kuittaa. Mikäli CON-pyynnön sijasta käytettäisiin NON-pyyntöä, myös vastaus toimitetaan NON-pyynnön mukana. CON-pyyntö ja kuittauksen mukana kuljetettava vastaus on havainnollistettu kuvassa 4.2.

CoAP-viestit voivat saapua väärässä järjestyksessä, duplikaatteja voi esiintyä tai ne voivat kadota [39]. Tästä syystä CON- ja NON-tyyppisten pyyntöjen toteuttama kevyt luotettavuusmekanismi on tarpeellinen epäluotettavaa UDP-protokollaa käytettäessä.



Kuva 4.2: CoAP-viestinnän havainnollistaminen [39]

4.1.3 Metodit

Pyyntökoodien osalta on huomattava, että vaikka ne vastaavat suurelta osin HTTP-protokollan vastaavia, mutta niissä on myös joitakin eroavaisuuksia [39]. Tässä aliluvussa käsitellään aliluvussa 4.1.1 esitetyt pyyntökoodit sekä niihin liittyvät yleisimmät vastauskoodit. Vastauskoodeja on useita ja ne on listattu liitteessä A, eikä niitä käsitellä tarkemmin tämän tutkielman puitteissa.

GET

GET-metodi palauttaa esityksen informaatiosta, joka vastaa sillä hetkellä pyynnön URI:n osoittamaa resurssia. Jos pyyntö sisältää Accept-valinnan, se osoittaa halutun sisältöformaatin vastaukselle. Onnistuneessa vastauksessa GET-metodiin tulee olla 2.05 (Content) tai 2.03 (Valid) vastauskoodi.

POST

POST-metodi pyytää, että pyyntöön sisältyvä esitys käsitellään. Todellinen toiminta metodiin liittyen määritetään palvelimella ja se riippuu kohderesurssista. Yleensä tällöin luodaan uusi resurssi tai päivitetään kohteena oleva resurssi.

Onnistuneessa resurssin luomisessa palvelimella tulisi vastauksessa palauttaa

2.01 (Created) vastauskoodi ja sisällyttää vastaukseen Location-Path tai Location-Query valinnat. Jos metodi onnistuu, mutta uutta resurssia ei luoda tulee vastauksessa olla 2.04 (Changed) vastauskoodi tai kohderesurssin poistamisen tapauksessa koodi 2.02 (Deleted).

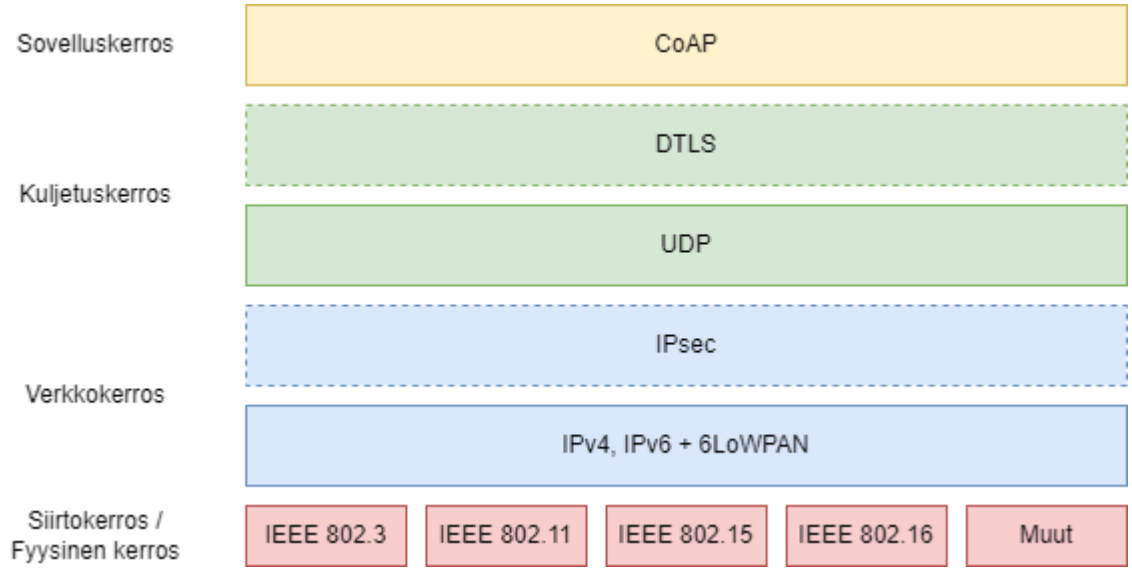
PUT
PUT-metodi pyytää URI:ssa määritettyä resurssia päivitettävän tai luotavan viestin esityksen mukaisesti. Esitysformaatti on määritelty mediatyypin ja sisältökoodauksen mukaisesti Content-Format-valinnassa.

Jos resurssi on olemassa pyydetyssä URI:ssa esitys tulisi ajatella modifioituna versiona tuosta resurssista ja 2.04 (Changed) vastauskoodi tulisi palauttaa. Jos resurssia ei ole, palvelin voi luoda uuden resurssin kyseiseen URI:in ja palauttaa 2.01 (Created vastauskoodin). Jos resurssia ei voida luoda tai modifioida, tulisi palauttaa asianmukainen virhekoodi.

DELETE
DELETE-metodi pyytää tunnistamaan pyynnön URI:n mukaisen resurssin poistamista. Vastauskoodi 2.02 (Deleted) palautetaan pyynnön onnistumisen yhteydessä tai siinä tapauksessa, että resurssia ei ollut olemassa ennen pyyntöä.

4.2 CoAP:n protokollapino

CoAP:n protokollapino rakentuu kuvan 4.3 mukaisesti. Kuljetuskerroksella käytetään UDP:ta (User Datagram Protocol) ja mahdollisesti DTLS:ää (Datagram Transport Layer Security), mikäli se on otettu käyttöön. Alempien kerrosten osalta CoAP-protokollaa ei ole sidottu tiettyihin protokolliin, vaan niillä voidaan käyttää IoT:n protokollapinon mukaisesti eri protokollia. CoAP:n suojaus on mahdollista toteuttaa myös verkkokerroksella IPsec:llä [39]. DTLS:n on kuitenkin katsottu olevan kevyempi ja helpompi toteuttaa resurssirajoitteisissa verkkonoodeissa, jolloin se on valittu CoAP:n kanavan oletussuojausmekanismiksi [6].



Kuva 4.3: CoAP:n protokollapino [33]

4.2.1 DTLS-protokolla

CoAP-protokollan suojaaminen toteutetaan DTLS-protokollalla, joka osaltaan myös varmistaa pakettien toimituksen käytettäessä yhteydetöntä UDP-protokollaa yhteydellisen TCP:n sijasta (Transmission Control Protocol) [18]. Kuten HTTP suojataan TLS:llä TCP:n yli, suojataan CoAP DTLS:llä UDP:n yli [39]. Käytännössä DTLS on TLS, johon on lisätty ominaisuuksia UDP:n epäluotettavan luonteen huomioimiseksi. DTLS:ää voidaan käyttää CoAP:n lisäksi myös muissa UDP:tä hyödyntävissä sovelluksissa, joista tunnettuja ovat esimerkiksi VoIP (Voice over IP) ja SIP (Session Initation Protocol) [2].

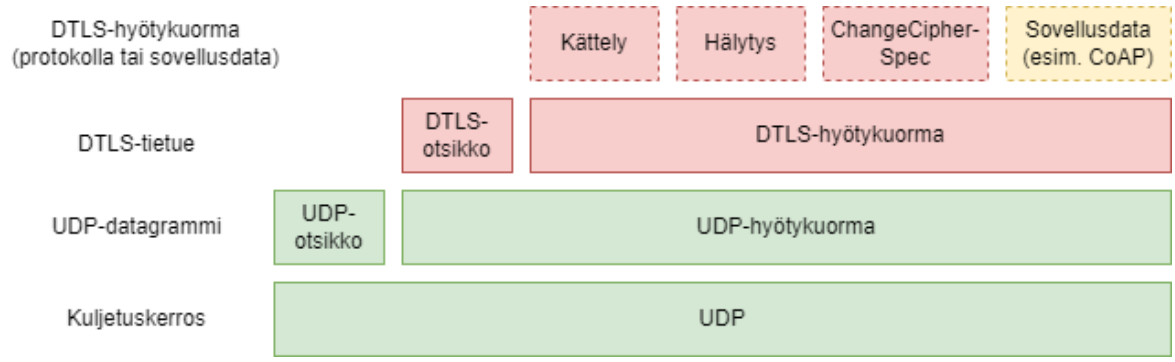
DTLS mahdollistaa todentamisen, tietojen eheyden, luottamuksellisuuden sekä automaattisen avaintenhallinnan [2]. Se tukee myös laajaa valikoimaa erilaisia salausalgoritmeja, mikä tekee siitä potentiaalisen mahdollisen suojausprotokollaehdokkaan. DTLS on kuitenkin alun perin kehitelty tehokkaille laitteille, jotka ovat yhdistettyjä luotettavan ja korkean siirtokapasiteetin yhteydellä [10]. Se on puhelias protokolla ja vaatii useita viestejä muodostaakseen suojatun istunnon ja on alun perin suunniteltu verkkoskenaarioihin, joissa viestin pituus ei ole ollut kriittinen tekijä, näin ollen se on sellaisenaan tehoton käytettäväksi rajoittuneissa IoT-laitteissa. Kaikkia DTLS:n tiloja ei rajoittuneisuuden takia voida välttämättä ollenkaan käyttää kaikilla laitteilla ja kaikissa verkoissa [39].

DTLS-tietue on 8 bittiä pidempi kuin TLS:n. DTLS:n kättelyn jälkeen lisätään 13 bittiä enemmän datagrammia kohden [33]. DTLS:n otsikot ovat myös liian pitkiä mahtuakseen yksittäiseen IEEE 802.15.4 MTU:hun (Maximum Transmission Unit, suurin siirtoyksikkö). Muita DTLS:n ongelmia ovat avainten hallinnan haasteet, sekä tuen puute CoAP:n välityspalvelin tilalle [33].

DTLS:n toiminta

DTLS:ää voidaan käyttää todennuksen protokollana verkkoon tulevien uusien laitteiden todentamisessa joko PSK-tilassa (PreSharedKey, ennalta jaettu avain), RawPublicKey-tilassa tai julkisen avaimen varmenteen (certificate, sertifikaatti) avulla [10]. Onnistuneen DTLS-kättelyn tuloksena luodaan suojattu kanava uuden laitteen ja valtuuttavan entiteetin välille. Suojattu kanava mahdollistaa valtuuttavan entiteetin jakaa L2-avaimen turvallisesti liittyvälle laitteelle verkon omistajan määrittämien sääntöjen perusteella. Sovellustiedot suojataan DTLS-tietuekerroksella (DTLS Record Layer) istuntoavaimella.

DTLS koostuu kahdesta kerroksesta: alempi kerros sisältää DTLS-tietueen ja ylempi sisältää joko jonkin kolmesta protokollasta, jotka ovat kättely (Handshake), hälytys (Alert) ja ChangeCipherSpec, tai suojattavan sovellusdatan [10]. DTLS-suojatun paketin rakenne on esitetty kuvassa 4.4



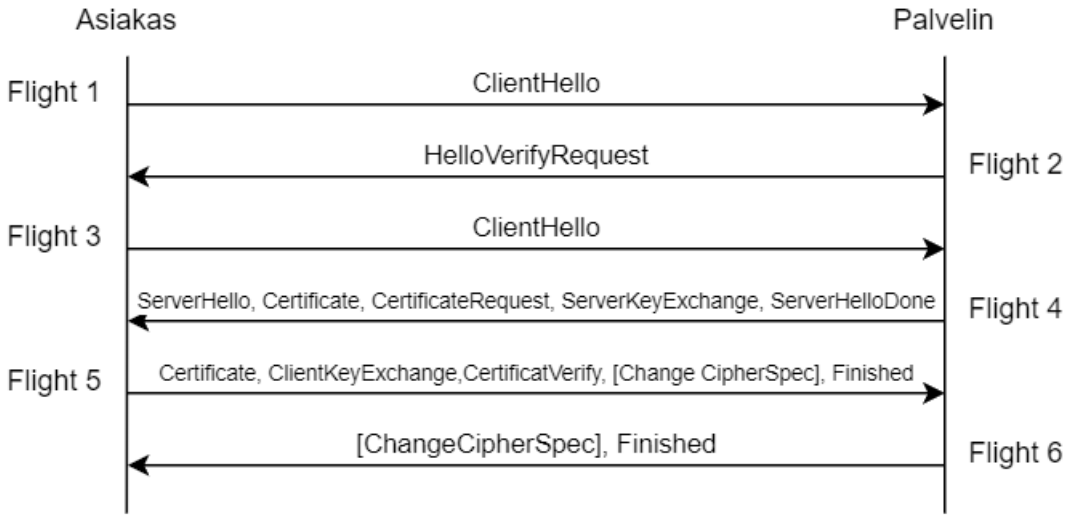
Kuva 4.4: DTLS-suojattu liikenne [35]

DTLS-tietueen otsikko sisältää muun muassa sisältötyypin, protokollan version ja tietueen järjestysnumeron [36]. Sisältötyypin arvon perusteella tietueen fragmenttikenttä, joka on kuvassa 4.4 esitetty DTLS:n hyötykuormana, sisältää joko kättelyprotokollan, hälytysprotokollan, ChangeCipherSpec-protokollan tai sovellusdatan.

Kättelyprotokolla on monimutkainen keskusteluprosessi ja sisältää lukuisia vies-

tien vaihtoja asynkronisella tavalla [10]. ChangeCipherSpec-protokollaa käytetään kättelyprosessin aikana osoittamaan, että tietueprotokollan tulee suojata seuraavat viestit äskettäin neuvotellun salauspaketin ja suojausavaimien avulla [35]. Hälytysprotokollaa puolestaan käytetään virheilmoitusten välittämiseen.

DTLS-kättelyssä käytetään kättelyviestejä, jotka yleensä järjestetään lennoiksi (flights) nimitettäviin ryhmiin, neuvottelemaan turva-avaimista, salaussarjoista ja pakkausmenetelmistä. Täydellinen kättelyprosessi on esitetty kuvassa 4.5.



Kuva 4.5: DTLS-kättely [10]

DTLS 1.3

DTLS:n versio 1.3, joka määritellään RFC 9147 -dokumentissa [37] sisältää suuren määrän muutoksia versioon 1.2 verrattuna. Seuraavassa listassa on käsitelty keskeisimmät muutokset RFC 9147 -dokumentin mukaisesti:

- Uusi kättelymalli, joka johtaa lyhyempään viestien vaihtoon
- Tuki vain AEAD-salauksille. Lisätietojen laskentaa yksinkertaistettu
- Heikkojen ja vanhempien salausalgoritmien tuen poisto
- HelloRetryRequest -pyynnön käyttäminen HelloVerifyRequestin sijaan
- Joustavampi salausratkaisun neuvottelu
- Uusi istunnon jatkamismekanismi

- PSK-todennuksen uudelleenmäärittely
- Uusi avaimen johtamisen hierarkia, joka hyödyntää uutta avaimen johtamisrakennetta
- Parannettu version neuvottelu
- Optimoitu tietuekerroksen koodaus ja koko
- Lisätty CID-toiminto
- Järjestysnumerojen salaus

Koska keskeinen ongelma CoAP:n salauksen käytössä on lisääntynyt liikennöintimäärä, voidaan CoAP:n osalta tärkeiksi muutoksiksi nähdä ainakin kaikki viestinnän määrään vaikuttavat ominaisuudet. Viestinnän määrää vähentäviä ominaisuuksia ovat ainakin uusi kättelymalli, salauksen lisätietojen laskennan yksinkertaistus ja tietuekerroksen optimointi. Toisaalta muutoksien joukossa voi olla myös muutoksia, jotka muiden ominaisuuksien, kuten esimerkiksi protokollan turvallisuuden parantamisen ohella voivat myös lisätä liikennöinnin määrää. Näin ollen ilman tarkempaa tarkastelua ei voida arvioida, mikä version 1.3 käytön kokonaisvaikutus CoAP-protokollan näkökulmasta olisi.

4.2.2 UDP-protokolla

UDP-protokolla on määritetty tarjoamaan datagrammitila pakettivälitteiseen tietokoneiden viestintään toisiinsa yhdistetyn tietokoneverkkojoukon ympäristössä [31]. Protokolla tarjoaa sovelluksille menetelmän lähettää viestejä muille ohjelmille minimaalisella protokollamekanismilla. Protokolla on tapahtumakeskeinen, eikä takaa toimitusta ja suojausta duplikaateilta.

UDP-otsikko muodostuu sen määrittelevän RFC 768 -dokumentin [31] mukaisesti viidestä kentästä, jotka on esitelty taulukossa 4.3

Taulukko 4.3: UDP-otsikon kentät [31]

Kenttä	**Selite**
Lähdeportti (Source Port)	Lähettävä portti (valinnainen)
Kohdeportti (Destination port)	Vastaanottava portti
Pituus (Length)	Datagrammin pituus oktetteina
Tarkistussumma (Checksum)	Datagrammin oikeellisuuden tarkastus
Dataoktetit (Data octets)	Paketin sisältämä data

CoAP:n näkökulmasta UDP tarjoaa kevyen kuljetuskerroksen protokollan, joka sopii hyvin rajoittuneille laitteille. Koska viestien toimitusta ei varmenneta UDP:n toimesta, vastuu luotettavuuden toteutuksesta siirtyy joko CoAP-protokollalle (CON-viestit) tai DTLS:lle.

4.2.3 IPsec

IPsec (Internet Protocol Security) on sarja protokollia, jotka tarjoavat suojausta Internet-kommunikaatiolle IP-kerroksella [14]. Yleisin käyttötapaus on VPN-yhteyksien (Virtual Private Network) tuottaminen joko kahden sijainnin välille tai etäkäyttäjän ja yrityksen verkon välille. IPsecillä voidaan toteuttaa myös päästä päähän -salausta. IKE (Internet Key Exchange) on avainten neuvottelu- ja hallintaprotokolla, jota käytetään yleisimmin tarjoamaan dynaamisesti neuvoteltua ja päivitettyä avainmateriaalia IPsecille.

IPsec on sovelluksesta riippumaton protokolla, joka voi suojata sekä sovellus- että siirtokerroksen sovelluksia [2]. IPSecin tarjoamia turvallisuusominaisuuksia ovat: yhteydetön eheys, pääsynhallinta, tietojen alkuperän todennus, luottamuksellisuus, toiston estomekanismi ja rajoitettu liikennevirran luottamuksellisuus.

CoAP:n määrittelevä RFC 7252 -dokumentti [39] esittää IPsecin yhdeksi tavaksi toteuttaa CoAP:n suojaus. Dokumentissa ei kuitenkaan oteta kantaa IPsecin käytön toteutukseen vaan viitataan aihetta käsittelevään RFC-luonnokseen. Luonnos on kuitenkin vuodelta 2012, eikä sitä ole myöhemmin edistetty. Näin ollen CoAP:n suojaamista IPsecillä ei ole yhtenäisesti määritelty.

Almghadi et al. esittävät artikkelissaan [2] useita ongelmia IPsecin käytössä CoAP:n suojaamiseksi. Kuten DTLS:n tapauksessa, myöskään IPseciä ei ole alkujaan suunniteltu rajoittuneille ympäristöille, jolloin laitteiden mahdollista rajoittuneisuut-

ta ei ole otettu huomioon suunnittelussa. Esimerkiksi lähetettäessä pieniä paketteja IPSecin salausprosessi tuottaa suuren kustannuksen, mikä heikentää verkon suorituskykyä.

IPSecin konfigurointi, hallinta ja vianetsintä on monimutkainen tehtävä, mikäli verkossa on suuri määrä rajoitettuja laitteita [2]. IPSecin turvallisuusparametrien virheelliset asetukset voivat johtaa tietoturva-aukkoihin tai suorituskykyongelmiin. IoT:n liikkuvuus on myös ongelmallinen IPsecin suhteen, sillä IP-osoitteen muuttuessa, joutuu IPsec muodostamaan SA:n (Security Association) uudelleen, mikä heikentää suorituskykyä. Kaikkia skenaarioita ja solmuja ei myöskään voida tukea IPsecillä. Esimerkiksi jos käytössä on kaksi eri ympäristöä erilaisilla hallintakäytännöillä ja toinen puoli ei tue IPSeciä, ei palveluita voida tarjota. Lisäksi IPSec-tuen toteuttaminen multicast-viestinnälle on haastavaa.

4.3 CoAP:n turvallisuustilat

CoAP-laitteen tarvitsemat suojaustiedot, mukaan lukien avainmateriaalit ja pääsylistat (Access control lists, ACL) toimitetaan provisiointivaiheen aikana [39]. Provisiointivaiheen lopussa laite on jossakin neljästä RFC-dokumentissa määritellystä suojaustilasta:

- NoSec
- PreSharedKey
- RawPublicKey
- Sertifikaatit

4.3.1 NoSec

NoSec-tilassa protokollatason suojaus eli DTLS ei ole käytössä [39]. Tilassa järjestelmä lähettää paketit normaalilla tavalla UDP:n yli IP:tä käyttäen. Järjestelmä voidaan turvata vain estämällä hyökkääjiä lähettämästä ja vastaanottamasta paketteja CoAP-laitteiden verkossa. Tällöin suojaus tulisi tarvittaessa toteuttaa alemmilla kerroksilla.

CoAP:n suojaukseen alemmilla kerroksilla on mahdollista käyttää esimerkiksi IPSeciä [46]. Lisäksi voidaan käyttää sovelluksen omaa sisäistä hyötykuorman sa-

lausta ja salauksen purkua. Tämä siirtää kuitenkin rajoitteet IoT-palvelun suunnittelijan ratkaistavaksi ja nostavat liikennöintimääriä kyseisellä kerroksella.

4.3.2 PreSharedKey

PreSharedKey-tilassa DTLS on käytössä ja laitteilla on olemassa lista ennalta jaettuja avaimia (pre-shared key), joista jokainen sisältää listan siitä, mihin noodeihin kommunikoitaessa kyseistä avainta voidaan käyttää [39]. Äärimmillään noodilla voisi olla oma avain jokaista noodia kohti, jonka kanssa sen on tarkoitus kommunikoida. Jos sama avain on käytössä useammilla noodeilla, mahdollistaa avain autentikoinnin vain ryhmän jäsenenä, ei tiettynä yksilönä. PreSharedKey sopii sovelluksille, jotka eivät voi hyödyntää julkisen avaimen kryptografiaa [33].

PreSharedKey-tilassa käytettävä avain on symmetrinen ja liikennöinnissä data salataan sekä allekirjoitetaan tätä jaettua salaisuutta käyttämällä [46]. Vastaanotossa kohde käyttää samaa avainta turvalliseen salauksen purkuun ja viestien autentikointiin. Jos salaus halutaan varmistaa, on suositeltavaa, että eri yhteyksille käytetään eri avaimia. Tällöin palvelinlaitteet ylläpitävät listaa avaimista ja tunnisteista, joita käytetään erilaisten suojattujen keskustelujen yhdistämiseen (tämä koskee myös asiakaslaitetta, joka kommunikoi useiden palvelinten kanssa). PreSharedKey-tilassa oletetaan, että avainten jakelu toteutetaan kaistan ulkopuolella (out-of-band) ja luotetaan vahvasti avainten turvalliseen säilömiseen.

Vaikka PreSharedKey-tilan ennalta jaetut avaimet mahdollistavat nopeamman käsittelyn ja suhteellisen helpon käyttöönoton, täydellisen eteenpäin salauksen (perfect forward secrecy) puute on ongelmallinen silloin, kun hyökkääjä kykenee tallentamaan liikennettä, jolloin on mahdollista purkaa ne myöhemmin [46]. Voidaan myös olettaa, että automaattisen luomisen sijaan ennalta jaetun salaisuuden voi syöttää myös ihminen. Tällöin vastuu loppukäyttäjälle on suuri, sillä heikko avain voi olla altis raaka voimaa hyödyntäville hyökkäyksille.

4.3.3 RawPublicKey

RawPublicKey-tilassa DTLS on käytössä ja laitteella on epäsymmetrinen avainpari ilman sertifikaattia (raw public key), joka on varmistettu ulkoisella mekanismilla [39]. Laitteessa on myös julkisesta avaimesta laskettu identiteetti ja luettelo noodeista, joiden kanssa se voi kommunikoida. RawPublicKey on pakollinen tila laitteille jotka vaativat julkiseen avaimeen perustuvan autentikoinnin [33]. Ennalta jaetun

listan ansiosta DTLS-istunto on mahdollista muodostaa ilman sertifikaattia. Etuna perinteisiin sertifikaatteihin on pienempi prosessointiaika, sillä luottamusketjujen validointia ei vaadita.

RawPublicKey-tila mahdollistaa täydellisen eteenpäin salauksen konseptin, varmistaen, että aiemmin kaapatun ja tallennetun viestinnän salausta ei voida purkaa, vaikka avainparit paljastuisivat [46]. Lisäksi ihminen poistetaan avainten generoinnista, mikä osaltaan parantaa turvallisuutta. Kommunikointitapa mahdollistaa avainparien käytön sertifikaattien tapaan, mutta ilman varsinaista sertifikaattia, joka autentikoisi julkisen avaimen omistajan. Lisäksi tilaa voidaan käyttää myös pienemmissä käyttöönotoissa, joissa kaistan ulkopuolinen jakelu on mahdollista, esimerkiksi ennakkojakeluna valmistushetkellä.

Julkisten avainten autentikointi voidaan toteuttaa ennakkojakelulla, jolloin molemmilla osapuolilla on lista, joka kuvaa identifioidun päätepisteen (esim. IP-osoite, domain-nimi) sen julkisella avaimella tai käyttämällä niin sanottua DNS-perusteista autentikointia nimetyille entiteeteille, jotka ylläpitävät periaatteessa samaa luetteloa keskitetysti [46]. Esimerkkinä tästä voisi olla älykodin tiloissa oleva gateway-laite, joka ylläpitää listaa kaikista laitteista kotona ja niiden julkisista avaimista. Kun kaksi osapuolta haluavat muodostaa yhteisen yhteyden ja luoda salatun dialogin RPK:n (Raw Public Keys) kautta, yhdyskäytävä todentaa heidän identiteettinsä.

4.3.4 Sertifikaatit

Sertifikaatteja käytettäessä DTLS on käytössä ja laitteella on epäsymmetrinen avainpari sekä X.509 sertifikaatti, joka sitoo sen kohteeseensa ja on allekirjoitettu yleisen luottamusjuuren toimesta [39]. Laitteella on myös lista luottamusjuurista, joita voidaan käyttää sertifikaatin validointiin. Oletuksena tilassa on, että turvallisuusinfrastruktuuri on olemassa[33]. Laitteet, joissa on asymmetrinen avain ja joilla on tuntematon X.509 sertifikaatti, voidaan varmistaa käyttäen sertifikaatteja ja luotettuja juuriavaimia.

Sertifikaatit ovat perinteinen tapa valvoa turvallisuutta ja siinä sovelletaan samaa konseptia kuin RawPublicKey-tilan kanssa [46]. Huomattavana erona on se, että avaimet validoidaan Certificate Authority:na (CA, sertifikaatin myöntäjä) tunnetun tahon vahvistamana. Kyseinen tekniikka on ollut perustana turvalliselle kommunikaatiolle koko internetissä, erityisesti HTTP:llä käytettävien resurssien osalta.

Kääntöpuolena sertifikaateilla on niiden raskas formaatti, mikä korostuu erityisesti rajoittuneiden laitteiden tapauksessa, sekä hinta, sillä oikeiden käyttöönotet-

tavien varmenteiden tulee olla vakiintuneiden turvallisuusyritysten allekirjoittamia [46]. Tämä voidaan kuitenkin kiertää tuottamalla itse allekirjoitettuja varmenteita päätepistelaitteisiin. Tilan selkeänä etuna on mahdollisuus peruuttaa varmenteet, jos päätepiste vaarantuu, kuten esimerkiksi fyysisen kaappauksen tapauksessa, jonka jälkeen sitä voitaisiin käyttää eri tarkoitukseen kuin ennen.

4.4 CoAP:n tietoturvauhat

Rahman ja Shah [33] ovat todenneet vuonna 2016, että CoAP-protokollan suurimpana haasteena on säilyttää korkea suorituskyky ja samalla säilyttää turvallisuuden standardit, sekä tarjota suojausta. Lisäksi todetaan, että keskeinen tutkimus turvallisuuden hallinnasta CoAP:n osalta on puutteellista, eikä turvallisille arkkitehtuureille ole olemassa luotettavia standardeja.

Nebbione et al. [10] ovat analysoineet CoAP:ia käyttävien laitteiden mahdollisia turvallisuusuhkia ja tunnistaneet analyysin pohjalta potentiaalisesti haavoittuvaisia prosesseja. Arvind ja Naraynan [4] ovat luetelleet CoAP:n RFC-dokumenttiin [39] pohjautuen siihen mahdollisesti kohdistettavia hyökkäyksiä. Edellisten perusteella voidaan erottaa viisi oleellista CoAP:hen kohdistettavaa hyökkäystä, sekä kaksi haavoittuvaista prosessia, joita voidaan käyttää hyökkäysten pohjana:

- IP-spoofing
- Vahvistushyökkäykset
- Protokollien väliset hyökkäykset
- Jäsennyshyökkäykset
- Välimuistihyökkäykset
- Bootstrapping-prosessi
- Heikko avainten generointi

4.4.1 IP-spoofing

IP-spoofingilla, eli IP-osoitteen väärentämisellä hyökkääjä kykenee väärentämään viestin alkuperän. CoAP:n osalta IP-osoitteen väärentämisen mahdollistaa UDP-protokollan kättelyn puute [39]. Hyökkääjän laite, jolla on mahdollisuus lukea ja

kirjoittaa rajoittuneessa verkossa kuljetettavia viestejä voi helposti hyökätä yksittäiseen laitteeseen, laiteryhmään tai koko verkkoon. Esimerkkejä spoofingia hyödyntävän hyökkääjän mahdollisista viesteistä ja niiden vaikutuksista on esitetty taulukossa 4.4.

Taulukko 4.4: Väärennetystä IP-osoitteesta lähetettävien viestin vaikutuksia [39]

Viesti	**Mahdollinen vaikutus**
Reset-viesti vastauksena CON- tai NON-viestiin	Kohteen jääminen "kuuroksi"
ACK-vastaus CON- tai NON-viestiin	Viestin uudelleenlähetyksen estyminen ja varsinaisen vastauksen hukkuminen
Vastauksen hyötykuorman tai valintojen väärentäminen	Vastauksen häirintä, välityspalvelimen välimuistin myrkyttäminen, verkon tilan tallentavien ja CoAP:ia viestinnässään (asetusten tai tilan päivittämiseksi) käyttävien komponenttien huijaaminen
Multicast-pyyntö	Verkon ruuhkautuminen tai kaatuminen, palvelunestohyökkäys uhria kohtaan, pakotettu herätys levosta

Reittien ulkopuolisten hyökkääjien vastauksien spooffaaminen voidaan havaita ja lieventää (mitigate) myös ilman siirtokerroksen suojausta käyttämällä satunnaista, ei-triviaalia tokenia pyynnöissä [39]. Muun tyyppisiä spoofing-hyökkäyksiä voidaan CoAP:ssa havaita periaatteessa vain CON-viestin semantiikkaan perustuen, koska spooffatusta laitteesta tulee odottamattomia ACK- tai RST-viestejä. Tämä edellyttää kuitenkin käytettyjen viestitunnuksien (Message ID) seurantaa, mikä ei aina ole mahdollista ja lisäksi havaitseminen onnistuu yleensä vasta kun vahinko on jo tapahtunut.

Spoofingia voidaan käyttää myös muiden hyökkäysten mahdollistamiseen kuten esimerkiksi vahvistushyökkäyksen tapauksessa tai niiden tukemiseen, kuten palvelimen ylikuormittamiseen pyrkiessä. Palvelimen ylikuormittamiseen pyrkiessä spoofingia voidaan hyödyntää hyökkäyksen jäljittämisen ja hyökkäyksen estämisen vaikeuttamiseksi [39]. Itse hyökkäyksessä lähetetään pyyntöjä (mieluiten monimutkaisia) palvelimelle. Koska CON-viestin kustannus on pieni, hyökkäys on kus-

tannustehokas ja helppo. Hyökkäyksen alainen rajoitettu solmu voi energiansa osalta kulua loppuun paljon suunniteltua nopeammin (virrankulutushyökkäys). Lisäksi, jos hyökkääjä käyttää CON-viestiä ja palvelin vastaa erillisellä CON-vastauksella (mahdollisesti spooffattuun) osoitteeseen, joka ei vastaa, on palvelimen varattava puskuria sekä uudelleenlähetyslogiikka jokaiselle vastaukselle, mikä rajoittaa resursseja asiallisen liikenteen käsittelyyn. Lisäksi spooffattua osoitetta käytettäessä palvelin voi mahdollisesti estää asialliset vastaukset osoitteen omistajalle.

Spoofing-hyökkäysten tehokas estäminen on mahdollista käyttämällä muuta kuin NoSec-turvallisuustilaa [39]. Tällöin hyökkäykset kohdistuvat vain suojaavaan protokollaan.

4.4.2 Vahvistushyökkäys

Vahvistushyökkäyksissä hyökkääjä käyttää loppulaitteita muuntaakseen pienemmän paketin suuremmaksi paketiksi [4]. CoAP on monien muiden protokollien tapaan potentiaalinen kohde erilaisille palvelunestohyökkäyksille (Denial of Service, DoS) ja vahvistushyökkäys on yksi näistä CoAP:hen kohdistuvista palvelunestohyökkäyksistä [3]. Vahvistushyökkäyksessä hyökkääjä käyttää hyväksi spoofingia siten, että hyökkääjän lähettää viestejä verkkoon käyttäen kohteen IP-osoitetta, jolloin vastaukset viesteihin lähetetään kohteelle hyökkääjän sijaan. Vastausten koko ja/tai määrä pyritään saamaan suureksi suhteessa alkuperäiseen viestiin, mistä tulee nimitys vahvistushyökkäys.

Vahvistushyökkäyksessä viestejä voidaan lähettää yksittäisestä kohteesta (palvelunesto) tai useista kohteista (hajautettu palvelunesto, Distributed Denial of Service, DDoS) ja niissä voidaan käyttää hyväksi yhtä palvelinta, useita palvelimia, sekä multicast- ja broadcast-viestejä [3]. Vahvistushyökkäysten vahvistuskerroin voidaan laskea hyökkääjän tuottaman tiedon ja uhrille lähetettyjen todellisten tietojen välisen suhteen perusteella. Erilaisia vahvistushyökkäyksen tyyppejä on esitetty taulukossa 4.5.

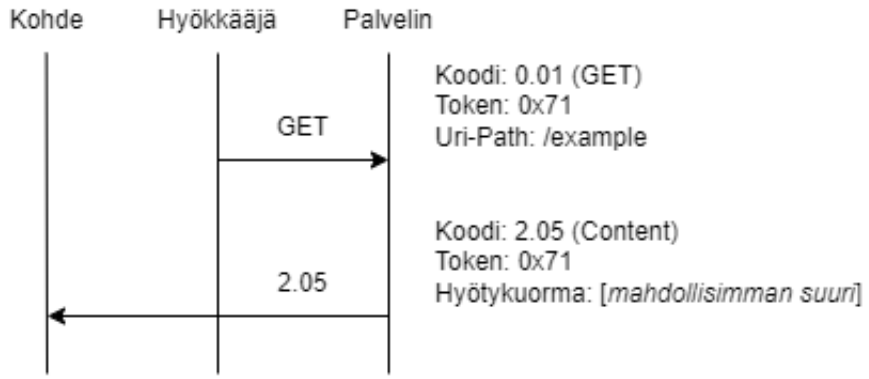
Trend Micro on toteuttanut vahvistushyökkäyksen testauksen CoAP-laitteiden ja -palvelinten verkossa kasvavalla hyötykuorman määrällä [20]. Raportin arvion mukaan CoAP voisi saavuttaa 32-kertaisen vahvistuskertoimen, joka on karkeasti DNS:n ja SSDP:n kertoimien välissä. Tällöin 1 Mbps yhteyden omaava hyökkääjä kykenee toteuttamaan 32 Mbps hyökkäyksen.

Taulukko 4.5: Vahvistushyökkäyksen tyyppejä [3]

Tyyppi	**Periaate**	**Kerroin**
Yksinkertainen	Palvelin vastaa pyyntöön kerran pyyntöä x kertaa suuremmalla vastauksella	x
Observe	Palvelin vastaa y kertaa pyyntöä kohti pyyntöä x kertaa suuremmalla vastauksella	x*y
Multicast	z palvelinta vastaa y kertaa pyyntöä kohti pyyntöä x kertaa suuremmalla vastauksella	x*y*z

Yksinkertainen vahvistushyökkäys

Yksinkertaisessa vahvistushyökkäyksessä hyökkääjä lähettää spooffatun pyynnön palvelimelle, joka vastaa kohteelle. Esimerkki yksinkertaisesta vahvistushyökkäyksestä on esitetty kuvassa 4.6.



Kuva 4.6: Yksinkertainen vahvistushyökkäys [3]

Yksinkertaisen hyökkäyksen vahvistuskerroin muodostuu pyynnön koosta suhteessa vastauksen kokoon [3]. Kertoimen maksimoimiseksi hyökkääjän kannattaa tavoitella mahdollisimman suurta vastausta, joka on CoAP-vastauksen maksimikoko eli 1024 tavua. Hyökkäys toimii vain spoofingin onnistuessa, sillä muuten vastaukset eivät kohdistu kohteelle. Hyökkäyksen kokonaisvaikutusta voidaan kasvattaa lähettämällä useita GET-viestejä.

Esimerkiksi vahvistuskertoimesta x yksinkertaisessa hyökkäyksessä, jossa pyyn-

nön koko p on 50 tavua ja vastauksen v koko 1024 tavua lasketaan

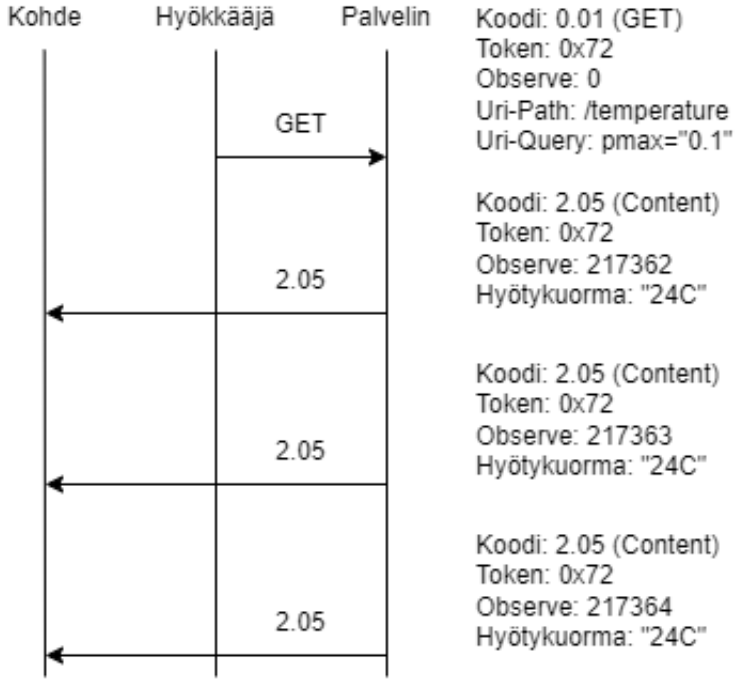
$$x = \frac{v}{p} = \frac{1024}{50} = \mathbf{20{,}48} \tag{4.1}$$

eli vahvistuskerroin on noin 20-kertainen.

Observe-tyyppinen vahvistushyökkäys

Observe-tyyppisessä vahvistushyökkäyksessä hyödynnetään CoAP:n tarkkailuominaisuutta (observe). Observe-ominaisuus on CoAP-protokollan laajennus, joka mahdollistaa CoAP-asiakkaille resurssien tarkkailun [16]. Asiakas voi esimerkiksi saada esityksen resurssista ja pitää tämän esityksen ajan tasalla suhteessa palvelimeen tietyn aikaa. Protokolla lähettää uusia esityksiä asiakkaille ja tuottaa lopulta yhdenmukaisen tilan asiakkaiden ja palvelimen todellisen tilan välille.

Hyökkäyksessä lähetetään palvelimelle observe-tyyppinen pyyntö, jolloin palvelin alkaa päivittämään kyseistä resurssia kohdelaitteelle. Pyyntö voi saada esimerkiksi 10 vastausta palvelimelta [3]. Observe-tyyppinen hyökkäys on esitetty kuvassa 4.7



Kuva 4.7: Observe-tyyppinen vahvistushyökkäys [3]

Observe-tyyppisen hyökkäyksen vahvistuskerroin muodostuu vastausten määrästä sekä koosta suhteessa lähettäjään ja on yleensä huonompi kuin käytettäessä

yksittäistä vastausta [3]. Kuten kuvan 4.7 esimerkissä, voi resurssin esitys olla hyvin pieni, kuten lämpötila-arvo ja todennäköisesti useimmat palvelinten tarkasteltavat resurssit ovatkin pieniä suhteessa pyynnön kokoon, eikä suurta vahvistusvaikutusta saada aikaiseksi. Toisaalta, jos käytössä on useita palvelimia, voidaan observe-pyyntöjä lähettää niille kaikille, jolloin liikenteen määrä kasvaa.

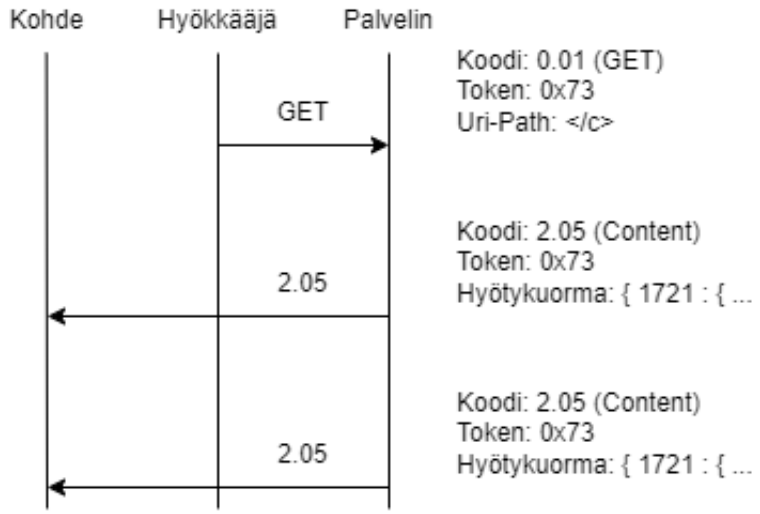
Esimerkiksi vahvistuskertoimesta x observe-tyyppisessä vahvistushyökkäyksessä, jossa pyynnön koko p on 50 tavua, vastauksen v koko 50 tavua ja vastauksien määrä y 10 lasketaan

$$x = \frac{v \times y}{p} = \frac{50 \times 10}{50} = \mathbf{10} \tag{4.2}$$

eli vahvistuskerroin on 10-kertainen.

Multicast-tyyppinen vahvistushyökkäys

Multicastia hyödyntävässä vahvistushyökkäyksessä pyyntö lähetetään useille palvelimille käyttämällä multicastia tai broadcastia [3]. Multicastia hyödyntävä vahvistushyökkäys on esitetty kuvassa 4.8.



Kuva 4.8: Multicast-tyyppinen vahvistushyökkäys [3]

Tämän hyökkäyksen tarkoituksena on tehdä CoAP-uhri toimintakyvyttömäksi. Kohteiden lukumäärää ei tunneta hyökkäyksessä, kuten ei myöskään väärennettyihin pyyntöihin vastaavien palvelimien määrää.

4.4.3 Protokollien väliset hyökkäykset

Yleisesti ottaen UDP-pohjaiset protokollat ovat kontekstin puutteen vuoksi suhteellisen helppoja kohteita protokollien välisille hyökkäyksille [39]. CoAP-laitetta voidaan protokollien välisissä hyökkäyksissä huijata lähettämään viestejä laitteille, jotka kuuntelevat UDP paketteja tietyssä osoitteessa ja portissa. Tässä voidaan käyttää spoofing-hyökkäystä. Paketti tulkitaan vastaanottajan käytössä olevan protokollan sääntöjen mukaan. Tällä voidaan kiertää esimerkiksi palomuureja, jotka voivat sallia liikenteen CoAP-laitteelta. CoAP-laite voi puolestaan olla kohteena esimerkiksi DNS-protokollan kautta. DTLS ei suojaa protokollien välisissä hyökkäyksissä, jos eri protokollat kulkevat saman DTLS-yhteyden sisällä.

Spoofingilla saavutettavaa lähdeosoitteen väärentämistä voidaan hyödyntää myös protokollien välisissä hyökkäyksissä (cross-protocol attack), joissa kohde kuuntelee UDP-paketteja annetusta osoitteesta (IP-osoite ja portti) [39]. RFC 7252 -dokumentissa esitetään protokollien välisen hyökkäyksen toteuttaminen CoAP:ssa seuraavasti:

1. Hyökkääjä lähettää CoAP-laitteelle viestin, jossa lähdeosoite on väärennetty
2. CoAP-laite vastaa viestillä annettuun lähdeosoitteeseen
3. Lähdeosoitteen omistava kohde saa UDP-paketin, jonka se tulkitsee toisen protokollan sääntöjen mukaan

Protokollien välistä hyökkäystä voidaan käyttää palomuurisääntöjen kiertämiseen, jotka estäisivät hyökkääjän ja uhrin välisen suoran viestinnän, mutta mahdollistavat tiedonsiirron CoAP-laitteelta kohteelle [39]. Toisaalta myös CoAP-laitteet voivat joutua toisen UDP-pohjaisen protokollan, kuten esimerkiksi DNS:n kautta generoidun protokollien välisen hyökkäyksen uhriksi. Molemmissa tapauksissa hyökkäykset mahdollistaa päätepisteiden suojauksen perustuminen IP-osoitteiden tarkistamiseen ja ulkopuolelta väärennetyillä IP-osoitteilla tulevien suorien hyökkäysten pysähtymiseen palomuuriin.

Yleisesti ottaen kahta protokollaa tarkasteltaessa voi toinen protokollista olla hyvinkin suunniteltu siten, että hyökkääjä voi saada aikaan vastauksia, jotka näyttävät toisen protokollan viesteiltä [39]. Protokollien väliset hyökkäykset voidaan estää täysin vain, jos päätepisteet eivät mahdollista hyökkääjän tavoittelemia toimenpiteitä toimia pelkästään paketin lähdeosoitteen perusteella.

NoSec-turvallisuustilaa käyttävä CoAP-ympäristö, joka luottaa täysin palomuuriin CoAP:n suojauksessa, tarvitsee palomuurin CoAP-päätepisteiden lisäksi myös kaikkien muiden protokollien päätepisteitä varten. Nämä päätepisteet voisivat lähettää UDP-viestejä CoAP-päätepisteisiin jollakin toisella UDP-pohjaisella protokollalla. DTLS:n osalta on myös huomioitava, että saman DTLS-yhteyden piirissä olevia eri protokollia voidaan myös käyttää protokollien välisiin hyökkäyksiin.

4.4.4 Jäsennyshyökkäys

Verkkoon yhteydessä oleva sovellus voi sisältää haavoittuvuuksia saapuvien pakettien prosessointilogiikassa [39]. Monimutkaiset jäsentimet (parser) ovat tunnettu lähde tällaisille haavoittuvuuksille. Nebbionen et al. mukaan [24] CoAP:n haavoittuvuuksiin liittyen yleisin tietoturvaongelma on viestien virheellinen jäsennys. Jäsennyshyökkäyksessä hyödynnetään CoAP-protokollan viestien jäsentimien haavoittuvuuksia, joissa asiakkaan ja palvelimen jäsentimien logiikka ei käsittele oikein tulevia viestejä. Haavoittuvuutta hyödyntäessä noodi voidaan esimerkiksi kaataa tai suorittaa siinä mielivaltaista koodia [39].

Esimerkkinä jäsennyksen haavoittuvuuksista voivat jotkut CoAP-kirjastot väärinkäyttää virheellisiä CoAP-viestin valintoja tai tiettyjä poikkeuksia vastaanottaessaan tietyllä tavalla muotoiltuja viestejä [24]. Lisäksi kirjastoihin voi liittyä myös ylivuotoon liittyviä haavoittuvuuksia saapuvaan viestin käsittelyn yhteydessä, kuten haavoittuvuudessa CVE-2019-17212[27].

Hyökkääjä voi hyödyntää näitä haavoittuvuuksia esimerkiksi lähettämällä virheellisen viestin noodille, jonka toiminta häiriintyy hyökkääjän tavoittelemalla tavalla, esimerkiksi noodin kaatuessa. Poikkeuksien väärästä käsittelystä esimerkki on haavoittuvuus CVE-2018-12679 [26], jossa CoAPthon3-kirjasto käsittelee tietyt poikkeukset väärin. Tämä johtaa tietynlaisen viestin saapuessa palvelunestoon kirjastoa käyttävissä sovelluksissa, kuten esimerkiksi CoAP-palvelimella tai CoAP-asiakkaalla. Haavoittuvuuden hyödyntämisessä voidaan käyttää esimerkkiä kyseisen ongelman aiheuttavasta hyötykuormasta, joka on saatavilla Githubista [15].

Nämä haavoittuvuudet ovat kuitenkin kirjasto- ja toteutuskohtaisia, eikä niiden hyväksikäyttö ole mahdollista, jos kirjasto tai sen haavoittuva versio eivät ole kohteessa käytössä. Vanhojen versioiden käyttö ja päivityksien laiminlyönti voivat kuitenkin mahdollistaa tunnettujen haavoittuvuuksien hyväksikäytön.

4.4.5 Välimuistihyökkäys

Välityspalvelimet ovat luonteeltaan miehiä välissä (man-in-the-middle) murtaen IPsec ja DTLS-suojaukset, joita suorassa CoAP-viestinvaihdossa voi olla [39]. Näin ollen ne ovat kiinnostavia kohteita hyökkääjille, jotka pyrkivät luottamuksellisuuden, eheyden tai saatavuuden murtamiseen. Uhka lisääntyy, kun myös välityspalvelimet käyttävät välimuistiin tallentamista.

Välityspalvelimien pääsynhallintamekanismien ja välimuistin väärä toteutus voi vaarantaa niiden sisällön ja näin vaarantaa CoAP-viestien luottamuksellisuuden ja eheyden [24]. Välimuistihyökkäyksissä välityspalvelin, jolla on kyky tallentaa välimuistiin, voi saada hallinnan, joka uhkaa välityspalvelimen asiakkaita, jotka eivät havaitse tunkeilijaa [4].

4.4.6 Bootstrapping-prosessi

CoAP-noodien turvallisesta bootstrapping-prosessin toteuttamisesta ei ole selvää sopimusta, varsinkaan tilanteessa, jossa laitteita ei ole varustettu esiasennetulla avainmateriaalilla [6]. Noodit ovat erityisen haavoittuvia bootstrapping-prosessin aikana. Niillä ei ole tietoa verkostosta ja viestintäkumppaneista.

Uusien solmujen vahingossa lisääminen naapuriverkkoon on estettävä [6]. Vihamielisten noodien ei pitäisi pystyä muodostamaan yhteyttä verkkoon, johon ne eivät kuulu. Siten vaaditaan todennusmekanismi, joka varmistaa, että uudet solmut kommunikoivat nimetyn viestintäkumppanin kanssa. Viestintäavaimien ja konfigurointitietojen hankkimiseksi on perustettava suojattu kanava. Koska bootstrapping-prosessista ei ole sovittu, eikä sen toteuttamiseen ole yhteistä mallia, voi huonosti toteutettu prosessi vaarantaa CoAP-laitteiden verkon ja laitteet.

4.4.7 Heikko avainten generointi

Rajoittunutta noodia ei tulisi käyttää avainten luomiseen, sillä riittävän entropian aikaansaanti voi olla vaikeaa [39]. Tällöin avaimet tulisi tuottaa muualla ja lisätä noodeihin valmistuksen tai käyttöönoton aikana.

Rajoittuneet noodit voivat myös olla asennettuina alttiisiin ympäristöihin, jolloin avaimet voivat olla saatavissa rajoittuneesta laitteista. Jaettu avain on tällaisessa tilanteessa huono vaihtoehto, sillä sen vuotaminen altistaa koko avainta käyttävän ryhmän.

5 CoAP:n tietoturvatestaus

CoAP-protokollan tietoturvallisuuden testauksen tavoitteena on tuottaa tuloksia tutkielman analyysiosiota varten. Testaus koostuu kahdesta osiosta: testausasetelman luomisesta, sekä itse toteutetusta testistä. Tutkielman laajuudessa ei ole mahdollista testata kaikkia mahdollisia hyökkäyksiä, joten testaukseen on otettu kaksi edellisen luvun aliluvussa 4.4 esiteltyä hyökkäystä: spoofing ja yksinkertainen vahvistushyökkäys.

Spoofing on useiden muiden hyökkäysten osalta joko mahdollistajana tai tukemassa hyökkäystä. Koska sen rooli on keskeinen useissa hyökkäyksissä kuten esimerkiksi vahvistushyökkäyksessä ja protokollien välisissä hyökkäyksissä on sen testaaminen perusteltua CoAP:n tietoturvan arvioinnin tukemiseksi.

Vahvistushyökkäys on UDP-protokollan mahdollistama palvelunestohyökkäys, joka käyttää hyväkseen spoofingia. CoAP:n mahdollinen vahvistuskerroin on korkea ja näin ollen vahvistushyökkäys on keskeinen hyökkäysmetodi. Vahvistushyökkäys tuottaa suuren määrän liikennettä palvelunestohyökkäykselle tyypilliseen tapaan, jolloin hyökkäyksen vaikutus on myös helposti tarkasteltavissa ja osoitettavissa.

Ennen spoofingia ja vahvistushyökkäystä toteutetaan myös CoAP-liikenteen kaappaaminen. Liikenteen kaappaaminen ei ole varsinainen hyökkäys, mutta on ensimmäinen vaihe spoofingin ja vahvistushyökkäyksen toteuttamiseksi. Liikennettä kaapataan sekä suojaamattomasta, että DTLS-suojatusta liikenteestä, jolloin myös suojauksen vaikutusta liikenteeseen voidaan tarkastella.

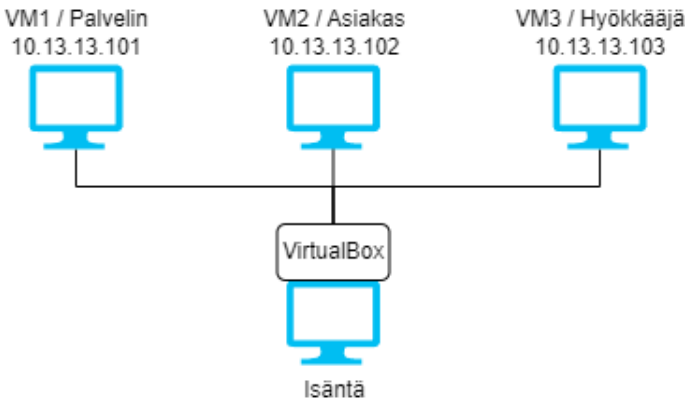
Testauksessa toteutetaan siis liikenteen kaappaus, spoofing ja yksinkertainen vahvistushyökkäys. Lisäksi tuotetaan, kaapataan ja tarkastellaan DTLS-suojattua CoAP-liikennettä. Näiden testausten tulosten perusteella yhdessä teoriatarkastelun kanssa, voidaan arvioida hyökkäysten toteutettavuutta, turvallisuustilojen vaikutusta, sekä protokollan tietoturvallisuutta yleisesti.

5.1 Testausasetelma

Testausasetelmalla tarkoitetaan tämän tutkielman yhteydessä CoAP-protokollan tietoturvatestauksen testausasetelmaa, joka sisältää ympäristön, käytettävät ohjelmat, sekä niiden konfiguraation ja valmistelun siten, että asetelmassa voidaan toteuttaa protokollaan kohdistuvaa tietoturvatestausta. Testausasetelman luomisvaiheessa kartoitettiin erilaisia mahdollisuuksia testausympäristön luomiseksi, sekä eri työkalujen ja sovellusten kyvykkyyksiä ja käytettävyyttä testauksen tavoitteiden näkökulmasta. Tavoitteena asetelman rakentamisessa oli hyödyntää yleisesti saatavilla olevia ilmaisia sovelluksia, ohjelmistoja ja työkaluja. Mahdollisia valmiita palvelin- ja asiakassovelluksia oli tarjolla useita ja näiden osalta pyrittiin mahdollisimman valmiiseen ja helppokäyttöiseen toteutukseen, joka ei vaatisi suuria määriä konfigurointia tai koodin tarkastelua ja muokkaamista, ennen kuin varsinaista testausta voitaisiin toteuttaa.

Koska testauksen kohteena oli vain protokollan tietoturvaominaisuudet, eikä esimerkiksi tietty protokollaa hyödyntävä tuote tai teknologia, valittiin testausympäristön toteutustavaksi virtuaalinen ympäristö. Virtuaalinen ympäristö yksinkertaisti ympäristön suunnittelua, rakentamista ja mahdollisti muutosten helpon toteutuksen ympäristössä. Lisäksi virtuaalinen ympäristö mahdollistaa paremman toistettavuuden testaukselle, sekä yksinkertaisemman lähtökohdan mahdolliselle testausasetelman myöhemmälle käytölle ja jatkokehitykselle.

Testausympäristö rakennettiin VirtualBox-alustalle ja sen arkkitehtuuri on esitetty kuvassa 5.1. Ympäristö sisälsi kolme virtuaalikonetta (palvelin, asiakas ja hyökkääjä), jotka olivat keskenään yhteydessä VirtualBoxin sisäisessä verkossa. Testauksen aikana virtuaalikoneiden verkkoyhteydet ympäristön ulkopuolelle oli poistettu käytöstä, jolloin kommunikointi oli mahdollista vain ympäristön sisällä.



Kuva 5.1: Testausympäristön arkkitehtuuri

Virtuaalikoneissa käytettiin Lubuntu- (palvelin ja asiakas) ja Kali Linux -käyttöjärjestelmiä (hyökkääjä), sekä useita eri työkaluja ja sovelluksia. Käytetyt sovellukset on listattu taulukossa 5.1.

Taulukko 5.1: Testauksessa käytetyt työkalut ja sovellukset

Työkalu/sovellus	**Tarkoitus**	**Laite**
Californium plugtest	CoAP-palvelin (konttitoteutus)	Palvelin
Docker	Palvelinkontin ajaminen	Palvelin
Copper for Chrome	CoAP-asiakas (Chrome-selaimen lisäosa)	Asiakas
Chromium	Selain Copper for Chrome -lisäosalle	Asiakas
Go-CoAP	DTLS-suojatun CoAP-liikenteen tuottaminen	Hyökkääjä
Wireshark	CoAP- ja DTLS-liikenteen kaappaaminen	Hyökkääjä
Scapy	CoAP-viestien muokkaaminen, muodostaminen ja lähettäminen	Hyökkääjä

5.1.1 VirtualBox

VirtualBox on Oraclen monialustainen virtualisointisovellus [43]. VirtualBox mahdollistaa useiden eri virtuaalikoneiden ja käyttöjärjestelmien ajamisen samassa isäntäkoneessa (host). Virtuaalikoneiden luominen on yksinkertaista ja nopeaa, eikä useita erillisiä laitteita tarvita.

VirtualBoxia käytettiin testauksessa testausympäristön luomisessa. VirtualBox mahdollisti aiemmin kuvattujen virtuaalikoneiden luomisen samaan ympäristöön ilman, että erillisiä fyysisiä laitteita olisi tarvittu. Lisäksi hyödytään myös mahdollisuudesta eristää verkko VirtualBoxin sisäiseksi verkoksi. Vahvistushyökkäys on yksi palvelunestohyökkäyksen muoto ja sisäisessä verkossa ei ole vaaraa, että virheellisesti toteutettu hyökkäys voisi aiheuttaa haittaa verkon ulkopuolella.

5.1.2 Virtuaalikoneet

VirtualBoxiin asennettiin kolme eri virtuaalikonetta. Virtuaalikoneiden osalta pyrittiin keveyteen, sillä isäntäkoneen kapasiteetti oli rajallinen, eikä asiakkaan ja palvelimen sovellusten asettama suorituskykyvaatimus isäntäkoneelle ollut suuri. Asiakas- ja palvelinkoneisiin käyttöjärjestelmäksi asennettiin Lubuntu ja hyökkääjän koneeseen Kali Linux.

Lubuntu-projektin tavoitteena on tarjota kevyt mutta toimiva Linux-jakelu, joka perustuu vankkaan Ubuntu-pohjaan [41]. Lubuntu tarjoaa yksinkertaisen, mutta modernin sekä tehokkaan graafisen käyttöliittymän ja mukana tulee laaja valikoima sovelluksia. Lubuntun valinnan perusteena oli sen keveys ja tutkielman tekijän aiempi kokemus Ubuntu-käyttöjärjestelmästä.

Kali Linux on avoimen lähdekoodin Debian-pohjainen Linux-jakelu, joka on tarkoitettu edistyneeseen penetraatio- ja tietoturvatestaukseen [28]. Kali Linux tarjoaa yleisiä työkaluja, konfiguraatioita ja automaatioita. Se sisältää valmiina useita satoja työkaluja, joita voidaan käyttää erilaisissa tietoturvallisuuteen liittyvissä tehtävissä, kuten penetraatiotestauksessa, tietoturvatutkimuksessa, digitaalisessa forensiikassa, takaisinmallinuksessa, haavoittuvuuksien hallinnassa ja Red Team -testauksessa. Kali Linux valittiin, sillä se sisälsi valmiiksi testauksessa tarvittavat työkalut.

5.1.3 Työkalut ja ohjelmistot

Testauksessa käytettävien työkalujen ja sovellusten valinnassa lähtökohtana oli käyttää vapaasti käytettäviä valmiita sovelluksia. CoAP-asiakkaita ja -palvelimia oli tarjolla useita erilaisia. Valinnan perusteena oli mahdollisimman helppo käyttöönotto esimerkiksi valmiiden esimerkkipalvelimien muodossa, joita voitaisiin suoraan käyttää testaamisessa ilman, että koodia tarvitsisi erikseen muokata.

Californium plugtest ja Docker

Californium on Eclipsen Java-toteutus CoAP-protokollalle ja se täyttää RFC 7252:n vaatimukset [32]. Se on tarkoitettu IoT-pilvipalveluille ja sen painopiste on skaalautuvuudessa ja käytettävyydessä sulautettujen laitteiden kaltaisen resurssitehokkuuden sijaan. Silti Californium sopii myös sulautetuille JVM:ille. Coap test server sisältää Docker-tiedoston, joka käynnistää Californiumin CoAP-testipalvelimen (plugtestin)[29]. Coap test server valittiin käytettäväksi, sillä konttitoteutuksena se on helposti käyttöönotettavissa.

Docker on avoin alusta sovellusten kehittämiseen, toimittamiseen ja suorittamiseen [11]. Se tarjoaa mahdollisuuden pakata ja suorittaa sovelluksia löyhästi eristetyssä ympäristössä, jota kutsutaan kontiksi. Eristyksen ja suojauksen ansiosta näitä kontteja voidaan käyttää useita samanaikaisesti tietyllä isännällä. Kontit ovat kevyitä ja sisältävät kaiken sovelluksen suorittamiseen tarvittavan, joten se ei ole riippuvainen siitä mitä konfiguraatioita tai asennuksia isännässä on. Kontti toimii samalla tavalla kaikissa ympäristöissä.

Copper for Chrome ja Chromium

Copper for Chrome (Copper4Cr) on CoAP-asiakassovellus, joka toimii Chrome-selaimen lisäosana [21]. Copper4Cr:n avulla voidaan lähettää CoAP-viestejä eri palvelimille osoitteen perusteella.

Nykyiset Chrome-selaimen versiot eivät enää tue Chrome-sovelluksia (Chrome App), jota Copper4Cr:n toteutuksessa on hyödynnetty. Näin ollen sen käyttämiseksi täytyy käyttää vanhempia versioita. Chromen vanhemmat versiot eivät ole luotettavasti saatavilla, joten vanhojen versioiden osalta tukeuduttiin Chromium-selaimen vanhoihin versioihin, jotka ovat virallisista lähteistä saatavilla.

Chromium on avoimen lähdekoodin projekti ja selaimen lähdekoodi [12]. Lähdekoodi on käännettävissä täysin toimivan selaimen saamiseksi. Google käyttää tätä lähdekoodia, lisää Googlen nimen ja logon, sekä automaattisen päivitysjärjestelmän ja kutsuu tätä selainta Google Chromeksi. Koska Chrome selain pohjautuu Chromiumiin, voidaan vanhaa Chromium-versiota käyttämällä saada tuki Chrome-sovelluksille ja käyttää Copper4Cr sovellusta.

Go-CoAP

Go-CoAP on sovellus CoAP-sovellutusten toteuttamiseen Go-ohjelmointikielellä [30]. Sen esimerkkitoteutukset CoAP-asiakkaasta ja -palvelimesta ovat helposti käyttöö-

notettavissa. Se sisältää myös DTLS-tuen ja toteuttaa CoAP:n korkeammat turvallisuustilat (PreSharedKey, RawPublicKey ja sertifikaatit). Esimerkkisovellukset ovat kuitenkin hyvin yksinkertaisia, eivätkä tarjonneet sellaisenaan riittävää laajuutta testausta varten.

Go-CoAP toteuttaa kuitenkin DTLS-salauksen, johon Copper4Cr ei kykene. Näin ollen sitä käytettiin DTLS-suojatun liikenteen tuottamiseen. Go-CoAP:n testipalvelin, joka toteuttaa DTLS:n PSK:n periaatteella jää käynnistämisensä jälkeen kuuntelemaan UDP-porttia 5668. Asiakas pyrkii käynnistämisensä jälkeen ottamaan yhteyden tuohon porttiin. Yhteyden onnistuessa suoritetaan yhteydenmuodostus/kättely, jonka jälkeen CoAP-liikenne lähetetään salattuna.

Wireshark

Wireshark on maailman johtava verkkoprotokolla-analysaattori [13]. Wiresharkilla voidaan kaapata verkon liikennettä, sekä tarkastella ja analysoida sitä. Kaapattua liikennettä voidaan tarkastella pakettikaappaustiedostostoista (pcap), joko Wiresharkilla tai muulla sopivalla ohjelmalla, kuten esimerkiksi Scapylla.

Testauksessa Wiresharkia käytetään seuraamaan sekä kaappaamaan asiakkaan ja palvelimen välistä liikennettä. Liikenteestä saadaan osoitetietoja, joita voidaan hyväksikäyttää spoofingissa tai edelleen vahvistushyökkäyksessä. Kaapattu liikenne voidaan tallentaa ja sitä voidaan tarkastella myöhemmin Wiresharkissa. Kaapattu liikenne viedään testauksessa edelleen Scapy-työkalulle, jolla sitä voidaan uudelleenlähettää tai muokata.

Scapy

Scapy on Python-pohjainen interaktiivinen paketinkäsittelyohjelma ja kirjasto [38]. Se pystyy väärentämään tai purkamaan useiden protokollien paketteja, lähettämään ja kaappaamaan niitä, sekä tallentamaan tai lukemaan niitä pcap- eli pakettikaappaustiedostojen avulla. Lisäksi se pystyy yhdistämään pyyntöjä ja vastauksia, sekä paljon muuta. Se on suunniteltu mahdollistamaan nopea pakettiprototyyppien luominen käyttämällä toimivia oletusarvoja. Scapy pystyy käsittelemään helposti useita tehtäviä, kuten skannausta, jäljitystä, luotausta, yksikkötestejä, hyökkäyksiä tai verkon löytämistä.

Scapylla voidaan myös käsitellä ja luoda CoAP-viestejä. CoAP-viestien rakenteen tunnistaminen vaatii scapy.contrib.coap -lisäosan tuomisen Scapyyn, joka lisää siihen CoAP-kerroksen. Koska kaikkia viestin kenttiä voidaan muokata Scapylla, on

myös IP-lähdeosoitteen muokkaaminen mahdollista, mikä mahdollistaa spoofingin. Testauksessa Scapya käytetään CoAP-viestien muokkaamiseen, luomiseen ja lähettämiseen.

5.2 Testauksen toteutus

Testaustilanteen lähtökohtana on oletus, että hyökkääjällä on jotakin kautta pääsy verkkoon, jossa CoAP-protokollaa viestinnässään käyttävät laitteet ovat. Hyökkääjä pystyy seuraamaan verkon liikennettä ja lähettämään sinne paketteja. Verkossa ei ole käytössä alemman kerroksen suojauksia, kuten esimerkiksi IPseciä.

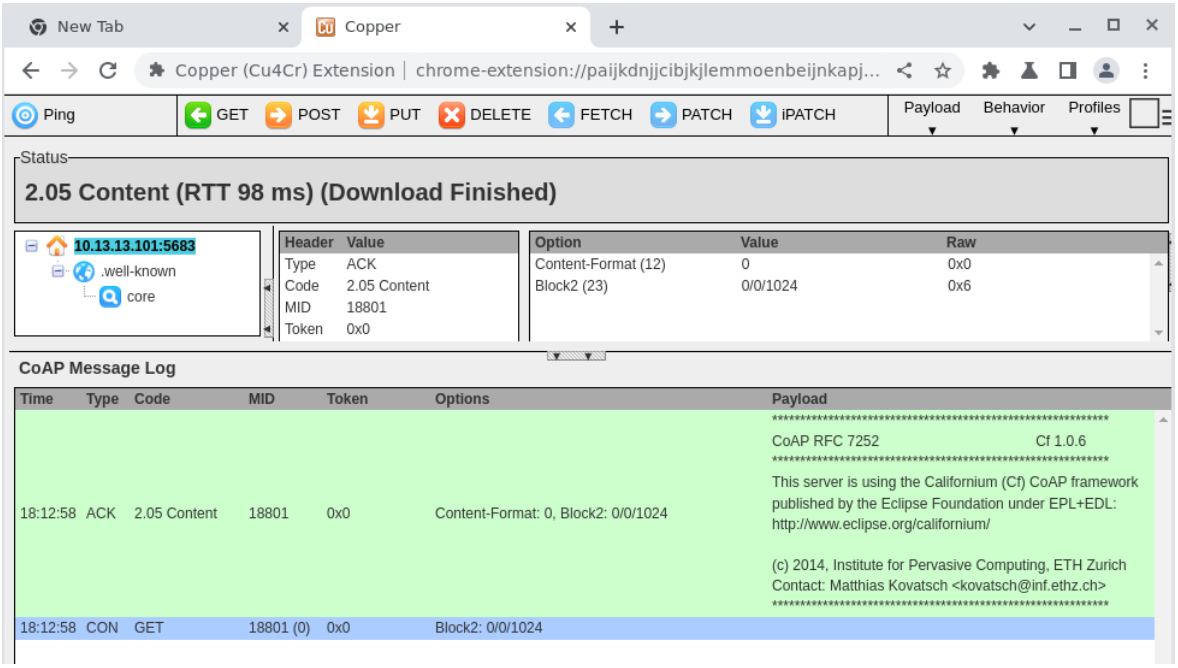
Ensimmäisenä testauksessa toteutetaan verkon liikenteen kaappaaminen. Kaapatusta liikenteestä hyökkääjän tulisi saada selville osoitetiedot (MAC- ja IP-osoitteet), joita voitaisiin käyttää hyväksi spoofingissa ja vahvistushyökkäyksessä. Lisäksi hyökkääjä voisi salaamattomasta liikenteestä tarkastella CoAP-liikenteen hyötykuorman sisältöä ja sen sisällöstä riippuen saada mahdollisesti käsiinsä luottamuksellista tietoa.

Kaappauksen jälkeen toteutetaan spoofing. Kaapatusta liikenteestä pitäisi olla saatavilla tarvittavat osoitetiedot, sekä valmis viesti, joka voidaan lähettää uudelleen verkkoon esimerkiksi Scapylla. Spoofing toteutetaan Scapylla toistamalla aiemmin kaapattu liikenne verkkoon, sekä lähettämällä Scapylla luotu, väärät osoitetiedot sisältävä viesti verkkoon. Lopputuloksena palvelimelle tulisi päätyä viestejä, joiden lähtöosoite (asiakkaan osoite) ei vastaa lähettäjää, joka on tässä tapauksessa hyökkääjä. Koska palvelin käyttää vastauksessaan viestissä olevaa lähdeosoitetta, vastausten tulisi mennä väärään osoitteeseen (asiakkaalle).

Spoofingin jälkeen toteutetaan yksinkertainen vahvistushyökkäys. Vahvistushyökkäyksessä käytetään hyväksi kaapatun liikenteen tietoja ja palvelimelle lähetettävän viestin luomisessa käytetään Scapya vastaavasti kuin spoofingissa. Yksinkertaisessa vahvistushyökkäyksessä lähetetään samaa GET-pyynnön sisältävää viestiä useita kertoja palvelimelle. Palvelimen vastausten tulisi kohdistua spooffatun IP-osoitteen omaavalle asiakkaalle ja kuormittaa tätä merkittävästi, mahdollisesti toteuttaen palveluneston.

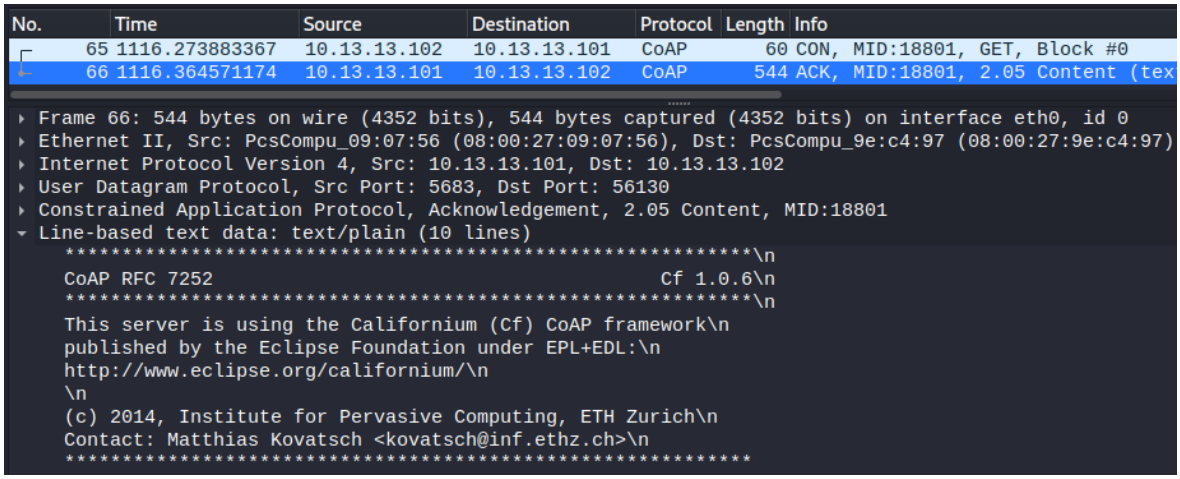
5.2.1 Verkon liikenteen kaappaus

Verkon liikenteen kaappaaminen toteutettiin Wireshark-ohjelmalla. Palvelimen ja asiakkaan välille tuotetaan liikennettä tekemällä GET-pyyntö asiakkaalta palvelimelle. Kaapattavan liikenteen aikaansaamiseksi tehtiin asiakkaalta GET-pyyntö palvelimelle. Asiakkaan pyyntö ja sitä vastaava palvelimen vastaus Copper for Chrome -sovelluksessa on esitetty kuvassa 5.2.



Kuva 5.2: GET-pyyntö ja palvelimen vastaus asiakkaan näkymässä

Pyynnön aikana Wiresharkilla kaapattiin verkon liikennettä, jolloin asiakkaan pyyntö ja vastaus saatiin kaapattua. Kaapattu liikenne on esitetty kuvassa 5.3, jossa näkyvät pyyntö- ja vastauspaketit. Kuvassa aktiivisena (paketti numero 66) on palvelimen vastaus, jolloin kuvan alaosassa näkyy kyseisen vastauksen tarkempi sisältö.



Kuva 5.3: GET-pyyntö ja vastaus Wiresharkin kaapatussa liikenteessa

Wireshark-kaappauksesta kuvassa 5.3 voidaan huomata, että palvelimen vastauksen kaikki sisältö, mukaan lukien hyötykuorma, on näkyvissä, kun turvallisuustila on NoSec ja DTLS-suojaus ei ole käytettävissä. Nähtävissä on myös osoitetiedot ja portit, joita liikennöinnissä käytettiin.

5.2.2 Spoofing

Kaapatun liikenteen tietoja hyväksikäyttämällä voidaan toteuttaa spoofing, jossa hyökkääjä lähettää verkkoon CoAP-viestejä, joiden lähtöosoitteena on asiakkaan osoite. Kuvassa 5.4 on edellisessä kohdassa kaapattu GET-pyyntö näkyvissä Scapyssa.

Kuva 5.4: Kaapattu GET-pyyntö tuotuna Scapyyn

Edellisen viestin tietoja hyväksikäyttämällä voidaan luoda uusi CoAP-viesti tai lähettää kaapattu viesti uudelleen. Scapyssa voidaan myös luoda vastaava paketti käyttäen hyväksi kaapatusta liikenteestä saatuja osoitetietoja. Paketti voidaan luoda seuraavalla komennolla Scapyssa:

```
>>> get_crafted = Ether(dst='08:00:27:09:07:56', src='08:00:27:9e:
    c4:97')/IP(src='10.13.13.102', dst'10.13.13.101')/UDP(sport
    =56130, dport=5683)/CoAP(ver=1, type'CON', tkl=0, code'GET',
    msg_id=999, options[(23, b'\x06')]
```

Luotu paketti esitetty kuvassa 5.5. Scapy luo automaattisesti tarvittavat arvot UDP- ja IP-tarkistussummille (chksum), sekä pakettien pituuksille (len), jolloin näi-

tä ei tarvitse paketin luomisessa huomioida erikseen. Luodun paketin sisältö on CoAP:n viestitunnusta (msg_id) lukuun ottamatta sama, kuin suoraan kaappauksesta tuodussa paketissa. Muuttamalla viestitunnusta, voidaan lähetetty viesti erottaa alkuperäisestä ja todeta, että hyökkääjällä on mahdollisuus muokata paketin sisältöä haluamakseen.

```
>>> get_crafted = Ether(dst='08:00:27:09:07:56', src='08:00:27:9e:c4:97')/IP(src=
... : '10.13.13.102', dst='10.13.13.101')/UDP(sport=56130, dport=5683)/CoAP(ver=1,
... :  type='CON', tkl=0, code='GET', msg_id=999, options=[(23, b'\x06')])
>>> get_crafted.show2()
###[ Ethernet ]###
  dst       = 08:00:27:09:07:56
  src       = 08:00:27:9e:c4:97
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 35
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = udp
     chksum    = 0x4be5
     src       = 10.13.13.102
     dst       = 10.13.13.101
     \options   \
###[ UDP ]###
        sport     = 56130
        dport     = 5683
        len       = 15
        chksum    = 0xc482
###[ CoAP ]###
           ver       = 1
           type      = CON
           tkl       = 0
           code      = GET
           msg_id    = 999
           token     = ''
           options   = [(23, b'\x06')]
           paymark   = ''

>>>
```

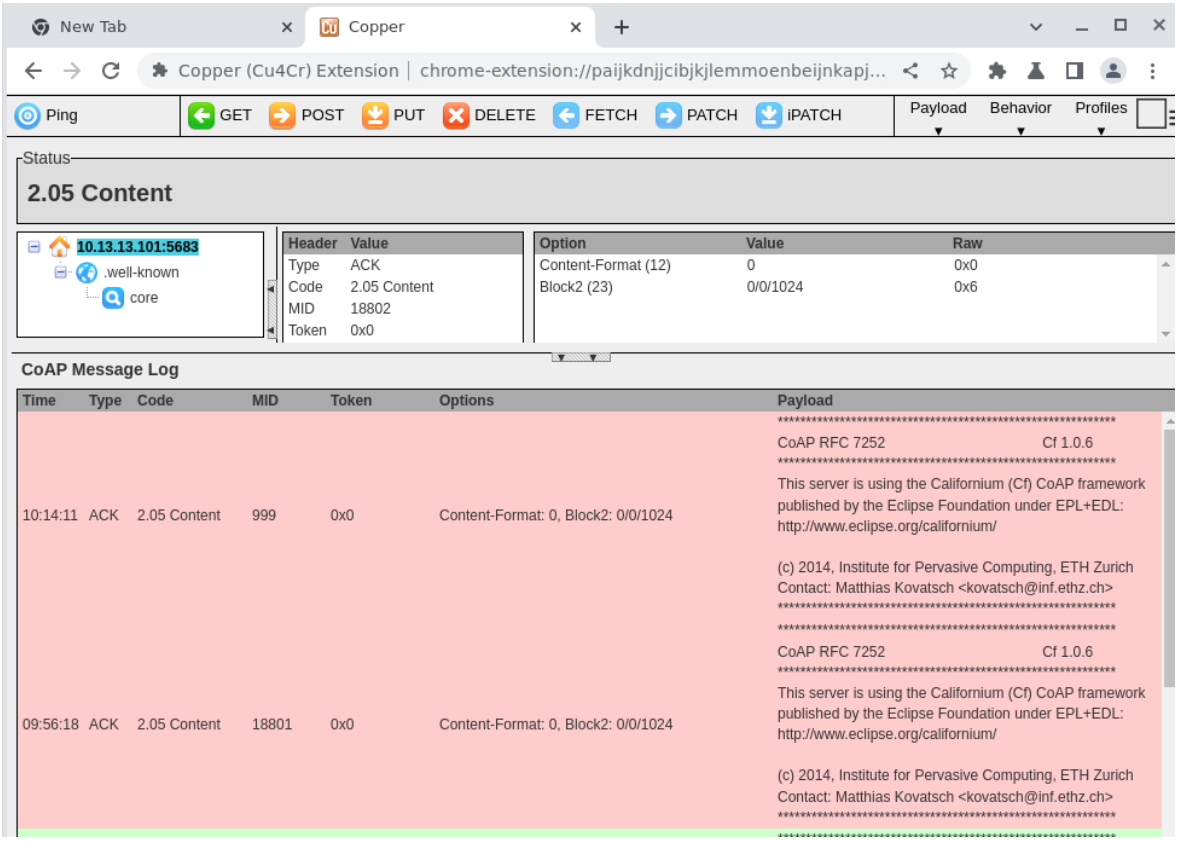
Kuva 5.5: Scapyssa luotu GET-pyyntö

Hyökkääjä lähettää molemmat paketit seuraavilla Scapyn komennoilla:

```
>>> sendp(get,iface'eth0')
>>> sendp(get_crafted,iface'eth0')
```

Paketit päätyvät palvelimelle, joka vastaa paketin osoitetietojen perusteella asiakkaalle. Asiakas vastaanottaa kaksi vastausta palvelimelta kuvassa 5.6. Kuvasta voidaan tunnistaa alkuperäinen viesti, joka on uudelleenlähetetty, sillä viestitunnus on

sama (18801). Scapylla luodussa paketissa viestitunnukseksi asetettiin 999, mikä näkyy myös kuvassa. Asiakkaan lokissa viestit näkyvät punaisella taustalla, sillä asiakas tunnistaa viestit vääriksi, sillä pyyntöön 18801 vastattiin jo aiemmin ja pyyntöä 999 se ei ollut tehnyt. Palvelin ei ole tästä kuitenkaan tietoinen, vaan vastaa pyynnönmukaisesti paketissa ilmoitettuun lähtöosoitteeseen.



Kuva 5.6: Vastaus toistettuun ja Scapyssa luotuun GET-pyyntöön

5.2.3 Vahvistushyökkäys

Edellisessä alaluvussa toteutettiin onnistunut spoofing. Vahvistushyökkäyksessä voidaan hyödyntää samoja menetelmiä. Yksinkertaisen vahvistushyökkäyksen tavoitteena on pyytää väärennetystä osoitteesta palvelinta lähettämään mahdollisimman paljon dataa, joka ohjautuisi siis IP-osoitteen oikealle omistajalle. Hyökkääjän kannalta haetaan siis mahdollisimman suurta vahvistusta, eli pienintä mahdollista pyynnön kokoa ja suurinta mahdollista pyynnön aiheuttamaa vastausta palvelimelta, joka ohjautuisi kohteelle.

Asiakkaan spoofingin perusteella vastaanottamat viestit ovat sen mielestä virheellisiä perustuen väärään MID-arvoon (message ID) tällöin asiakas ei kuittaa paketteja vastaanotetuksi tai pyydä seuraavia osia lähetettäväksi isommista resursseista. Näin ollen suurin mahdollinen vastaus, jonka asiakas voi vastaanottaa yhtä pyyntöä kohti on yhteen pakettiin mahtuva tietomäärä, eli tässä tapauksessa 1024 tavua. Tätä isommat viestit jaetaan paloihin (block), eikä seuraavia paloja lähetetä ennen kuittausta, jota ei siis tässä tapauksessa tule.

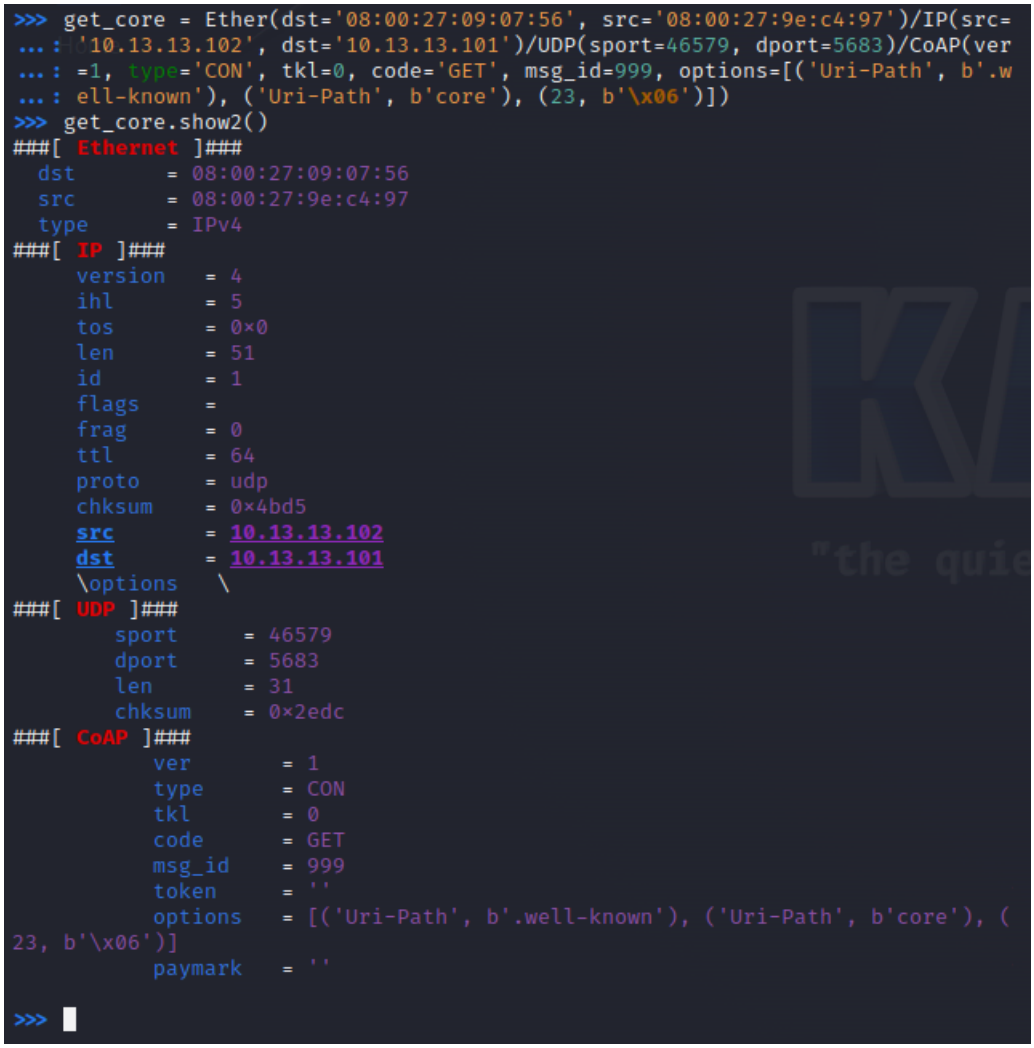
Kuten aiemman kuvan 5.3 Wireshark-kaappauksesta voidaan nähdä, on aiemmin tehtyjen GET-pyyntöjen koko 60 tavua ja vastauksen koko 544 tavua. Tämä ei vielä maksimoi vahvistusvaikutusta, jolloin palvelimelta täytyy pyytää suurempaa vastausta. CoAP-palvelimissa on RFC:n mukaisesti yleensä toteutettuna URI:t *.well-known* ja sen alla *core*. Kun GET-pyyntö kohdistetaan URI:in *core*, palauttaa se kaiken sisällön, mitä palvelin tarjoaa. Koska sisältöjä on yleensä paljon, voidaan olettaa, että useissa palvelimissa GET-pyyntö URI:in *core* palauttaa maksimikokoisen vastauksen. *Core* on siis hyvä kohde hyökkääjälle, joka etsii mahdollisimman suurta vastausta.

Kuten edellisessä alaluvussa, luodaan Scapyn komennolla core:en kohdistuvan GET-pyynnön sisältävä paketti. Paketti luodaan komennolla (UDP:n lähtöportti on nyt eri, sillä ajankohta on eri kuin aiemmin ja asiakassovellus on vaihtanut käyttämänsä portin):

```
>>> get_crafted = Ether(dst='08:00:27:09:07:56', src='08:00:27:9e:
   c4:97')/IP(src='10.13.13.102', dst'10.13.13.101')/UDP(sport
   =46579, dport=5683)/CoAP(ver=1, type'CON', tkl=0, code'GET',
   msg_id=999, options[('Uri-Path', b'well.known'),('Uri-Path', b'
   core'), (23, b'\x06')]
```

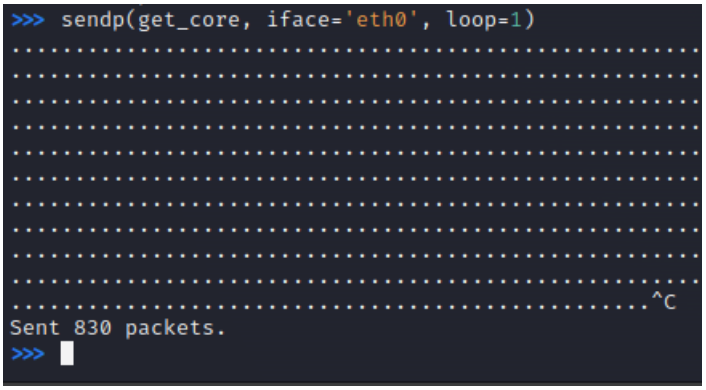
Luotu paketti näkyy kuvassa 5.7. Paketti voidaan nyt lähettää palvelimelle, joka vastaa asiakkaalle URI:n *core* sisällöllä. Koska vahvistushyökkäyksellä pyritään kuormittamaan kohdetta ja toteuttamaan palvelunesto, tulee viestejä lähettää useita. Hyökkääjä aloittaa jatkuvan viestien lähettämisen komennolla:

```
>>> sendp(get_core,iface'eth0', loop=1)
```

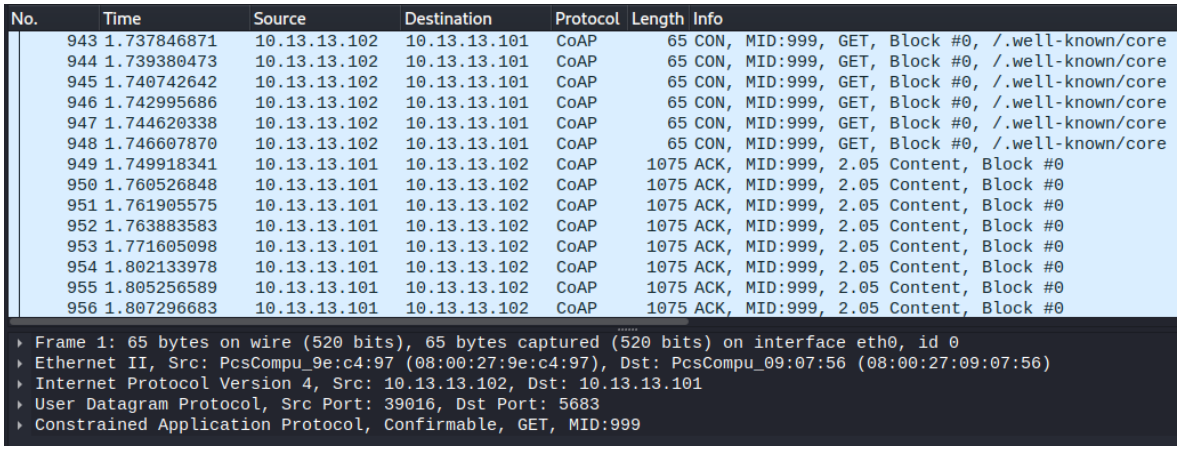


Kuva 5.7: Scapyssa vahvistushyökkäystä varten luotu GET-pyyntö URI:in core

Lähetys katkaistaan manuaalisesti siten, että hyökkääjä ehtii kuvan 5.8 mukaisesti lähettää 830 pakettia (CoAP-viestiä). Lähetetyt viestit kaapataan Wiresharkilla, jonka kaappaus näkyy kuvassa 5.9. Kuvasta huomataan, että GET-pyynnön sisältävän paketin koko on 65 ja vastauspakettien koko 1075 tavua. Kuvan kaappauksessa paketti numero 1 olisi ensimmäisenä lähtenyt GET-pyyntö. Kuvassa on näkyvissä viimeinen GET-pyyntö (numero 948), joka lähtee noin 1,75 sekuntia ensimmäisen pyynnön jälkeen. Kaappauksessa on myös muutamia vastausviestejä ennen tätä ajankohtaa, mutta pääosa paketeista on tuossa kohtaa vasta lähdössä palvelimelta.



Kuva 5.8: Scapyn silmukassa lähettämät GET-pyynnöt

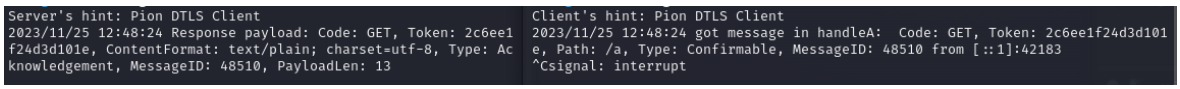


Kuva 5.9: Kaappaus toistetuista GET-pyynnöistä

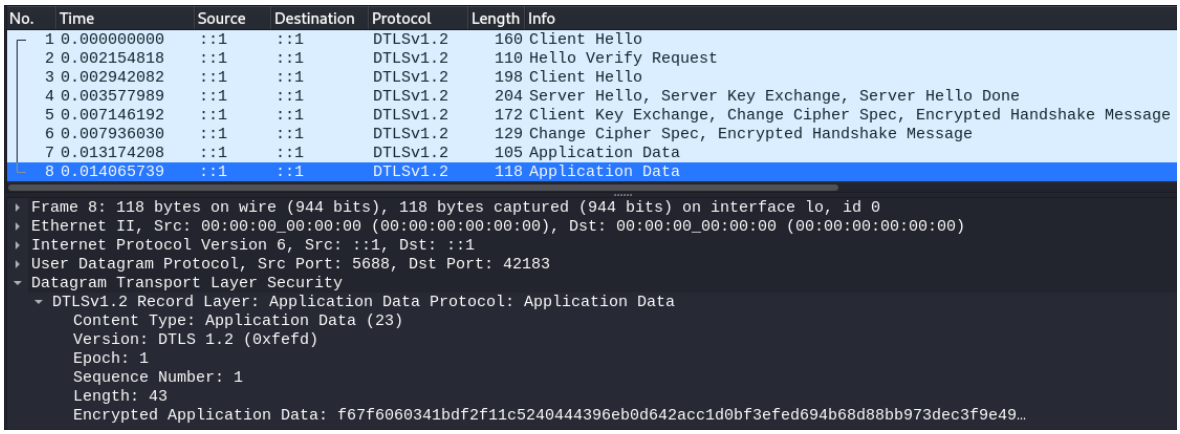
5.2.4 DTLS-suojattu liikenne

DTLS-suojattua liikennettä tuotettiin testausta varten hyökkääjän Kali-koneella. Go-CoAP:n sellaisenaan käytetyillä esimerkkitoteutuksilla liikenne on koneen sisäistä, jolloin sen kaappaaminen on yksinkertaisinta samalta koneelta. Go-CoAP:sta käynnistettiin esimerkkitoteutus PreSharedKey-turvallisuustilaa käyttävästä palvelimesta ja asiakkaasta. Palvelin jää käynnistyessään kuuntelemaan porttia 5688. Asiakas lähettää käynnistyessään Hello-viestin palvelimelle, jonka jälkeen suoritetaan yhteydenmuodostus. Yhteydenmuodostuksen jälkeen asiakas lähettää salatun CoAP-protokollan GET-pyynnön ja palvelin vastaa tähän. Asiakkaan sekä palvelimen tulosteet ovat näkyvissä kuvassa 5.10 ja tästä liikennöinnistä Wiresharkilla kaapatut

paketit kuvassa 5.11.



Kuva 5.10: Go-CoAP asiakkaan ja palvelimen tulosteet



Kuva 5.11: Kaapattu DTLS liikenne

Kuten Wiresharkin kaappauksesta voimme nähdä, ei CoAP-liikenne ole enää näkyvissä, vaan kaappauksessa näkyy vain DTLS paketteja. Kahdessa viimeisessä paketissa siirtyy CoAP-liikennettä, joista jälkimmäisen CoAP-datan alkuosa on näkyvissä salattuna kohdassa "Encrypted Application Data". Palvelin vastaa ainoastaan DTLS-salattuun liikenteeseen, jolloin se hylkää salaamattoman CoAP-liikenteen.

5.3 Tulokset

Tässä aliluvussa tarkastellaan toteutetun testauksen tulokset. Tulokset käsitellään aliluvuittain testausta vastaavassa järjestyksessä. Kokonaisuudessaan testaus toteutui onnistuneesti ja jokaisesta testausvaiheesta saatiin tarkastelukelpoisia tuloksia.

5.3.1 Liikenteen kaappaus

Liikenteen kaappauksessa Wiresharkilla onnistuttiin kaappaamaan koko liikenne CoAP-asiakkaan ja palvelimen välillä. Suojaamattomasta liikenteestä olivat näh-

tävillä kaikki CoAP-viestien tiedot, mukaan lukien osoitetiedot ja hyötykuorma. Kaappaamisen toteutus oli yksinkertaista, eikä vaatinut erityisiä valmisteluja.

Hyökkääjän päästessä verkkoon, jossa on CoAP-liikennettä, on liikenteen kaappaaminen ja seuraaminen siis hyvin yksinkertaista sopivalla ohjelmistolla, kuten esimerkiksi testauksessa käytetyllä Wiresharkilla. Suojaamaton CoAP-liikenne kaikkine tietoineen, mukaan lukien osoitetiedot ja hyötykuorma, on suoraan luettavissa kaappauksesta sellaisen hyökkääjän toimesta, joka pystyy kaappaamaan verkon liikennettä. Kaapatun liikenteen tiedot mahdollistavat erilaisten hyökkäyksien, kuten spoofingin ja vahvistushyökkäyksen valmistelun. Lisäksi siirrettävien tietojen luottamuksellisuus vaarantuu, mikäli luottamuksellista tietoa siirretään suojaamattomassa verkossa.

5.3.2 Spoofing

Spoofing onnistuttiin toteuttamaan uudelleenlähettämällä yksittäinen viesti aiemmin kaapatusta liikenteestä sekä luomalla ja lähettämällä uusi viesti kaapatun liikenteen tietoja hyväksikäyttämällä. Uuden viestin luominen, sekä molemmat lähetykset toteutettiin Scapylla. Molemmat viestit päätyivät palvelimelle, joka käsitteli niitä normaalina liikenteenä. Koska molemmissa viesteissä lähdeosoite oli asiakkaan osoite, vastasi palvelin näiden viestien perusteella asiakkaalle, vaikka viestien oikea lähettäjä oli tässä tapauksessa hyökkääjä.

Testauksen perusteella spoofingin toteuttaminen on yksinkertaista. Pelkkä yksittäisen viestin uudelleenlähetys palvelimelle kaapatusta liikenteestä toimii spoofingin tavoin, kun palvelin vastaanottaa viestin ja lähettää vastauksen alkuperäisen lähdeosoitteen mukaisesti. Liikenteen tietojen perusteella väärennetyillä osoitetiedoilla varustettujen uusien viestien luominen on myös yksinkertaista. Testauksessa lähetettiin yksinkertainen GET-pyyntö, mutta viestin tyyppi ja sisältö voitaisiin valita mielivaltaisesti hyökkääjään toimesta. Näitä viestejä voivat olla esimerkiksi Reset- ja ACK-viestit tai vastauksen hyötykuorman väärentäminen aliluvussa 4.4 esitellyllä tavalla laitteiden toiminnan häiritsemiseksi.

Sekä spoofingin, että uudelleenlähetyksen onnistumisen taustalla on NoSec-turvallisuustila, sekä CoAP:n käyttämä UDP-protokolla, jossa ei ole erillistä yhteydenmuodostusta. DTLS:n tulisi suojata CoAP-liikenne, jolloin sen mukaiset pyynnöt eivät pääsisi läpi ilman muodostettua DTLS-yhteyttä. Myös kuljetuskerroksella vaadittava yhteydenmuodostus voisi estää spoofingin, sillä CoAP-viestejä ei voitaisi lähettää ilman onnistunutta yhteydenmuodostusta, mikä ei ole mahdollista kättely-

viestien ohjautuessa asiakkaalle hyökkääjän sijaan.

5.3.3 Vahvistushyökkäys

Vahvistushyökkäys onnistuttiin toteuttamaan siten, että asiakkaaseen kohdistui suuri määrä liikennettä palvelimelta hyökkääjän lähettämien spooffattujen GET-pyyntöjen perusteella, jotka sisälsivät lähdeosoitteenaan asiakkaan osoitteen. GET-pyynnön sisältävä paketti oli kooltaan 65 tavua ja vastauspaketti 1075 tavua (paketti sisältää CoAP:n lisäksi myös UDP-, IP- ja Ethernet-kerroksen tietoja, jolloin CoAP-paketin maksimikoko 1024 ylittyy). Näin olleen vahvistuskertoimeksi x saadaan

$$x = \frac{1075}{65} = 16,53 \tag{5.1}$$

Koska Scapyllä lähetettiin 830 pakettia, oli hyökkääjän aiheuttaman liikenteen koko

$$k_h = 830 \times 65 = 53950 \tag{5.2}$$

eli noin 54 kt. Tällöin asiakkaalle lähetettävän liikenteen koko vahvistuskertoimella laskettuna on

$$k_a = 830 \times 65 \times 16,5 = 890175 \tag{5.3}$$

eli noin 890 kt.

Liikenteen koon lisäksi voidaan tarkastella myös tiedonsiirtonopeuksia. Tiedonsiirtonopeus riippuu kuitenkin useista asioista, kuten hyökkääjän kyvystä tuottaa paketteja, verkon nopeudesta ja palvelimen kyvystä tuottaa vastauspaketteja. Näin ollen tiedonsiirtonopeuksien osalta voidaan laskea vain suuntaa antavia lukuja.

Tiedonsiirtonopeuksien osalta luvut tarkoittavat paketit lähetettiin ajassa 1,75 sekuntia eli tiedonsiirtonopeus asiakkaalta hyökkääjälle on

$$n_h = \frac{53950 \times 8}{1,75} = 246628,57 \tag{5.4}$$

eli noin 246 kbit/s (kilobittiä sekunnissa). Vastaavasti palvelin voisi maksimissaan lähettää asiakkaalle nopeudella

$$n_pmax = \frac{890175 \times 8}{1,75} = 4069371,42 \tag{5.5}$$

eli noin 4,1 Mbit/s (megabittiä sekunnissa). On kuitenkin huomioitava, että palvelimen kapasiteetti lähettää näitä paketteja on rajallinen ja myös verkon nopeus vaikuttaa tähän, joten todellinen tiedonsiirtonopeus palvelimelta asiakkaan suuntaan jää pienemmäksi.

Yhteenveto vahvistushyökkäyksen luvuista on esitetty taulukossa 5.2 Edellisten kohtien perusteella yksinkertaisen vahvistushyökkäyksen toteuttaminen on myös helppoa. Saavutettu vahvistuskerroin 16,5 mahdollistaa hyökkääjälle tehokkaan tavan kohdistaa suuri määrä liikennettä haluamaansa kohteeseen.

Taulukko 5.2: Vahvistushyökkäyksen tunnusluvut

Liikenteen suunta	**Liikenteen määrä**	**Siirtonopeus**
Hyökkääjältä palvelimelle	54 kt	246 kbit/s
Palvelimelta asiakkaalle	890 kt	4,1 Mbit/s

5.3.4 DTLS-suojattu liikenne

DTLS-suojatun liikenteen kaappaus onnistui ja kaappausta tarkastelemalla voitiin todeta, ettei CoAP-liikenne ole enää näkyvissä liikenteessä, vaan on DTLS-liikenteen sisällä salattuna. DTLS:n toimintaperiaatteen mukaisesti yhteyttä palvelimeen ei myöskään voida muodostaa ilman turvallisuustilan vaatimia oikeita avaimia ja mahdollista sertifikaattia (sertifikaattitila).

Osoitetiedot ovat edelleen näkyvissä, mutta spoofingin osalta ne ovat hyödyttömiä sillä yhteydenmuodostus ei onnistu. Palvelimelle voidaan lähettää Hello-viesti, mutta mikäli lähdeosoite on väärä ei kättelyä voida jatkaa pitemmälle sillä palvelin vastaa osoitteen oikealle haltijalle. Koska vahvistushyökkäyksen toteutus vaatii onnistuneen spoofingin, ei myöskään yksinkertaista vahvistushyökkäystä ole mahdollista toteuttaa.

Tämän tutkielman laajuudessa ei testata heikkojen DTLS-avaimien murtamista, mikä voisi mahdollistaa viestien purkamisen ja kommunikoinnin palvelimen kanssa. Tämä voisi johtaa ainakin tietojen luottamuksellisuuden menettämiseen. Murretut avaimetkaan eivät kuitenkaan vaikuttaisi DTLS:n yhteydenmuodostukseen, joka estää spooffatun osoitteen käyttämiseen. Näin ollen muiden kuin NoSec-turvallisuustilan tulisi estää spoofing ja vahvistushyökkäykset.

6 Analyysi

Luvussa käsitellään ensimmäiseksi CoAP:n tietoturvauhkien analyysi noudattaen aiempien lukujen järjestystä, jonka jälkeen analysoidaan CoAP:n turvallisuustiloja. Viimeisessä aliluvussa yhteenvedetään analyysi CoAP-protokollan tietoturvallisuudesta. Analyysin perustana on aiemmissa luvuissa toteutettujen CoAP-protokollan tarkastelun ja tietoturvatestauksen tulokset.

6.1 Tietoturvauhat

Aliluvussa analysoidaan CoAP-protokollaan kohdistuvien tietoturvauhkien vaikutusta ja merkitystä. Tietoturvauhkia analysoidaan sekä suojaamattoman (NoSec), että DTLS-suojatun (PreSharedKey ja korkeammat turvallisuustilat) CoAP-protokollan näkökulmasta.

6.1.1 Spoofing

CoAP-protokollaan kohdistuva spoofing on aiemmassa tutkimuksessa todettu mahdolliseksi, johtuen UDP-protokollan ominaisuuksista. Testauksessa sen toteutus todettiin hyvin helpoksi, jos DTLS-suojaus ei ole käytössä. Spoofingin helppo toteutettavuus mahdollistaa ja tukee myös sitä käyttäviä kehittyneempiä hyökkäyksiä, mikä nostaa myös niiden toteutusmahdollisuuksia.

Turvallisuustilan nostaminen NoSec:iä korkeammaksi tuo DTLS-suojauksen ja sen mukana yhteydenmuodostuksen. Spoofingia ei voida toteuttaa tässä tutkielmassa käsitellyllä tavalla yhteydenmuodostuksen keskeytyessä. Spoofing voidaan siis estää käyttämällä korkeampaa turvallisuustilaa.

6.1.2 Vahvistushyökkäys

Vahvistushyökkäys ja CoAP:n korkea potentiaalinen vahvistuskerroin on tunnistettu aiemmassa tutkimuksessa. Testauksessa todettiin, että yksinkertainen vahvistushyökkäys on helposti toteutettavissa, kun suojaus ei ole käytössä. Testauksessa saavutettiin myös korkea vahvistuskerroin toteutetussa vahvistushyökkäyksessä, mikä

vahvistaa mainitun korkean potentiaalin korkealle kertoimelle. Helppo toteutettavuus sekä korkea vahvistuskerroin mahdollistavat hyökkääjälle tehokkaan keinon palvelunestoon yksittäisen kohteen osalta ja lisäksi palvelimen sekä koko verkon kuormittamisen suurilla liikennemäärillä.

Koska vahvistushyökkäys perustuu onnistuneeseen spoofingiin, ei myöskään vahvistushyökkäystä voida toteuttaa tutkielmassa kuvatulla tavalla NoSec:iä korkeamman turvallisuustilan ollessa käytössä. Vahvistushyökkäys voidaan siis estää käyttämällä korkeampaa turvallisuustilaa.

6.1.3 Protokollien väliset hyökkäykset

UDP-pohjaiset protokollat ovat alttiita protokollien välisille hyökkäyksille. Protokollien välinen hyökkäys voidaan yksinkertaisimmillaan toteuttaa lähettämällä CoAP-laitteelle väärän lähdeosoitteen sisältävä viesti, joten toteutustapa sama kuin testauksessa toteutetussa spoofingissa (erotuksena, että lähdeosoitteessa olisi toista protokollaa käyttävä palvelu, joka tulkitsisi CoAP-laitteen lähettämän viestin toisen protokollan sääntöjen mukaan). Näin ollen voidaan samoin perustein kuin spoofingin osalta todeta, että myös protokollien väliset hyökkäykset, ainakin yksinkertaiset sellaiset, ovat helppoja toteuttaa, kun suojausta ei ole toteutettu.

Protokollien väliset hyökkäykset vaikeutuvat, jos turvallisuustila on NoSec:iä korkeampi. Aiemmassa tutkimuksessa on todettu, että saman DTLS-yhteyden piirissä olevien protokollien välillä hyökkäykset ovat mahdollisia. Näin ollen esimerkiksi murretun avaimen omaava hyökkääjä, jolla on pääsy DTLS-yhteyteen voi mahdollisesti toteuttaa protokollien välisiä hyökkäyksiä näillä protokollilla.

6.1.4 Jäsennyshyökkäykset

CoAP:n yleisimmät haavoittuvuudet liittyvät aiemman tutkimuksen mukaan virheelliseen jäsennykseen. Virheellinen jäsennys avaa mahdollisuuden jäsennyshyökkäyksille. Koska IoT-laitteiden päivitys on usein haastavaa, on mahdollista, että myös CoAP:ia hyödyntävistä laitteista löytyy jäsennykseen liittyviä haavoittuvuuksia, joita ei ole korjattu. Koska haavoittuvuudet liittyvät tietyn sisältöisten viestien käsittelyyn, on niitä hyödyntävän hyökkääjän toimitettava tällainen viesti kohteelle. Kuten testausvaiheessa todettiin, on mielivaltaisen paketin luominen yksinkertaista ja lähettäminen suojaamattomassa verkossa helppoa. Näin ollen suojaamaton verkko on altis myös jäsennyshyökkäyksille.

Jäsennyshyökkäykset vaikeutuvat, jos NoSec:iä korkeammat turvallisuustilat ovat käytössä. Samoin kuin protokollien välisissä hyökkäyksissä, tarvitsee hyökkääjä tällöin DTLS-yhteyden, esimerkiksi murretun avaimen kautta.

6.1.5 Välimuistihyökkäykset

Välimuistihyökkäykset ovat välityspalvelimien vääränlaisen toteutuksen mahdollistamia hyökkäyksiä, joissa hyökkääjä pääsee käsiksi välimuistin tietoihin. Aiemmasta tutkimuksesta ei tunnistettu välimuistihyökkäysten tarkempaa käsittelyä CoAP:n osalta, eikä hyökkäys sisältynyt testattaviin hyökkäyksiin. Väärin konfiguroitu palvelin on kuitenkin aiemmassa tutkimuksessa tunnistettu riskiksi, joka koskee myös DTLS- ja IPsec-suojattua CoAP:ia, sillä välityspalvelimella nämä suojaukset puretaan. Tällöin hyökkääjällä voisi olla mahdollisuus päästä tietoon käsiksi välityspalvelimella.

6.1.6 Haavoittuvat prosessit

Bootstrapping-prosessi on todettu aiemmassa tutkimuksessa haavoittuvaksi. Erityisesti tämä korostuu tilanteessa, jossa avainmateriaalia ei ole olemassa valmiiksi vaan se täytyisi jakaa ja käyttöönottaa prosessin aikana. Aiemmasta tutkimuksesta löydettiin mainintoja bootstrapping-prosessin ongelmallisuudesta sekä joitakin esitettyjä ratkaisuja. Uhan tasoa ei kuitenkaan kyetty tarkemmin määrittämään aiemman tutkimuksen perusteella, eikä prosessin testausta suoritettu tässä tutkielmassa.

Bootstrapping-prosessiin liittyy myös toinen tunnistettu haavoittuva prosessi, heikko avainten generointi. Rajoittuneen laitteen kyky luoda vahvoja avaimia on yleensä rajoittunut, jolloin avaimen luominen ei tulisi olla laitteen tehtävä vaan se tulisi saada muualta tai olla esiasennettuna laitteessa. Heikko avain mahdollistaa sen murtamisen raa'an voiman menetelmällä ja murretun avaimen avulla laite voi turvallisuustilasta riippuen kommunikoida verkossa ja/tai päästä käsiksi kaapatun liikenteen tietoihin.

Bootstrapping-prosessin ja avainten generoinnin osalta todetaan, että ne sisältävät uhkia, jotka on mahdollista ainakin osittain kiertää käyttämällä ennalta jaettuja avaimia. Toisaalta tämä voi johtaa laitteiden käyttöönoton vaikeutumiseen ja verkkojen hallinnan monimutkaistumiseen.

6.2 Turvallisuustilat

CoAP:n turvallisuus rakentuu turvallisuustilojen päälle. NoSec:iä korkeampien turvallisuustilojen käyttäminen nostaa protokollan turvallisuutta, mutta myös vaadittavaa prosessointitehoa, mikä on IoT-laitteissa ja siten myös CoAP:ssa yleensä varsin rajallinen.

NoSec-tila ei nimensä mukaisesti tarjoa minkäänlaista suojausta, eikä sitä tule käyttää missään tilanteessa, jossa suojaukselle on tarve, jos suojausta ei ole toteutettu alemmilla kerroksilla. CoAP:n käyttämän UDP-protokollan ominaisuuksien takia NoSec on erityisen haavoittuva monille hyökkäyksille verrattuna muihin protokolliin ilman suojausta, jotka käyttävät kuljetuskerroksella TCP:tä. Testauksen perusteella NoSec-tilassa hyökkäyksien toteuttaminen CoAP-laitteiden verkossa on hyökkääjälle helppoa.

PreSharedKey-tilan ennalta jaetut symmetriset avaimet tarjoavat kevyimmän DTLS-suojauksen protokollalle. Heikkoudeksi jää mahdollisuus heikkoon avaimeen, joka on altis raa'an voiman hyökkäyksille, sekä mahdollisuus purkaa kaapattua liikennettä myöhemmin, mikäli avain saadaan murrettua. Toisaalta heikkojenkin avainten käytöllä tuodaan liikennöintiin mukaan UDP:sta puuttuva yhteydenmuodostus, jolloin spoofing ja vahvistushyökkäys voidaan estää.

RawPublicKey:n epäsymmetrinen avainpari ja ulkoisen mekanismin varmistus tuottavat vahvemman suojauksen. Luottamuksellisuuden suoja on vahvempi, sillä aiemman viestinnän salausta ei voida murtaa, vaikka avainpari paljastuisi. Vaikka tila on PreSharedKey-tilaa raskaampi, pienentää luottamusketjun validoinnin puute prosessointiaikaa verrattuna sertifikaatteihin. Toisaalta suojauksen taso verrattuna sertifikaattien käyttöön on matalampi.

Sertifikaattien epäsymmetrinen avainpari, sekä X.509 sertifikaatti tuottavat korkeimman turvallisuuden tason. Sertifikaattien ulkoinen validointi sertifikaatin myöntäjän toimesta on käytössä ja turvallisuusinfrastruktuurin olemassaolo vaaditaan. Tärkein etu tilassa on mahdollisuus peruuttaa varmenteita, jolloin tietyn laitteen estäminen verkossa on mahdollista tarvittaessa. Toisaalta sertifikaattien käyttö on kaikista raskain turvallisuustiloista.

Turvallisuustilan valinta on siis tasapainottelua turvallisuuden ja suorituskyvyn välillä. Rajoittuneiden laitteiden ollessa kyseessä ei voida olettaa, että kaikissa, tai edes useimmissa, käyttötapauksissa olisi mahdollista käyttää korkeinta turvallisuustilaa. Toisaalta ilman mitään suojausta eli NoSec-tilassa protokolla on todella altis useille eri hyökkäyksille, joten suojauksen käyttämättä jättäminen ei ole vaih-

toehto, jos verkkoon on mahdollista kohdistua uhkia. Korkeampaa turvallisuustilaa ja etenkin symmetrisiä avaimia käytettäessä vahvojen avaimien käyttäminen on keskeistä, jotta raa'an voiman hyökkäykset saadaan estettyä. Jos suojaus toteutetaan alemmilla kerroksilla, on NoSec-tila mahdollinen vaihtoehto. Todennäköisesti NoSec olisi myös hyvä vaihtoehto tässä tapauksessa, sillä rajoittuneiden laitteiden tapauksessa alemman kerroksen suojaus kuormittaa jo liikennettä, jolloin korkeammat turvallisuustilat kuormittaisivat sitä entisestään.

6.3 CoAP-protokollan tietoturvallisuus

CoAP-protokollan tietoturvallisuudessa keskeinen asia on sen pohjautuminen UDP-protokollaan. UDP on yhteydetön protokolla, jonka käyttäminen on yleisesti tunnistettu sisältävän useita erilaisia uhkia, jos asianmukaiset suojaukset eivät ole käytössä. Näitä uhkia ovat esimerkiksi alttius spoofingille, vahvistushyökkäykset ja protokollien väliset hyökkäykset. Nämä kaikki aiemmassa tutkimuksessa todettu myös CoAP-protokollan tietoturvauhiksi.

Testausvaiheessa toteutetun spoofingin ja vahvistushyökkäyksen havaintojen perusteella voidaan todeta, että nämä uhkat ovat CoAP:n osalta relevantteja ja helposti toteutettavia, jos suojausta ei ole käytössä. Tämä koskee myös testin ulkopuolisia uhkia, sillä niiden toiminta pohjautuvat pitkälti onnistuneeseen spoofingiin, mikä testauksessa toteutettiin.

DTLS-suojauksen käyttöönotolla PreSharedKey-, RawPublicKey- ja sertifikaattitilassa voidaan näitä tietoturvauhkia rajoittaa merkittävästi. Merkittävä osa tästä rajoittamisesta perustuu siihen, että yhteydettömän UDP:n sijasta DTLS sisältää kättelymekanismin. Tällöin spoofing ja siihen perustuvat hyökkäystavat eivät ole enää mahdollisia, ainakaan tutkielmassa käsitellyllä tavalla.

DTLS:n ollessa käytössä jäljelle jäävät kuitenkin jäsennyshyökkäykset, protokollien väliset hyökkäykset ja välimuistihyökkäykset. NoSec:iä korkeammissa turvallisuustiloissa hyökkäykset vaativat kuitenkin onnistuakseen DTLS-yhteyden muodostamista. Tällöin hyökkääjällä tulisi siis olla käytössään esimerkiksi murrettu avain. Poikkeuksen tekee välimuistihyökkäys, joka perustuu välityspalvelimen vääränlaiseen toteutukseen. Koska hyökkäys kohdistuu välityspalvelimeen, ei salattuun liikenteeseen ei salauksella ole merkitystä sen toimintaan. Salattu liikenne voi tosin olla hyökkäyksen kautta uhattuna, sillä salaus puretaan välityspalvelimella.

Yhteenveto CoAP-protokollan kohdistuvista hyökkäyksistä ja DTLS:n käytön

vaikutuksesta niiden toteutettavuuteen on esitetty taulukossa 6.1. Taulukossa on esitetty NoSec-tilassa toteutettavissa olevat hyökkäykset, sekä DTLS-suojauksesta huolimatta toteutettavissa olevat hyökkäykset, joista DTLS-yhteyden muodostamista (esimerkiksi murretulla avaimella) vaativat hyökkäykset on merkitty sulkumerkein.

Taulukko 6.1: Turvallisuustilojen vaikutus hyökkäyksiin

Hyökkäys	**NoSec**	**DTLS**
Spoofing	x	
Vahvistushyökkäykset	x	
Jäsennyshyökkäykset	x	(x)
Protokollien väliset hyökkäykset	x	(x)
Välimuistihyökkäykset	x	x

Yhteenvetona voidaan todeta, että DTLS-suojausta käyttävä CoAP-protokolla on tietoturvallinen, jos käytettävä turvallisuustila ja tilan toteutus avainten vahvuuden osalta vastaavat uhkatasoa. Lisäksi kirjastojen ja toteutusten, kuten välityspalvelimen tulee olla valittuja ja päivitettyjä siten, että ne eivät sisällä tunnettuja haavoittuvuuksia, sekä myös bootstrapping-prosessin on oltava turvallinen. Haasteen turvalliselle CoAP-protokollan käytölle tuottaa rajoittuneiden laitteiden kontekstissa turvallisuustilojen käytön aiheuttama epätoivottu prosessoinnin ja liikennöinnin lisääntyminen, joka voi johtaa liian matalan turvallisuustilan valintaan, sekä laitteiden päivittämisen haasteellisuus.

7 Yhteenveto

IoT-laitteiden tietoturvallisuuteen kohdistuu uhkia kaikilla protokollapinon kerroksilla, mukaan lukien sovelluskerros. UDP:ta käyttävät protokollat, kuten CoAP, ovat erityisen alttiita tietyille hyökkäyksille, kuten spoofingille, vahvistushyökkäyksille ja protokollien välisille hyökkäyksille. Nämä hyökkäykset ovat myös keskeisiä CoAP:hen kohdistuvia hyökkäyksiä.

Tutkielman tavoitteena oli vastata tutkimuskysymykseen ”Onko CoAP-protokolla tietoturvallinen?” Kysymykseen vastaamiseksi toteutettiin kirjallisuuskatsaus, sekä rakennettiin konstruktiivista tutkimusmenetelmää hyödyntäen CoAP-protokollan tietoturvallisuuden testausasetelma. Testausasetelmaa hyödynnettiin testaamalla siinä protokollaan kohdistuva spoofing ja yksinkertainen vahvistushyökkäys, sekä DTLS-suojatun liikenteen tarkastelu. Kirjallisuuskatsausta ja testauksen tuloksia yhdistäen toteutettiin CoAP:n tietoturvallisuuden analyysi, jolla vastattiin tutkimuskysymykseen.

Tutkielman päätuloksena oli, että DTLS-suojattu CoAP-protokolla on tietoturvallinen, kun sen turvallisuustiloja käytetään oikein uhkataso huomioiden. Lisäksi käytettävien kirjastojen ja toteutuksien tulee olla päivitettyjä siten, että ne eivät sisällä tunnettuja haavoittuvuuksia. Ongelmaksi jää, halutaanko turvallisuustiloja käyttää ja toteutetaanko niiden käyttö tehokkaasti. Rajoittuneiden IoT-laitteiden tapauksessa turvallisuustilojen raskaus ja lisääntynyt liikenne voivat houkutella matalamman turvallisuustilan käyttämiseen, jolloin saavutetaan virransäästöä, sekä mahdollisuus käyttää yksinkertaisempia ja edullisempia laitteita.

Tutkimusprosessin aikana tunnistettiin mahdollisia kohteita jatkotutkimukselle. Keskeisin näistä on IPsec-suojauksen mahdollisuuksien selvittäminen CoAP:n yhteydessä. Protokollan määrittelyssä ja aiemmassa tutkimuksessa IPsec on tunnistettu tavaksi toteuttaa CoAP:n suojaus. Sen toteutuksesta ja vertailusta DTLS:n suhteen ei kuitenkaan löydetty tutkimusta.

Toinen keskeinen jatkotutkimuksen kohde on kevyiden DTLS-toteutusten vertailu, sekä DTLS:n versio 1.3 merkitys CoAP:n suojauksessa. Aiemmassa tutkimuksessa on tuotu esimerkiksi esiin pakattu versio DTLS-protokollasta. Kevyemmällä toteutuksella korkeampien turvallisuustilojen käyttäminen voisi olla houkuttele-

vampaa. Lisäksi tutkielman aikana tunnistettiin, että bootstrapping-prosessiin, protokollien välisiin hyökkäyksiin sekä välimuistihyökkäyksiin liittyvä aiempi tutkimus CoAP:n näkökulmasta on vähäistä. Nämä voisivat myös olla yksi jatkotutkimuksen kohde.

Lähteet

[1] AL-FUQAHA, A., GUIZANI, M., MOHAMMADI, M., ALEDHARI, M., JA AYYASH, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials 17*, 4 (2015), 2347–2376.

[2] ALGHAMDI, T. A., LASEBAE, A., JA AIASH, M. Security analysis of the constrained application protocol in the internet of things. Julkaisusarjassa *Second International Conference on Future Generation Communication Technologies (FGCT 2013)* (Lontoo, Iso-Britannia, Marraskuu 2013), IEEE, 163–168.

[3] ALMEGHLEF, S. M., AL-GHAMDI, A. A.-M., RAMZAN, M. S., JA RAGAB, M. Machine learning-based dos amplification attack detection against constrained application protocol. *Applied Sciences 13*, 13 (2023).

[4] ARVIND, S., JA NARAYANAN, V. A. An overview of security in coap: Attack and analysis. Julkaisusarjassa *2019 5th International Conference on Advanced Computing Communication Systems (ICACCS)* (Coimbatore, Intia, Maaliskuu 2019), IEEE, 655–660.

[5] ATZORI, L., IERA, A., JA MORABITO, G. The internet of things: A survey. *Computer Networks 54*, 15 (2010), 2787–2805.

[6] BERGMANN, O., GERDES, S., SCHÄFER, S., JUNGE, F., JA BORMANN, C. Secure bootstrapping of nodes in a coap network. Julkaisusarjassa *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)* (Pariisi, Ranska, Toukokuu 2012), IEEE, 220–225.

[7] BUCKL, C., SOMMER, S., SCHOLZ, A., KNOLL, A., KEMPER, A., HEUER, J., JA SCHMITT, A. Services to the field: An approach for resource constrained sensor/actor networks. Julkaisusarjassa *2009 International Conference on Advanced Information Networking and Applications Workshops* (Bradford, Iso-Britannia, Toukokuu 2009), IEEE, 476–481.

[8] BUTUN, I. *Prevention and Detection of Intrusions in Wireless Sensor Networks-Networks*. PhD thesis, University of South Florida, Digital Commons, Tammikuu 2013.

[9] BUTUN, I., ÖSTERBERG, P., JA SONG, H. Security of the internet of things: Vulnerabilities, attacks, and countermeasures. *IEEE Communications Surveys Tutorials 22*, 1 (2020), 616–644.

[10] CHAVAN, A. A., JA NIGHOT, M. K. Secure coap using enhanced dtls for internet of things. *International Journal of Innovative Research in Computer and Communication Engineering 2*, 12 (2014), 7601–7608.

[11] DOCKER. Docker overview. URL `https://docs.docker.com/get-started/overview/`, viitattu 23.11.2023.

[12] FETTE, I. Google chrome, chromium, and google. URL `https://blog.chromium.org/2008/10/google-chrome-chromium-and-google.html`, viitattu 26.11.2023.

[13] FOUNDATION, W. About wireshark. URL `https://www.wireshark.org/about.html`, viitattu 23.11.2023.

[14] FRANKEL, S., JA KRISHNAN, S. *IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap*. 6071, Helmikuu 2011.

[15] GITHUB. Denial of service vulnerability caused by improper exception handling while parsing of coap messages. URL `https://github.com/Tanganelli/CoAPthon3/issues/16`, viitattu 31.10.2023.

[16] HARTKE, K. *Observing Resources in the Constrained Application Protocol (CoAP)*. 7641, Elokuu 2015.

[17] ILIC, A., STAAKE, T., JA FLEISCH, E. Using sensor information to reduce the carbon footprint of perishable goods. *IEEE Pervasive Computing 8*, 1 (2009), 22–29.

[18] JOHNSON, D., JA KETEL, M. Iot: Application protocols and security. *International Journal of Computer Network and Information Security 11*, 4 (2019), 1–8.

[19] LUKKA, K. Kari lukka: Konstruktiivinen tutkimusote. URL https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/, viitattu 26.10.2023.

[20] MAGGI, F., VOSSELER, R., JA QUARTA, D. *The Fragility of Industrial IoT's Data Backbone: Security and Privacy Issues in MQTT and CoAP Protocols*. Raportti, Trend Micro Incorporated, Texas, Yhdysvallat, Joulukuu 2018.

[21] MKOVATSC. Copper for chrome (cu4cr) coap user-agent. URL https://github.com/mkovatsc/Copper4Cr, viitattu 23.11.2023.

[22] MOSENIA, A., JA JHA, N. K. A comprehensive study of security of internet-of-things. *IEEE Transactions on Emerging Topics in Computing 5*, 4 (2017), 586–602.

[23] NAIK, N. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. Julkaisusarjassa *2017 IEEE International Systems Engineering Symposium (ISSE)* (Wien, Itävalta, Lokakuu 2017), IEEE, 1–7.

[24] NEBBIONE, G., JA CALZAROSSA, M. C. Security of iot application layer protocols: Challenges and findings. *Future Internet 12*, 3 (2020).

[25] NIYATO, D., HOSSAIN, E., JA CAMORLINGA, S. Remote patient monitoring service using heterogeneous wireless access networks: architecture and optimization. *IEEE Journal on Selected Areas in Communications 27*, 4 (2009), 412–423.

[26] NVD, N. V. D. Cve-2018-12679 detail. URL https://nvd.nist.gov/vuln/detail/CVE-2018-12679, viitattu 31.10.2023.

[27] NVD, N. V. D. Cve-2019-17212 detail. URL https://nvd.nist.gov/vuln/detail/CVE-2019-17212, viitattu 1.11.2023.

[28] OFFSEC. What is kali linux? URL https://www.kali.org/docs/introduction/what-is-kali-linux/, viitattu 25.11.2023.

[29] PIXEP. Coap test server. URL https://github.com/Pixep/coap-testserver-docker, viitattu 25.11.2023.

[30] PLGD DEV. Go-coap. URL https://github.com/plgd-dev/go-coap, viitattu 23.11.2023.

[31] POSTEL, J. *User Datagram Protocol*. 768, Heinäkuu 1980.

[32] PROJECT, E. C. Californium (cf) - coap for java. URL https://github.com/eclipse-californium/californium, viitattu 25.11.2023.

[33] RAHMAN, R. A., JA SHAH, B. Security analysis of iot protocols: A focus in coap. Julkaisusarjassa *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)* (Masqat, Oman, Maaliskuu 2016), IEEE, 1–7.

[34] RATHOD, D., JA PATIL, S. Security analysis of constrained application protocol (coap): Iot protocol. *International Journal of Advanced Studies in Computers, Science and Engineering 6*, 8 (2017), 37–41.

[35] RAZA, S., SHAFAGH, H., HEWAGE, K., HUMMEN, R., JA VOIGT, T. Lithe: Lightweight secure coap for the internet of things. *IEEE Sensors Journal 13*, 10 (2013), 3711–3720.

[36] RESCORLA, E., JA MODADUGU, N. *The Datagram Transport Layer Security Version 1.2*. 6347, Tammikuu 2012.

[37] RESCORLA, E., TSCHOFENIG, H., JA MODADUGU, N. *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3*. 9147, Huhtikuu 2022.

[38] SECDEV. Scapy. URL https://github.com/secdev/scapy, viitattu 23.11.2023.

[39] SHELBY, Z., HARTKE, K., JA BORMANN, C. *The Constrained Application Protocol (CoAP)*. 7252, Toukokuu 2014.

[40] SPIESS, P., KARNOUSKOS, S., GUINARD, D., SAVIO, D., BAECKER, O., DE SOUZA, L. M. S., JA TRIFA, V. Soa-based integration of the internet of things in enterprise services. Julkaisusarjassa *2009 IEEE International Conference on Web Services* (Los Angeles, Yhdysvallat, Heinäkuu 2009), IEEE, 968–975.

[41] TEAM, L. Lubuntu manual. URL https://manual.lubuntu.me/stable/, viitattu 25.11.2023.

[42] VILAMOVSKA, A.-M., HATZIANDREU, E., SCHINDLER, H. R., VAN ORANJE-NASSAU, C., DE VRIES, H., JA KRAPELS, J. *Study on the Requirements and Options for RFID Application in Healthcare: Identifying Areas for Radio Frequency Identification Deployment in Health Care Delivery; A Review of Relevant Literature*. Raportti, Rand Corporation, Yhdysvallat, Huhtikuu 2009.

[43] VIRTUALBOX. Virtualbox manual: Chapter 1. first steps. URL https://www.virtualbox.org/manual/ch01.html, viitattu 23.11.2023.

[44] WOOD, A. D., JA STANKOVIC, J. A. Denial of service in sensor networks. *Computer 35*, 10 (2002), 54–62.

[45] YASSEIN, M. B., SHATNAWI, M. Q., JA AL-ZOUBI, D. Application layer protocols for the internet of things: A survey. Julkaisusarjassa *2016 International Conference on Engineering MIS (ICEMIS)* (Agadir, Marokko, Syyskuu 2016), IEEE, 1–4.

[46] ZAMFIR, S., BALAN, T., ILIESCU, I., JA SANDU, F. A security analysis on standard iot protocols. Julkaisusarjassa *2016 International Conference on Applied and Theoretical Electricity (ICATE)* (Craiova, Romania, Lokakuu 2016), IEEE, 1–6.

A CoAP-viestin vastauskoodit ja valinnat

Taulukko A.1: CoAP-viestin vastauskoodit[39]

Koodi	**Vastaus**
2.01	Created
2.02	Deleted
2.03	Valid
2.04	Changed
2.05	Content
4.00	Bad Request
4.01	Unauthorized
4.02	Bad Option
4.03	Forbidden
4.04	Not Found
4.05	Method Not Allowed
4.06	Not Acceptable
4.12	Precondition Failed
4.13	Request Entity Too Large
4.15	Unsupported Content-Format
5.00	Internal Server Error
5.01	Not Implemented
5.02	Bad Gateway
5.03	Service Unavailable
5.04	Gateway Timeout
5.05	Proxying Not Supported

Taulukko A.2: CoAP-viestin valinnat ja valintanumerot [39]

Numero	**Valinta**	**Selite**
0	(Reserved)	Varattu
1	If-Match	Pyynnön toteutus vain vastaavuuden täyttyessä
3	Uri-Host	Resurssin isännän määrittäminen (Internet)
4	ETag	Paikallinen resurssitunniste saman resurssin eri esitystavoille
5	If-None-Match	Pyynnön toteutus vain, kun vastaavuutta ei löydy
7	Uri-Port	Resurssin porttinumero kuljetuskerroksella
8	Location-Path	Yksi segmentti resurssin absoluuttisesta polusta (sijainti)
11	Uri-Path	Yksi segmentti resurssin absoluuttisesta polusta (kohde)
12	Content-Format	Hyötykuorman esitystavan määrittäminen
14	Max-Age	Maksimiaika vastauksen välimuistiin tallentamiselle
15	Uri-Query	Yksi argumentti resurssin parametrien asettamiseksi (kohde)
17	Accept	Sallittujen hyötykuorman esitystapojen määritys
20	Location-Query	Yksi argumentti resurssin parametrien asettamiseksi (sijainti)
35	Proxy-Uri	Välityksen pyytäminen välityspalvelimelta
39	Proxy-Scheme	Välityspalvelimen kaavan määrittäminen
60	Size1	Resurssin esityksen koon määrittäminen
128	(Reserved)	Varattu
132	(Reserved)	Varattu
136	(Reserved)	Varattu
140	(Reserved)	Varattu