Alex Mytkäniemi

Riskiarvio työpöytä- ja mobiilialustojen käynnistyslaiteohjelmistojen kyberuhista

Tietotekniikan kandidaatintutkielma

17. toukokuuta 2021

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Alex Mytkäniemi

Yhteystiedot: mytkanam@student.jyu.fi

Ohjaaja: Tuomo Rossi

Työn nimi: Riskiarvio työpöytä- ja mobiilialustojen käynnistyslaiteohjelmistojen kyberuhista

Title in English: Risk assessment of cyber threats concerning boot firmware of desktop and mobile platforms

Työ: Kandidaatintutkielma

Opintosuunta: Kaikki opintosuunnat

Sivumäärä: 46+0

Tiivistelmä: Tässä tutkielmassa käsitellään yleisimpien työpöytä- ja mobiilialustojen käynnistyslaiteohjelmistoihin kohdistuvia kyberuhkia. Tarkastelun kohteena ovat UEFI-laiteohjelmisto Intel-arkkitehtuurin työpöytäalustoilla sekä Qualcommin laiteohjelmisto ARM-arkkitehtuurin mobiilialustoilla. Uhkien kartoitus tehtiin analysoimalla National Vulnerability Databasen haavoittuvuuksia sekä tutkimuksissa että kentällä toteutettuja hyökkäyksiä. Tulokset viittaavat siihen, että mobiilialustat ovat vähemmän alttiita laiteohjelmistouhkille kuin työpöytäalustat.

Avainsanat: Laiteohjelmisto, Laiteohjelmistoturvallisuus, UEFI, Qualcomm, Kandidaatintutkielmat

Abstract: This document covers cyber threats targeting boot firmware of most common desktop and mobile platforms. These include UEFI Firmware on Intel architechture based desktop platforms and Qualcomm Firmware on ARM architechture based mobile platforms. For threat analysis, exploits from National vulnerability database as well as previous PoC and in the wild attacks of firmware were examined. Results seem to indicate that mobile platforms are more resilient to firmware threats than desktop platforms.

Keywords: Firmware, Firmware security, UEFI, Qualcomm, Bachelor's Theses

Termiluettelo

Laiteohjelmisto	Laitteen lukumuistissa säilytettävä ohjelmisto (Rose 1967).
Käynnistyslataaja	Laiteohjelmiston osa, joka lataa varsinaisen käyttöjärjestelmän (Sloss, Symes ja Wright 2004).
JTAG	(Joint Test Action Group) Nimitys standardille, joka määrittelee mm. mikropiirien testaukseen käytettävän testausväylän (IEEE 2013).
Suojauskehät	(Rings of Priviledge) Etenkin Intel-arkkitehtuureissa yleinen tapa kuvata suoritettavien käskyjen oikeudet kehärakenteena, jossa oikeutetuimmat käskyt suoritetaan sisimmällä kehällä (esim. Intel 2020a).
Poikkeustasot	(Exception Levels) ARM-arkkitehtuurin tapa kuvata suoritettavien käskyjen oikeuksia, niin että oikeutetuimmat käskyt suoritetaan korkeimmalla poikkeustasolla (ARM 2021).
CoT	(Chain of Trust) Turvallisuusmenettely, jossa ohjelmat latautuvat ketjussa, jonka jokainen vaihe varmistaa seuraavan vaiheen eheyden ennen sen suorittamista (esim. Redini ym. 2017).
Secure Boot	UEFI:n käyttämä sertifikaatteihin perustuva käynnistyksen eheyden varmistus (Nystrom, Nicoles ja Zimmer 2011).
Trusted Boot	ARM-referenssitoteutus luottamusankkurillisesta käynnistyksen eheyden varmistuksesta, jota hyödynnetään Qualcommin käynnistysketjussa (Redini ym. 2017).
PBL	(Primary Boot Loader) Qualcomm-järjestelmäpiirien ensimmäisen vaiheen lataaja, jota säilytetään lukumuistissa (Dent 2019).
SBL	(Secondary Boot Loader) Qualcomm-järjestelmäpiirien toisen vaiheen lataaja, joka lataa ja alustaa turvallisen suoritusympäristön (Redini ym. 2017).
XBL	(Extensible Boot Loader) Uudempien Qualcomm-järjestelmäpiirien hyödyntämä toisen vaiheen lataaja, joka perustuu UEFI-moduuleihin

	(Zhang 2018)
Aboot	(Android Bootloader) Qualcomm-järjestelmäpiirien viimeisen vaiheen lataaja, joka lataa käyttöjärjestelmän ytimen tai palautusympäristön (Appere 2019).
EDL	(Emergency Download Mode) PBL:n suoritustila, jossa voidaan korjata laitteen korruptoitunut laiteohjelmisto hyödyntäen erityistä Firehose-ohjelmoijaa (Aleph Security 2018b).
Fastboot	Abootin suoritustila ja protokolla, jonka avulla voidaan suorittaa erilaisia järjestelmän hallintatoimintoja (Hay 2017; Google 2012)
UEFI	(United Extensible Firmware Interface) BIOS-laiteohjelmistojen korvaajaksi kehitetty alustariippumaton laiteohjelmistokokonaisuus (UEFIForum 2020b).
SMM	(System Management Mode) Intel- ja AMD-prosessorien erityinen toimintatapa laiteohjelmistojen käyttöön, jossa voidaan suorittaa esimerkiksi järjestelmän hallintatoimintoja (AMD 2021; Intel 2020a).
SMRAM	(System-Management RAM) SMM:n käyttöön tarkoitettu erillinen muistialue, jossa voidaan säilyttää esimerkiksi järjestelmän hallintatietoja (Intel 2020a).
Secure Monitor Mode	ARM-prosessorien erityinen toimintatapa, jota käytetään esimerkiksi turvallisen suoritusympäristön ylläpitämiseen (ARM 2021).
BMC	(Baseboard Management Controller) Etenkin palvelimissa käytettävä autonominen alijärjestelmä, jolla voidaan etähallita järjestelmää laitteistopohjaisesti (Frazelle 2020a).
Intel CSME	(Converged Security and Management Engine) Intel-arkkitehtuuriin sisältyvä järjestelmän suojaus- ja hallintatoimintoihin kykenevä autonominen alijärjestelmä (Intel 2020c).
Mobiiliverkkopino	(Baseband Stack) Erillisen prosessorin suorittama ohjelmistokokonaisuus, joka toteuttaa älypuhelimen mobiiliverkkotoi-

	minnot (Grassi, Liu ja Xie 2018).
Hayes-komennot	Puhelinverkkolaitteiden, kuten modeemien ohjaamiseen kehitetyt komennot (Tian ym. 2018).

Taulukot

Sisällys

1 Johdanto

Digitalisaation myötä monet tavalliset sähkölaitteetkin tarvitsevat ohjelmakoodia toimiakseen. Tätä laitteesta erottamatonta ohjelmakoodia kutsutaan laiteohjelmistoksi ja sen avulla toteutetaan laitteen välttämätön toiminnallisuus. Joissain laitteissa, kuten tietyissä jääkaapeissa tai varavirtalähteissä, yksittäinen laiteohjelmiston sisältävä mikrokontrolleri riittää toiminnan ohjaamiseen (Chen ja Chen 2020; Belman-Flores ym. 2015). Tietokonejärjestelmissä asia ei kuitenkaan ole aivan näin yksinkertainen, sillä emolevyn laiteohjelmiston lisäksi myös jokainen oheislaite voi sisältää laiteohjelmiston (Frazelle 2020b). Jopa prosessorin toiminnan mahdollistavan mikrokoodin voi nähdä laiteohjelmistona.

Tieto- ja kyberturvallisuuden arkipäiväistyessä on saatu useita viitteitä siitä, että käyttöjärjestelmä- ja sovellusturvallisuuden parantuessa hyökkäyspinta siirtyy alaspäin kohti järjestelmän alempia kerroksia (UEFIForum 2019). Tässä tutkielmassa otetaan selvää siitä, minkälaisia uhkia näihin matalammalla tasolla toimiviin laiteohjelmistoihin kohdistuu. Tarkastelu kohdistettiin yleisiin työpöytä- ja mobiilialustoihin. Työpöytäalustoilla viitataan tässä tutkielmassa kuluttaja-, yritys- ja palvelintietokoneisiin jotka noudattavat Intel-arkkitehtuuria Nehalem-arkkitehtuurista alkaen. Mobiilialustoilla taas viitataan ARM-arkkitehtuuria noudattavia Qualcomm Snapdragon-järjestelmäpiirejä hyödyntäviin mobiililaitteisiin.

Tarkasteltavat laitekokonaisuudet valittiin yleisyyteen perustuen, sillä sekä Intel että Qualcomm omaavat suurimman markkinaosuuden tarkasteltavien alustatyyppien laitteistoissa (Alsop 2021, 2020). Laiteohjelmistot, joita tässä tutkielmassa tarkastellaan ovat emolevyihin liittyviä laiteohjelmistoja, nk. käynnistyslaiteohjelmistoja, jotka vastaavat järjestelmän alustamisesta. Työpöytäalustoilla tämä tarkoittaa UEFI:a ja mobiilialustoilla Qualcommin omaa laiteohjelmistokokonaisuutta.

2 Laiteohjelmistoturvallisuuden taustaa yleisesti

Aikaisempien tutkimusten valossa laiteohjelmistoturvallisuus näyttää jääneen perinteistä ohjelmistoturvallisuutta vähemmälle huomiolle (esim. Costin ym. 2014). Kiinnostus aihetta kohtaan näyttää kuitenkin kasvaneen viime vuosina, sillä lähes 66% National Vulnerability Databasen "Firmware" hakusanalla löydettävistä haavoittuvuuksista on kirjattu viimeiseltä neljältä vuodelta. Aihepiirin ajankohtaisissa tutkimuksissa on löydetty esimerkiksi vakavia haavoittuvuuksia oheislaitteiden laiteohjelmistojen päivitysmekanismeista (esim. Eclypsium 2020c).

Laiteohjelmistot ovat kannattavia hyökkäyskohteita, koska niiden avulla voidaan piiloutua syvälle järjestelmään ja hyödyntää lähes rajattomia suoritusoikeuksia (UEFIForum 2018). Laiteohjelmistouhkien kehitys ei kuitenkaan ole helppoa ja vaatii monesti olemassa olevan laiteohjelmiston takaisinmallinnusta. Kehitystyö on vielä ongelmallisempaa, jos laitetyypille ei ole olemassa laajemmin käytettyä ohjelmistoa. Tällöin uhkan mahdollistava haavoittuvuus saatetaan joutua kehittämään jokaisen valmistajan laitteelle erikseen.

Uhkien kehittäminen tässä tutkielmassa tarkasteltaville laiteohjelmistoille on varsin kannattavaa ohjelmistojen laajan hyödyntämisen vuoksi. Kumpaankin laiteohjelmistoon liittyy avoimia referenssitoteutuksia (Tianocore 2020b; Swetland 2015), joten kehitystyö on myös helpompaa kuin täysin suljetun ohjelmiston tapauksessa. Toisaalta, myös laiteohjelmistojen ja niiden alustojen rakenne sekä turvallisuusmallit, joita käsitellään seuraavaksi, vaikuttavat uhkakehitykseen.

2.1 UEFI

UEFI eli United Extensible Firmware Interface on lähtöisin Intelin Itanium-arkkitehtuurin laiteohjelmistoksi kehitetystä EFI-projektista (Zimmer, Rothman ja Marisetty 2017). Verrattuna edeltäviin BIOS-laiteohjelmistoihin, UEFI:n on tarkoitus olla laitteisto- ja alustariippu-

maton sekä modulaarinen ratkaisu (UEFIForum 2020b).

UEFI-laiteohjelmisto koostuu Platform Initialization eli PI-moduuleista, jotka alustavat laitteiston toiminnallisuuden sekä UEFI-moduuleista jotka mm. toimivat ajureina ja alustavat käyttöjärjestelmän (Zimmer, Rothman ja Marisetty 2017). Kuten BIOS, PI-moduulit ovat ennaltamäärättyjä ja laitteistokohtaisia (Zimmer, Rothman ja Marisetty 2017). Niitä voidaankin pitää järjestelmän todellisena laiteohjelmistona. UEFI-moduulit sen sijan ovat alusta- ja laitteistoriippumattomia ja toimivat rajapintana PI-moduulien ja käyttöjärjestelmän välillä (UEFIForum 2020b; Zimmer, Rothman ja Marisetty 2017). Perinteiseen BIOS-laiteohjelmistoon verrattuna UEFI:n siirrettävyys on siis hyvä, sillä vain pieni osa toiminnallisuudesta joudutaan toteuttamaan erikseen eri laitteistoille.

Laiteohjelmistona UEFI soveltuu myös mobiililaitteille (UEFIForum 2013). Niinpä nykyään lähes kaikki isommat laitevalmistajat hyödyntävät UEFI:a tavalla tai toisella (UEFIForum 2021). Laitevalmistajat käyttävät yleensä omaa toteutustaan UEFI:n referenssitoteutuksesta, Intelin Tianosta (Monroe, Branco ja Zimmer 2017). Toisin kuin laitevalmistajien toteutukset, Tiano on ollut pitkään avoimen lähteen alainen toteutus (Tianocore 2020b). Nykyään Tianocore-nimellä kutsuttavan projektin tarjoamat kehitysympäristöt mahdollistavat UEFI-ympäristön ohjelmistokehityksen käytännössä kenelle tahansa.

Käynnistettäessä UEFI-laiteohjelmisto siirtyy SEC-vaiheeseen (Security), jossa varmistetaan laiteohjelmiston eheys. Lisäksi tämän vaiheen SEC-moduulit vaihtavat prosessorin toimintatavan Protected Modeen ja konfiguroivat seuraavaan vaiheen moduuleille tilapäisen muistialueen C-koodin suoritusta varten (Tianocore 2020a). PEI-vaihe (Pre-EFI Initialization), johon siirrytään seuraavaksi, alustaa välttämättömät minimiresurssit järjestelmän ja seuraavien vaiheiden toiminnan kannalta. Näihin sisältyvät esimerkiksi pysyvä fyysinen muisti ja tietyt UEFI-protokollat (UEFIForum 2020a; Tianocore 2020a; Zimmer, Rothman ja Marisetty 2017).

Edellisten vaiheiden aikana ympäristön resurssit ovat varsin rajalliset, eivätkä monet alustan kannalta tärkeät turvallisuusominaisuudet ole vielä käytössä (UEFIForum 2020a). Tämänkin vuoksi kyseisten vaiheiden ohjelmakoodin muokkaaminen on tarkoituksenmukaista vain laitevalmistajille. Kun nämä ympäristön alustavat vaiheen on suoritettu, siirrytään ensimmäiseen UEFI-moduuleja hyödyntävään vaiheeseen, jota kutsutaan DXE:ksi (Driver Execution Environment) (Zimmer, Rothman ja Marisetty 2017).

DXE-vaiheessa ladataan kaikista tärkeimmät ajurit ja UEFI-moduulit sekä kartoitetaan oheislaitteet. Samalla alustetaan järjestelmän turvallinen suoritusympäristö (UEFIForum 2020a; Tianocore 2020a). Esimerkiksi x86-arkkitehtuurissa tällä tarkoitetaan SMM-moduulien, kuten SMI-keskeytyskäsittelijöiden lataamista SMRAM-muistialueelle ja muistialueen lukitsemista (Yao ja Zimmer 2015). Järjestelmä on nyt vakaa ja turvallisuusominaisuudet ovat käytössä.

Seuraavaksi suoritettava BDS-vaihe (Boot Device Selection) merkkaa sitä pistettä käynnistysketjussa, jossa kaikki tarvittavat resurssit käyttöjärjestelmän käynnistämiseksi on ladattu (Zimmer, Rothman ja Marisetty 2017). Eri laitteiden UEFI-ajurit ovat hallitseva moduulityyppi tässä vaiheessa (UEFIForum 2020a). Myös UEFI-sovelluksia on mahdollista suorittaa tässä vaiheessa ja esimerkiksi järjestelmän käynnistyslataaja, kuten Windows BootManager, on tosiasiassa UEFI-sovellus (Microsoft 2017; Zimmer, Rothman ja Marisetty 2017).

UEFI-laiteohjelmistoon liittyviä tietoja säilytetään ympäristömuuttujissa, joista osa on pysyviä ja tallessa haihtumattomassa NVRAM-muistissa (UEFIForum 2020b). UEFI-laiteohjelmiston moduulit itsessään toimivat pitkälti hyödyntämällä eri laiteajurien tarjoamia rajapintoja sekä nk. palvelutauluja, esimerkiksi käynnistyspalveluja (Boot Services) ja ajonaikaisia palveluita (Runtime Services) (Zimmer, Rothman ja Marisetty 2017). Etenkin myöhempien vaiheiden moduulien palveluiden tarjoaminen toisille moduuleille näiden mekanismien kautta on tarkoituksenmukaista (UEFIForum 2020b). Laiteohjelmiston moduuli voi olla joko käynnistyksenaikainen tai ajonaikainen. Jälkimmäistä ei poisteta muistista käyttöjärjestelmän käynnistyksen jälkeen (Tianocore 2020a).

UEFI-ympäristön käyttökelpoisuutta hyökkäyspintana rajoittaa sen yksisäikeisyys, joka estää moduulin tausta-ajon (Fish 2012). Käytetystä moduulista ei myöskään ole hyötyä laiteohjelmiston käynnistysketjun päätyttyä, ellei se ole ajonaikainen. Ajonaikaisen moduulin toteuttaminen on kuitenkin vaikeaa etenkin siksi, että moduulin tulee tarvittaessa kyetä vaihtamaan virtuaalisiin muistiosoitteisiin tekemättä oletuksia ladattavasta käyttöjärjestelmästä (Tianocore 2018).

UEFI-laiteohjelmistoa vastaan on onnistuttu hyökkäämään kaikesta huolimatta sekä tutkimuksissa että kentällä. Tähän mennessä onnistuneita hyökkäyksiä on kohdistettu jokaiseen ketjun osaan paitsi SEC-vaiheeseen (ESET 2018; Oleksiuk 2016; Kallenberg ym. 2014; Kaczmarek 2013). Myös SMM:a vastaan on hyökätty onnistuneesti UEFI-ympäristössä ainakin kerran (Oleksiuk 2015). Monen onnistuneen hyökkäyksen mahdollistajana näyttää olleen väärin konfiguroitu laiteohjelmisto.

2.2 Aboot ja Qualcomm-käynnistysketju

Työpöytäalustoilla UEFI on johtava laiteohjelmisto, mutta esimerkiksi mobiilialustoilla hajontaa on enemmän. Yksi vaihtoehto on Qualcommin hyödyntämä Little Kernel (LK). LK on Travis Geiselbrechtin projekti kevyen 32-bittisen käyttöjärjestelmän kehittämiseksi, joka soveltuu esimerkiksi sulautettuihin järjestelmiin laiteohjelmistoksi että yleisemmin käynnistyslataajaksi (Geiselbrecht 2018; Swetland 2015). Aboot on Qualcommin johdannainen tästä projektista (Redini ym. 2017).

Qualcommin mallin mukaisessa käynnistysketjussa Aboot on viimeisen vaiheen lataaja. Ensimmäisenä laitteen käynnistyessä ladataan PBL. PBL on tallessa erillisellä lukumuistilla, joten sen muuttaminen on käytännössä erittäin vaikeaa. Siksi se toimiikin käynnistysketjun luottamusankkurina (Anchor of Trust) (Dent 2019; Appere 2019). PBL toteuttaa myös EDL:n, jonka avulla ketjun seuraavat osat voidaan päivittää (Aleph Security 2018a). Seuraa-

van vaiheen lataaja haetaan Flash-muistista eikä ole sama kaikissa järjestelmäpiireissä. Monissa laitteissa lataajasta käytetään termiä SBL ja sillä ladataan sekä järjestelmän turvallinen suoritusympäristö että Aboot eli alustetaan järjestelmään "normaali maailma"ja "turvallinen maailma" (Guilbon 2018b; Redini ym. 2017).

Joissain uudemmissa laitteissa kuvio on hieman erilainen. PBL suorittaa kaksi toisen vaiheen lataajaa. Näistä ensimmäisenä suorittettava XBL_SEC lataa turvallisen suoritysympäristön kun taas XBL lataa Abootin (Dent 2019). Oli käytettävä latausketju mikä tahansa, kun suoritus siirtyy Abootille, sen tehtävänä on ladata käyttöjärjestelmä sekä alustaa muistinhallinnan ja oheislaitteiden toiminnallisuus (Qualcomm 2016).

Oikeuksien lisäksi Aboot on kannattava hyökkäyskohde siksi, että uudemmissa Android-järjestelmissä Aboot tai LK voi olla muistissa silloinkin, kun käyttöjärjestelmä on käynnissä (Mahate 2016). Kannattavuutta lisää Abootin kyky kirjoittaa osioihin sekä suorittaa erillinen palautusympäristö (Redini ym. 2017). Aboot voi olla erittäin hyödyllinen lataaja myös siinä mielessä, että moni laitevalmistaja laajentaa sen toimintaa omilla komennoillaan (Hay 2017).

Toisin kuin UEFI-laiteohjelmistossa, Qualcomm-lataajista ei vaikuttaisi olevan rajapintaa turvalliseen suoritusympäristöön sen alustamisen jälkeen. Lisäksi kyseinen suoritusympäristö, QTEE ja sen sovellukset, Trustletit, toimivat samalla poikkeustasolla kuin normaali käyttöjärjestelmä ja sen sovellukset (Appere 2019). Jos järjestelmä halutaan täydelliseen hallintaan, voidaan hyökkäys kohdistaa eri maailmojen väylänä toimivaan Secure Monitoriin, joka suorittuu korkeimmalla mahdollisella poikkeustasolla. Appere (2019) päätyi siihen tulokseen, että myös PBL ja SBL suorittuvat korkeimmalla poikkeustasolla. Abootia edeltävien lataajien käytettävyydestä järjestelmän käynnistymisen jälkeen ei kuitenkaan vaikuta olevan julkista tietoa.

2.3 Hallintaohjaimet

Vaikka useassa prosessoriarkkitehtuurissa on erillinen toimintatapa järjestelmään hallintaan ja turvallisen suoritusympäristön toteuttamiseen, myös erillistä laitteistopohjaista ratkaisua voidaan käyttää samojen asioiden toteuttamiseen (Regenscheid 2018; Ning ym. 2017). Nykyään tällainen nk. hallintaohjain on kiinteästi integroitu järjestelmän käynnistysketjuun ja voidaankin siten nähdä osana järjestelmän käynnistyslaiteohjelmistoa (Frazelle 2020b).

Järjestelmän kokoonpanosta riippuen ohjaimella voidaan esimerkiksi hakea komponenttien tietoja, tarkkailla niiden kuntoa ja lämpötilaa, hallita järjestelmän virtatilaa sekä päivittää laiteohjelmisto (Giri, Iler ja Krebs 2020; Regenscheid 2018). Moni hallintaohjain on myös yhteydessä verkkoon erillisen hallintakanavan kautta, mahdollistaen etähallinnan. Tällöin puhutaan kaistan ulkopuolisesta hallinnasta (Out-of-Band Management) (Mrazek ja Bay 2020).

Hallintaohjain on käynnissä aina kun laitteen komponentit saavat virtaa, myös silloin kun muu järjestelmä on pois päältä (Intel 2020c; Regenscheid 2018). Järjestelmän tila ei siis vaikuta sen hallittavuuteen. Laitteistopohjainen etähallinta voikin olla erittäin hyödyllinen ominaisuus esimerkiksi kustannusten vähentämisen sekä nopeaan viankorjaukseen. Siksi etenkin palvelimissa on tavallisesti ollut BMC hallintaohjaimena jo vuosia (Regenscheid 2018).

Vähitellen hallintaohjaimia alettiin kuitenkin hyödyntämään myös muissa laitteissa. Vuonna 2005 Intel esitteli ensimmäisen version Active Management Technologysta, joka mahdollisti minkä tahansa yhteensopivan laitteiston etähallinnan hyödyntäen verkkokorttiin lisättyä hallintaohjainta (Intel 2005). Kun Intel siirtyi uuteen Nehalem-arkkitehtuuriin vuonna 2008, samaa hallintaohjainta, jota kutsutaan Intel Management Engineksi, alettiin integroida suoraan arkkitehtuurin piirisarjaan (Intel 2009). Samoihin aikoihin myös servereissä alettiin hyödyntää hallintaohjaimen integrointia emolevylle erillisen PCI-kortin sijasta (Dell 2008).

Management Enginestä tuli entistä tiiviimpi osa arkkitehtuuria, kun sitä alettiin käyttää järjestelmän turvallisena suoritusympäristönä ja luottamusankkurina (Ning ym. 2017). Nyky-

ään Intel-arkkitehtuuria hyödyntävä laite ei edes käynnisty ilman tätä alijärjestelmää, sillä ME suorittaa osan laitteiston alustuksesta (Intel 2020c). Uudemmista Management Enginen versioista käytetään nykyään lyhennettä CSME (Intel 2020b).

Rakenteellisesti CSME käyttää Minix-kerneliä jonka päälle rakennettu sovelluspino sisältää esimerkiksi Intel AMT:n (Intel 2020c). Vaikka CSME:n etähallintaominaisuudet on mahdollista poistaa käytöstä niitä tukevilla alustoilla, itse järjestelmän täysi käytöstä poistaminen on usein mahdotonta (Positive Technologies 2017).

Intel-arkkitehtuurissa piirisarjasta on rajapinta käytännössä kaikkeen järjestelmässä (Intel 2009). Siten myös CSME:stä voisi periaatteessa hyödyntää miltei mitä tahansa resurssia. Myös BMC pääsee käsiksi moneen eri järjestelmän resurssiin hyödyntämällä piirisarjan rajapintoja (Intel 2020e). Etähallintaohjaimet ovat osin juuri tästä syystä tavoiteltuja hyökkäyspintoja. Sen lisäksi, että ne mahdollistavat lähes täydellisen järjestelmän hallinnan, on niiden toiminta monesti näkymätöntä muulle järjestelmälle (Eclypsium 2020a). Hallintaohjaimen läpi tapahtuvaa hyökkäystä voikin olla erittäin vaikea havaita.

Etähallintaohjaimia vastaan on löydetty useita hyökkäysmekanismeja, joista ehkä tunnetuin on Cipher Zero. Kyseessä on BMC-ohjainten käyttämän IPMI-protokollan "ominaisuus", joka mahdollistaa turvallisuuden ohittamisen hyödyntämällä selvätekstistä tunnistautumista (Rapid7 2013). Samankaltainen haavoittuvuus koskettaa myös Intel AMT:n verkkorajapintaa, hyödyntäen Digest-tunnistautumisen manipulointia (Nixawk 2017). Voidaan siis todeta, että järjestelmän hallintaohjain voi hallintamahdollisuuksien lisäksi helpottaa huomattavasti järjestelmän varsinaiseen laiteohjelmistoon hyökkäämistä.

3 Laiteohjelmistohyökkäyksen toteuttaminen

Aikaisemmissa tutkimuksessa on todettu, että laiteohjelmistohaavoittuvuuksien mekanismit ovat pitkälti samoja kuin muissakin haavoittuvuuksissa (David, Partush ja Yahav 2018). Niihin liittyvien hyökkäysvektorien saavuttaminen on kuitenkin haastavampaa matalamman abstraktiotason johdosta. Hyökkääjän etuna on kuitenkin se, ettei laiteohjelmistoja päivitetä kovinkaan usein, joten kehitetty haavoittuvuus voi olla käyttökelpoinen tavallista pidempään.

Kehitetystä haavoittuvuudesta jalostetun uhkan toimittaminen järjestelmään on mahdollista pääsääntöisesti samoilla tavoilla kuin muissakin uhkatyypeissä. Fyysisesti sekä ohjelmallisesti, joko paikallisesti tai etäältä (Papp, Ma ja Buttyan 2015). Olipa keino sitten mikä tahansa, hyökkääjäällä on käytössään useita tilannekohtaisia menetelmiä laiteohjelmiston suojausten kiertämiseksi.

3.1 Fyysiset hyökkäykset

Laitevalmistajien on kyettävä muokkaamaan ja testaamaan tuotteen laiteohjelmistoa silloinkin kun se ei ole osa toimivaa järjestelmää. Piirilevyjen ja niiden komponenttien hallitsemiseen ja testaamiseen onkin siten olemassa monia erilaisia väyläprotokollia, joita voidaan hyödyntää myös järjestelmää vastaan hyökätessä (Carey ja Bathurst 2013).

Käyttääkseen näitä protokollia hyökkääjällä tulee olla fyysinen pääsy laitteelle sekä mahdollisesti työkalut laitteen purkamiseen. Lisäksi eri väylät ja niiden käyttämät protokollat vaativat usein ylimääräisiä laitteita ja ohjelmistoja niiden hyödyntämiseksi (Heiland 2019). Tietyt fyysiset hyökkäykset onnistuvat vain tietyissä laitteissa, sillä monet laitevalmistajat vaikeuttavat niiden toteuttamista erilaisin toimenpitein (Skorobogatov 2012).

3.1.1 Kirjoitussuojauksen ohittaminen

Intel-arkkitehtuuria noudattavissa järjestelmissä laiteohjelmisto on normaalisti suojattu ohjelmallisia kirjoitusyrityksiä vastaan (Intel 2013). Jotkin laitevalmistajat kuitenkin lisäävät emolevyille liitännän, jonka oikosulkemalla kirjoitussuojaus on mahdollista ohittaa (Coreboot 2021a; Supermicro 2013). Tällöin laiteohjelmiston päivitys on mahdollistaa toteuttaa ohjelmallisesti.

Lisäksi kirjoitussuojauksen ohittamien on mahdollista lähettämällä signaali järjestelmän IO-ohjaimelle käynnistyksen yhteydessä. Käytännössä tämä tapahtuu oikosulkemalla tietyt äänipiirin nastat (Intel 2013). Kirjoitussuojauksen fyysinen ohittaminen mahdollistaa usein rajattoman laiteohjelmistoon kirjoittamisen, myös sellaisille alueille, joille kirjoittaminen on yleensä estetty (Intel 2013).

Qualcommin-järjestelmäpiireissä vastaava tapa on PBL:n pakottaminen EDL-tilaan oikosulkemalla tietyt emolevyn nastat (Aleph Security 2018a). Koska EDL on nimenomaan suunniteltu laiteohjelmiston korjaamiseen, kaikkien osioiden kirjoittaminen on mahdollista rajoituksetta (Appere 2019).

3.1.2 Piiriohjelmointi

Jos järjestelmän laiteohjelmisto on päivitettävissä ja sen sisältävä muistipiiri tai siihen liittyvät väylät ovat saavutettavissa, voi piirin ohjelmoida suoraan erillisellä laitteella. Yleensä piiriä ei tarvitse edes irroittaa vaan ohjelmointi on tehtävissä suoraan piirilevyllä (Analog Devices 2007). Ohjelmointilaitteella voidaan ohittaa täysin muistipiirin ohjaimen asettamat luku- ja kirjoitusrajoitukset.

Menetelmän suoraviivaisuudesta huolimatta ei kuitenkaan ole näyttöä siitä, että piiriohjelmoinnilla voitaisiin kiertää laitteen turvallinen käynnistysketju kirjoittamalla suojattuihin muistialueisiin. Lisäksi ohjelmoinnissa on jonkin verran riskejä, sillä vääränlainen kytkentä

voi tuhota muistin tai ohjelmointilaitteen (Coreboot 2021c). Pahimmassa tapauksesssa laitteen muistipiiri voi olla kytketty niin, että myös levyn muut komponentit vahingoittuvat ohjelmointilaitteen virrasta (Coreboot 2021b).

3.1.3 JTAG

Vaikka mistipiiriin ei suoraan päästäisi käsiksi, voi sen ohjelmointi siitä huolimatta olla mahdollista (Laittice Semiconductor 2017). Yksi vaihtoehto ohjelmointiin on JTAG, joka on muodostunut eräänlaiseksi de facto-standardiksi valmiiden piirilevyjen testauksessa (Rosenfeld ja Karri 2010). JTAG-standardin mukainen testiportti löytyykin lähes laitteesta kuin laitteesta. Etenkin sulautetuissa järjestelmissä JTAG voi olla piirin ainoa vianetsintä- ja testausliitäntä sen monikäyttöisyyden vuoksi. Testiliitäntää voidaan käyttää esimerkiksi väylätestaukseen, vianetsintään ja tiedonsiirtoon sekä myös piiriohjelmointiin (Oshana 2013).

IEEE:n standardi ei määrittele testiportille mitään tiettyä rakennetta ja tämän vuoksi monella valmistajalla on omanlaisensa JTAG-liitäntä. Liitännässä voi olla esimerkiksi 10, 14, 16 tai 20 nastaa (Senrio 2016). On myös mahdollista ettei erillistä liitäntää ole ollenkaan, jolloin testipisteisiin on tehtävä juotoksia niiden hyödyntämiseksi (Wu ym. 2017). JTAG ei myöskään määrittele mitään tiettyä protokollaa testiliitäntöihin (Senrio 2016). Onkin siis hyvin mahdollista, että tietyn valmistajan liitännän käyttö vaatii kyseisen valmistajan työkaluja.

3.1.4 USB-rajapinnat

Levyn sisäisten väylien käyttö vaatii laitteen purkamista, mikä on erityisesti mobiilialustojen kohdalla usein vaivalloista. (Hay 2017). Hyökkääjä voi kuitenkin hyödyntää myös USB-liitännän kautta erilaisia rajapintoja hyökkäyksen toteuttamiseksi. Esimerkiksi tietyissä laitteissa on mahdollista käynnistää PBL EDL-tilassa erityisen USB-kaapelin avulla (Aleph Security 2018a). Jos tätä mahdollisuutta ei ole, voidaan hyödyntää Abootin hallintatoiminnot toteuttavaa Fastboot-protokollaa, joka on helpommin saatavilla (Hay 2017; Google 2012).

Oma lukunsa ovat aikaisemmissa tutkimuksissa löydetyt piilotetut rajapinnat laitteen mobiiliverkkopinoon. Näistä rajapinnoista on paljastunut mm. sellaisia laitevalmistajan lisäämiä Hayes-komentoja, joilla on mahdollista ohittaa laitteen lukitus ja ylikirjoittaa laiteohjelmiston osia (esim. Hay 2017).

3.2 Ohjelmistovälitteiset hyökkäykset

Fyysinen pääsy laitteelle on vain harvoin mahdollista, joten jo laitevalmistajienkin intressien vuoksi laiteohjelmiston päivittäminen on yleensä mahdollista ohjelmallisesti. Työpöytäalustoilla tämän mahdollistaa monen prosessoriarkkitehtuurin tapa käsitellä oheislaitteita tavalla, josta käytetään nimitystä Memory Mapped IO. Toisin sanoen, eri oheislaitteet kartoitetaan osaksi fyysisiä muistialueita (Kovah ja Kallenberg 2020). Jos siis tiedetään flash-muistiohjaimen osoite, voidaan laiteohjelmistoon tehdä muutoksia.

Qualcommin järjestelmäpiirejä käyttävillä mobiilialustoilla päivittäminen on tätäkin helpompaa, sillä laiteohjelmisto on tallessa sisäisen muistin suojatussa osiossa (Wu ym. 2017). Tällöin pääkäyttäjän oikeudet riittävät laiteohjelmiston muokkaamiseen.

Useimmissa käyttöjärjestelmissä tavallinen ohjelma ei voi muokata suojattuja osioita eikä varsinkaan käyttää fyysisiä muistiosoitteita. Usein ei ole myöskään mahdollista ajaa oikeutetumpaa ohjelmaa suoraan järjestelmässä, varsinkin jos ohjelma hyödyntää rajapintoja, joiden käyttö vaatii ytimen tasoisia oikeuksia (Rutkowska 2006). Tästä huolimatta hyökkääjän on mahdollista käyttää suojattuja rajapintoja turvautumalla haavoittuvuuksiin ja mekanismeihin, jotka korottavat käytettävissä olevia oikeuksia (Perla ja Oldani 2010).

Fyysiseen hyökkäykseen verrattuna ohjelmalliset hyökkäykset eivät siis ole yhtä suoraviivaisia mutta mahdollistavat hyökkäämisen etäältä. Tämän lisäksi hyökkääjällä on käytössään potentiaalisia hyökkäsvektoreita paljon enemmän kuin fyysisessä hyökkäyksessä.

3.2.1 Laitevalmistajien päivitystyökalut

Yksi tapa laitevalmistajien päivitysten asentamiseen ovat päivitysohjelmat. Päivityksen tekemiseksi päivitysohjelmassa on oltava jokin mekanismi suojattujen rajapintojen käyttämiseksi. Aikaisemmissa tutkimuksissa näistä ohjelmista on löydetty laitevalmistajien omia työkaluja päivitysten tekemiseen, joista osa ei edes tarkista kirjoitetun datan eheyttä (Ermolov ja Goryachy 2018; Matrosov 2017). Kun hyökkääjällä on käytettävissään tällainen päivitystyökalu, voidaan hyökkäys toteuttaa ilman laiteohjelmistoon yltävää haavoittuvuusketjua ja siten vaadittava tekninen osaaminen vähenee huomattavasti.

3.2.2 Ajurien väärinkäyttö

Päivitystyökalujen lisäksi myös muita oikeutettuja ohjelmia, kuten järjestelmäajureita voidaan väärinkäyttää. Aiemmissa tutkimuksissa on löydetty useita laajasti käytettyjä ajureita, jotka sisältävät ulkopuolisen ohjelmakoodin suorituksen mahdollistavia haavoittuvuuksia (Eclypsium 2019). Haavoittuvaisia ajureita on löytynyt edellä mainittujen päivitysohjelmien lisäksi esimerkiksi myös laitteiston diagnostiikkatyökaluista sekä käyttöjärjestelmien omista ajureista (Augusto 2018; Corina ym. 2017). Joitain ajureita on myös kehitetty nimenomaan käyttöjärjestelmän suojausten kiertämistä varten (Microsoft 2020).

Etenkin Android-käyttöjärjestelmässä laiteajurien väärinkäyttöön liittyvä haavoittuvuusketju on yleinen tapa oikeuksien korottamiseen ja monesta Qualcommin laiteajurista on löytynyt tämän mahdollistavia haavoittuvuuksia (Mazuera-Rozo ym. 2019; Welton ja Grassi 2015).

3.2.3 Ketjulataamisen haavoittuvuudet

Käyttöjärjestelmä ei ole ainoa ohjelmallinen reitti laiteohjelmistoon, sillä haavoittuvuudet laiteohjelmiston käynnistysketjussa voivat mahdollistaa ulkopuolisen ohjelmakoodin lataamisen (Frazelle 2020b). Näin ladattava uhka on tietenkin helpommin poistettavissa kuin laiteohjelmistoon asennettu, mutta myös helpommin suoritettavissa.

Käynnistysketjun haavoittuvuudet ovat siinä määrin erityisiä, että ne eivät koske vain alustan toimittajan moduuleja. Itse asiassa yksi viimeisimmistä löydetyistä haavoittuvuuksista liittyy Linux-käyttöjärjestelmän hyödyntämään GRUB2-käynnistyslataajaan Eclypsium (2020d). Eclypsiumin tutkijoiden mukaan haavoittuvuusmekanismi on liian pitkä merkkijono lataajan konfiguraatiotiedostossa, mikä johtaa sen käsittelijämoduulin puskurin ylivuotamiseen, mahdollistaen ulkopuolisen ohjelmakoodin suorittamisen (Eclypsium 2020d). Löydön seurauksena myös Ubuntun tutkijat löysivät samasta lataajasta vastaavia haavoittuvuuksia (Murray 2021).

Latausketjun haavoittuvuuden ollessa käyttöjärjestelmän moduulissa uhkakehitys ei ole sidottu mihinkään tiettyyn laiteohjelmistoversioon. Tällöin mahdollisia kohteita voi olla paljon. Mikä pahempaa, laiteohjelmiston päivittäminen ei poista haavoittuvuutta joka ei ole laitevalmistajan omissa komponenteissa. Ketjulataamisen haavoittuvuudet lienevätkin tällä hetkellä parhaimpia ohjelmallisia hyökkäysvektoreita laiteohjelmistoon.

4 Havaittuja laiteohjelmistouhkia

Laiteohjelmistoihin kohdistuvia uhkia on ollut olemassa jo pitkään, mutta niiden hyödyntäminen kyberhyökkäyksissä näyttää uudelta ilmiöltä. Esimerkiksi ensimmäisenä UEFI-laiteohjelmistoon kohdistuvana hyökkäyksenä pidetään vuonna 2018 ESETin tutkijoiden löytämää LoJax-haittaohjelmaa (ESET 2018). LoJax hyväksikäyttää etenkin kannettavista tietokoneista löytyvää CompuTrace UEFI-mooduulia, muokaten laiteohjelmistoa siten, että CompuTracen suoritus saadaan hyökkääjän haltuun (ESET 2018). LoJax toimii kuitenkin vain sellaisissa järjestelmissä, joissa kirjoitussuojaukset ja Secure Boot ovat konfiguroimattomia (ESET 2018).

Vain pari vuotta LoJaxin ilmestymisen jälkeen Kasperskyn tutkijat löysivät MosaicRegressor-nimellä tunnetun modulaarisen uhan, johon sisältyy UEFI-bootkit (Kaspersky 2020). LoJaxin tapaan sekin muokkaa suoraan järjestelmän laiteohjelmistoa (Kaspersky 2020). MosaicRegressor on kuitenkin edeltäjäänsä edistyneempi siinä mielessä, että se kykenee kiertämään alustan kirjoitussuojaukset hyödyntäen haitallista SMM-moduulia (Eclypsium 2020b).

Hieman myöhemmin löydettiin vielä kolmas UEFI:in kohdistuva uhka Eclypsiumin tutkijoiden toimesta, TrickBot-troijalaiseen liitetty TrickBoot (Eclypsium ja ADVINTEL 2020). Toistaiseksi TrickBoot näyttää vain keräävän tietoa järjestelmästä, tarkistaen mm. ovatko laiteohjelmistojen kirjoitussuojaukset asetettu oikein ja mikä on järjestelmän piirisarja (Eclypsium ja ADVINTEL 2020). Kyseiset tiedot ovat tarpeellisia laiteohjelmistoon kirjoittamisen kannalta ja onkin todennäköistä että moduuli päivitetään myöhemmin kirjoituskykyiseksi (Eclypsium ja ADVINTEL 2020).

Keskenään havaituilla uhilla on paljon yhteistä. Ensinnäkin, kaikkien liittymisestä valtiollisiin toimijoihin on vahvaa näyttöä (Morley 2021; Eclypsium ja ADVINTEL 2020; ESET 2018). Toisekseen, ne kaikki hyödyntävät aikaisemmin julkaistujen ohjelmien valmiita komponentteja ja käyttävät pitkälti samoja mekanismeja järjestelmän haltuunottamiseen.

Yksi näistä ohjelmista on Italialaisen Hacking Team-yhtiön vuonna 2015 vuodettu Vector-EDK haittaohjelma, joka näyttäisi olevan keskeinen tekijä sekä LoJaxin että MosaicRegressorin ohjelmakoodissa. Vanhvimmin HackingTeamiin on yhdistetty kummankin uhkan käyttämä NTFS-ajuri, joka mahdollistaa Windows-tietojärjestelmien käsittelyn (Kaspersky 2020; ESET 2018).

Yhtäläisyydet eivät kuitenkaan lopu tähän. Kaikki näistä uhista nimittäin käyttävät matalan tason toiminnallisuuksien kuten laiteohjelmiston lukemisen ja kirjoittamisen toteuttamiseksi RWEverything-diagnostiikkaohjelman ajuria (Eclypsium ja ADVINTEL 2020; Kaspersky 2020; ESET 2018). Kyseinen ajuri on allekirjoitettu ja sisältää rajapinnan niin arvojen lukemiseen kuin kirjoittamiseen ja lisäksi sille on kirjoitettu avoimen lähteen rajapinta. Nämä seikat yhdessä selittänevät juuri kyseisen ajurin käytön kaikissa uhkissa.

Vaikka julkisesti tiedossa olevia UEFI-laiteohjelmistohyökkäyksiä on muutama, Qualcommin laiteohjelmistokokonaisuuteen kohdistuneita hyökkäyksiä ei etsinnöistä huolimatta löytynyt.

5 Uhkakartoituksen toteuttaminen

Eri kyberuhkiin mahdollisesti liittyvien haavoittuvuuksien kartoittamiseksi National Vulnerability Database käytiin läpi sekä työpöytä- että mobiilialustojen osalta useilla eri hakutermeillä. Kartoitettujen haavoittuvuuksien hyöty määräytyi sen mukaan, miten laajasti niitä on mahdollista hyödyntää eri laitteissa ja tuoteversioissa. Sellaiset haavoittuvuudet, jotka eivät mahdollistaneet pääsyä järjestelmään jätettiin katsauksen ulkopuolelle.

Kyberuhkia itsessään koottiin hyödyntäen aikaisempia tutkimuksia laiteohjelmistoturvallisuuteen liittyen ja niiden riskiarvio toteutettiin NIST SP 800-30 Rev. 1 mukaisia periaatteita noudattaen

Taulukko 1. National Vulnerability Databasen hakutermit

Työpöytäalustat	Mobiilialustat
United Extensible Firmware Interface	Firmware
UEFI	CAF
Firmware	CodeAuroraForum
SMM	MSM
System Management Mode	Qualcomm
Baseboard Management Controller	Little Kernel
BMC	LK
Intel ME	Aboot
Intel AMT	Android Bootloader
Management Engine	SBL
Active Management Technology	PBL
DRAC	XBL
MegaRAC	QTEE
iDRAC	QSEE
iLO	TrustZone
AST 2400	SMC
Supermicro	Secure Monitor
Nuvoton	Secure Monitor Mode

6 Johtopäätökset

Sekä työpöytä- että mobiilialustojen osalta tunnistettiin useita käynnistyslaiteohjelmistoihin kohdistuvia uhkia, joiden avulla tehtäviä hyökkäyksiä voi pitää realistisina. Sellaisia uhkia, joista on nähty vain yksittäisiä laitekohtaisia toteutuksia ei arvioitu. Myöskään oikean hyökkäystilanteen kannalta epärealistisia uhkia ei arvioitu.

6.1 Työpöytäalustat

Intel-arkkitehtuuriin liittyvää UEFI-laiteohjelmistoa koskevien uhkien arvioitiin koostuvan DXE-vaiheesta eteenpäin ladattavista moduuleista sekä etähallintaohjaimiin kohdistuvista uhista. DXE-vaihetta edeltävien alustusvaiheiden uhkia ei pidetty realistisina, sillä mahdollistavia haavoittuvuuksia tai toteutettuja hyökkäyksiä ei löytynyt yksittäisen esimerkkitutkimuksen lisäksi (Kallenberg ym. 2014).

6.1.1 Bootkit

Verrattuna muihin UEFI:in kohdistuviin uhkiin bootkit ei vaadi varsinaisen laiteohjelmiston muuttamista, minkä vuoksi sen toteuttaminen on muita uhkia helpompaa. Sekä käyttöjärjestelmän käynnistyslataajan korvaaminen, sen muokkaaminen että laiteohjelmistoajurin injektointi ovat mahdollisia tapoja toteuttaa bootkit (Harley 2014). On kuitenkin huomioitava, että käyttöjärjestelmän lataaja kutsuu UEFI:n ExitBootServices-rajapinnan, mikä lopettaa käynnistyspalvelut (Tianocore 2018). Bootkitillä on käytössään vain hyvin vähän palveluita tämän jälkeen, mikä rajoittaa sen käytettävyyttä. Tämäntyyppinen uhka ei myöskään selviydy järjestelmän alustuksesta.

National Vulnerability Databasesta ei löydetty mitään sellaista haavoittuvuutta, joka mahdollistaisi alustan CoT:n kiertämisen bootkitille. On siis epätodennäköistä, että uhka suorittuisi oikein konfiguroidulla alustalla. Näistä seikoista johtuen UEFI Bootkitin alustaan kohdistama riski voidaan arvioida matalaksi.

6.1.2 Käynnistyksenaikainen UEFI-moduuli

Bootkitistä seuraava taso ylöspäin on UEFI-ympäristössä suorittuva käynnistyksenaikainen moduuli. Moduulilla on mahdollisuus hyödyntää miltei kokonaisuudessaan UEFI-rajapintaa (Tianocore 2020a) ja siten esimerkiksi käyttää kaikkia ajurien alustamia laitteita järjestelmässä sekä lukea ja asettaa UEFI-muuttujia. Uhkan suorittaminen on mahdollista oikein konfiguroidussa järjestelmässä hyödyntäen aikaisemmin esitettyjä ketjulataamisen mahdollistavia haavoittuvuuksia. Käynnistyksenaikainen moduuli lakkaaa kuitenkin olemasta uhka siinä vaiheessa kun käyttöjärjestelmä on latautunut (Zimmer, Rothman ja Marisetty 2017), minkä vuoksi alustaan kohdistuva riski voidaan arvioida kohtalaiseksi.

6.1.3 Ajonaikainen UEFI-moduuli

Ajonaikainen UEFI-moduuli säilyy muistissa käyttöjärjestelmän käynnistymisen jälkeenkin, mutta se kykenee hyödyntämään vain pientä osaa niistä rajapinnoista, joita käynnistyksenaikainen moduuli voi hyödyntää (Zimmer, Rothman ja Marisetty 2017). Ajonaikainen moduuli voi esimerkiksi hallita järjestelmän virtatilaa ja lukea sekä asettaa UEFI-muuttujia. Myös ajonaikaisen moduulin suorittaminen on mahdollista oikein konfiguroidussa järjestelmässä hyödyntäen ketjulataamisen mahdollistavia haavoittuvuuksia. Käynnistyksenaikaiseen moduuliin verrattuna sellaisen kirjoittaminen on kuitenkin paljon haastavampaa, etenkin muistinhallintaan liittyvien ongelmien vuoksi (Tianocore 2018). Kaiken kaikkiaan haitallisen ajonaikaisen moduulin alustaan kohdistama riski voidaan arvioida malatalaksi.

6.1.4 SMM-moduuli

Vaikka System Management Modelle on varattu oma muistilohko keskusmuistista (SMRAM), suorittuu myös kyseisen muistialueen ulkopuolinen koodi toimintatavan oikeuksilla. Tätä ominaisuutta on voitu hyväksikäyttää haitallisen koodin suorittamiseen, kun UEFI:n SMI-käsittelijä on käsitellyt lohkon ulkopuolella olevia rakenteita, kuten UEFI-käynnistyspalvelutaulua (Bazhaniuk ym. 2015). Myös varsinaisen DXE-vaiheessa SMRAM-muistiin ladattavan SMM-ajurin kirjoittaminen on mahdollista. Ilmeisesti tällaista SMM-ajuria ei ole kuitenkaan onnistuttu aikaisemmin lataamaan sellaisenaan vaan ainoastaan toiseen ajuriin

liitettynä (Oleksiuk 2015). SMM-ajurin lataaminen ketjulataamiseen perustuvilla haavoittuvuuksilla ei myöskään onnistu, sillä SMRAM-muisti lukitaan ennen haavoittuvaisen lataajan suorittamista.

Ulkopuolinen koodi SMM-tilassa on kaikesta huolimatta vakava uhka, sillä mikään muistialue, rekisteri tai väylä ei ole sen ulottumattomissa. Lisäksi ulkopuolisella koodilla olisi vapaa pääsy SMRAM:in sisältöön, kuten luottamuksellisiin avaimiin. Käytännössä SMM-uhka voisi kiertää minkä tahansa alustan turvajärjestelyn, myös virtualisaatioon perustuvat. Lisäksi se olisi täysin näkymätön muulle järjestelmälle (Embleton, Sparks ja Zou 2013; Szefer ja Lee 2012). Sekä kirjallisuus että National Vulnerability Databasesta löydetyt haavoittuvuudet kuitenkin osoittavat, että SMM-hyökkäysvektorit ovat tähän asti olleet lähinnä laitekohtaisia. Hyökkäystä SMM-moduulilla voikin pitää epätodennnäköisenä, mutta sen alustaan kohdistama riski on silti arvioitava kohtalaiseksi seurausten vakavuudesta johtuen.

6.1.5 CSME-uhka

Aikaisemmat tutkimukset ovat osoittaneet, että CSME:n laiteohjelmistoa vastaan hyökkääminen on mahdollista hyödyntäen samoja fyysisiä ja ohjelmallisia mekanismeja, joilla voidaan hyökätä UEFI-laiteohjelmistoa vastaan. On pystytty esimerkiksi osoittamaan, että CSME:n etähallintaominaisuudet voidaan ottaa käyttöön tietyissä niitä tukemattomattomissa piirisarjoissa (Ermolov ja Goryachy 2017). Tällaista haavoittuvuutta hyväksikäyttämällä hyökkääjä voi AMT:n etähallintatoimintojen avulla poistaa käytöstä alustan CoT:n ja kirjoittaa UEFI-laiteohjelmistoon mitä vain.

Myös valmiiksi konfiguroituja vPro-alustoja vastaan voi hyökätä luotettavasti. Haku National Vulnerability Databasesta osoitti, että jokaista AMT-versiota vastaan on olemassa järjestelmän haltuunoton mahdollistavia haavoittuvuuksia. CSME:n ja sen sovellusten kautta tapahtuvan hyökkäyksen alustaan kohdistama riski voidaan siten arvioida korkeaksi.

6.1.6 BMC-uhka

BMC:n käyttömahdollisuudet hyökkäystilanteessa ovat pitkälti Intel AMT:n kaltaisia. Edelliseen verrattuna riskiprofiilia kuitenkin nostaa Cipher Zeron kaltaiset "ominaisuushaavoittuvuudet" hallintaohjainten käyttämissä protokollissa. Käytettäessä haavoittuvaista protokollaa, laitteen valmistajalla tai tuoteversiolla ei ole väliä. Tilanne on entistä huonompi kun protokollalla vai hallita käytännössä kaikkea järjestelmässä, jopa päivittää laiteohjelmiston suoraan (Intel 2020d). Pahinta kuitenkin on, että tietyt laitevalmistajat ovat aiemmin suhtautuneet ilmoitettuihin haavoittuvuuksiin vähätellen (esim. NIST 2013). BMC-ohjaimeen kohdistuvaa uhkaa on syytä pitää vakavana hyökkäystapana ja sellaisen alustaan kohdistama riski voidaan arvioida erittäin korkeaksi.

Taulukko 2. Työpöytäalustojen kyberuhat

Uhka	Todennäköisyys	Vakavuus	Kokonaisriski
BMC-uhka	todennäköinen	katastrofaaalinen	erittäin korkea
CSME-uhka	satunnainen	katastrofaalinen	korkea
SMM-moduuli	epätodennäköinen	katastrofaalinen	kohtalainen
UEFI-moduuli (BT)	todennäköinen	vakava	kohtalainen
UEFI-moduuli (RT)	satunnainen	vähäinen	matala
Bootkit	harvinainen	vakava	matala

6.2 Mobiilialustat

Qualcommin laiteohjelmistokokonaisuutta hyödyntäviä ARM-arkkitehtuurin mobiilialustoja vastaan tunnistettiin joitain uhkia. Näistä kahden arvioitiin liittyvän käynnistysketjun viimeiseen lataajaan, Abootiin. Lisäksi tunnistettiin QTEE:n turvallisiin sovelluksiin liittyvä uhka. Sekä Abootia edeltäviin lataajiin että Secure Monitoriin kohdistuvia uhkia pidettiin epärealistisina. Secure Monitoriin liittyen on tehty tutkimuksia (esim. Guilbon 2018a), mutta niissä käytetyn haavoittuvuuden lisäksi ei löydetty muita Secure Monitorin haltuunoton mahdollistavia haavoittuvuuksia. Eräässä toisessa tutkimuksessa laitteen koko käynnistysketju saatiin

haltuun hyödyntämällä EDL-tilaa. Tulosten yleistettävyyttä voi kuitenkin pitää huonona, sillä hyökkäysvektorina oli laitevalmistajan lisäämä funktio (Aleph Security 2018b).

6.2.1 Aboot-Bootkit

Joillain laitteilla Abootin lukitus voidaan avata niin, ettei kernelin eheyttä varmisteta (Hay 2017). Hyökkääjä voi hyväksikäyttää tätä ominaisuutta bootkitin asettamiseksi järjestelmään. Lukituksen avaaminen on kuitenkin yleensä mahdollista vain laitteen kehittäjätilassa, kun laitteelle on fyysinen pääsy. Lisäksi avaaminen poistaa käyttäjän datan laitteelta (Hay 2017), joten hyökkääjän tulisi kyetä palauttamaan datan sisältävä osio, jotta hyökkäys olisi huomaamaton.

Mikäli Aboot on jo valmiiksi avattu, voisi hyökkääjä toteuttaa vastaavan hyökkäyksen pääkäyttäjän oikeuksilla etänä tai paikallisesti. Näistä seikoista johtuen voidaan todeta, että hyökkäys on hankalasti toteutettava eikä se ole helposti yleistettävissä suureen määrään laitteita. Tätä näkemystä tukee se, että oletettavasti ainoa julkisesti tunnettu Android-bootkit on monen vuoden takaa. On myös vahvaa näyttöä siitä, että kyseinen bootkit oli päätynyt laitteisiin suoraan toimitusketjusta (Xiao ym. 2014). Abootin lataaman bootkitin alustaan kohdistama riski arvioidaan matalaksi.

6.2.2 Abootin koodi-injektio

National Vulnerability Databasesta löydettiin haavoittuvuuksia liittyen aikaisempiin Little Kernelin ja Abootin versioihin, jotka mahdollistavat ulkopuolisen koodin suorittamisen erilaisia vektoreita hyödyntäen. On siis oletettavissa, että jos alustan turvallisen Aboot-version saa korvattua allekirjoitetulla haavoittuvaisella versiolla, voi sitä hyödyntäen ladata ajonaikaisen uhkan. Tällainen uhka suorittuisi samoilla oikeuksilla kuin itse Aboot. Hyökkäyksen käytännöllisyyttä kuitenkin rajoittaa Abootin kapea syötepinta. Uhka voitaisiin ladata joko fyysisesti fastboot-protokollan kautta tai ohjelmallisesti seuraavaksi ladattavan käyttöjärjestelmän mukana. Koodi-injektion alustaan kohdistama riski arvioidaan kohtalaiseksi.

6.2.3 Trustletin koodi-injektio

Qualcommin turvallinen suoritusympäristö varmistaa Trustlettien eheyden ennen niiden lataamista (Guilbon 2018b). Tästäkin huolimatta trustletteihin on onnistuttu tekemään koodi-injektioita hyödyntäen haavoittuvuuksia niissä ja niiden lataajissa (Chen ym. 2017). Kuten Abootin injektoinnissa, Trustletteja on myös onnistuttu vaihtamaan haavoittuvaisiin versioihin hyökkäyksen toteuttamiseksi (Chen ym. 2017).

Sekä TrustZoneen että Qualcommin turvalliseen suoritusympäristöön liittyen löytyi useita haavoittuvuuksia National Vulnerability Databasesta. Haavoittuvuuksien kuvauksiin perustuen suurin osa niistä näyttää vaikuttavan vain muutamaan laitteeseen tai järjestelmäpiiriin. Lisäksi on epäselvää, mikä merkitys tietyn haavoittuvuuden hyödyntämisellä on kokonaisturvallisuuden kannalta, sillä ladattu Trustlet ei normaalisti pääse käsiksi muiden Trustlettien muistiin (Makkaveev 2019). Kun huomioidaan, että injektoitu Trustlet on rajattu tiettyyn muistialueeseen, suorittuu alhaisimmalla poikkeustasolla (Guilbon 2018a) eikä välttämättä lataudu kovin monella laitteella, voidaan alustaan kohdistuvaa riskiä pitää kohtalaisena.

Taulukko 3. Mobiilialustojen kyberuhat

Uhka	Todennäköisyys	Vakavuus	Kokonaisriski
Ajonaikainen injektio (Trustlet)	satunnainen	vakava	kohtalainen
Ajonaikainen injektio (Aboot)	satunnainen	kriittinen	kohtalainen
Bootkit	harvinainen	vakava	matala

6.3 Päätelmät

Tulosten valossa Qualcommin järjestelmäpiirejä hyödyntävät mobiilialustat ovat turvallisempia kuin Intel-arkkitehtuuria noudattavat työpöytäalustat. Sen lisäksi että mobiilialustojen realistisia uhkia löytyi vähemmän, oli niiden muodostama riski pienempi kuin työpöytäalustojen uhkien. Vaikuttaisi siltä, että UEFI:n laajennettavuus suhteessa Qualcommin laiteohjelmistoon helpottaa hyökkäyksen toteuttamista. Samoin etähallintaohjaimet näyttävät

suurentavan työpöytäalustojen hyökkäyspintaa tarpeettomasti mobiilialustoihin verrattuna.

7 Yhteenveto

Tässä tutkielmassa tarkasteltiin moderneissa järjestelmissä yleisesti hyödynnettävien käynnistyslaiteohjelmistojen turvallisuutta. Aihepiiristä on tehty tutkimuksia aiemminkin, mutta ilmeisesti yksikään tutkimus ei ole vielä tarkastellut eri uhkilla toteutettavien hyökkäysten realistisuutta. Parhaan saatavilla olevan tiedon mukaan aikaisemmissa tutkimuksessa ei ole myöskään tehty vertailua eri alustatyyppien käynnistyslaiteohjelmistojen turvallisuuteen liittyen. Tutkielmaa voidaan siis pitää johdatuksena näihin aiheisiin. Saatujen tulosten yleistettävyys ei kuitenkaan ole mahdollista alustakohtaisista vaihteluista johtuen.

Työpöytäalustoihin liittyvät tulokset soveltuvat CSME:n osuutta lukuun ottamatta kaikille x86-, ja x86_64-arkkitehtuurien laitteille, joiden laiteohjelmisto on UEFI. Kattavuutta voidaan siis pitää hyvänä. Mobiilialustoilla tilanne on hajonnan johdosta toinen, Qualcommin järjestelmäpiirien markkinaosuuden ollessa noin 31%. Tulosten perustaminen julkisiin haavoittuvuuksiin tutkimuskirjallisuuden ohella tuo mukanaan myös tiettyjä ongelmia. Julkisen haavoittuvuusdatan saatavuus oli Qualcommin komponenttien kohdalla hyvä, mutta eri UEFI-toteutusten kohdalla huono. Erityisen hälyttävää oli, että julkisia Tianocoreen liittyviä haavoittuvuuksia löydettiin vain 12 kappaletta koko tarkastelun ajalta, vaikka Intel vahvisti yksistään vuosina 2015-2017 yhteensä 77 referenssitoteutukseen vaikuttavaa haavoittuvuutta (Monroe, Branco ja Zimmer 2017).

On siis mahdollista, että UEFI-laiteohjelmiston uhat ovat tosiasiallisesti arvioitua korkeampi riski alustaa kohtaan ja asiaa olisi syytä tutkia enemmän. Tämän lisäksi olisi tärkeää selvittää mitä UEFI-laiteohjelmistoon perustuva käynnistysketju tarkoittaa mobiilialustojen turvallisuudelle. Jo nyt uudempien laitteiden XBL hyödyntää UEFI:n PI-moduuleja ja lisäksi on vahvoja viitteitä siitä, että Aboot on muuttumassa tai on jo muuttunut LK:n johdannaisesta UEFI-moduuleja hyödyntäväksi ratkaisuksi (CodeAuroraForum 2021).

Lähteet

Aleph Security. 2018a. “Exploiting Qualcomm EDL Programmers (1): Gaining Access PBL Internals”. Viitattu 26. huhtikuuta 2021. https://alephsecurity.com/2018/01/22/qualcomm-edl-1/.

———. 2018b. “Qualcomm EDL Firehose Programmers Peek and Poke Primitives”. Viitattu 7. huhtikuuta 2021. https://alephsecurity.com/vulns/aleph-2017028.

Alsop, Thomas. 2020. “Smartphone application processor (AP) market vendor revenue share worldwide from 2014 to 2020”. Viitattu 25. huhtikuuta 2021. https://www.statista.com/statistics/233415/global-market-share-of-applications-processor-suppliers.

———. 2021. “Distribution of Intel and AMD x86 computer central processing units (CPUs) worldwide from 2012 to 2021, by quarter”. Viitattu 25. huhtikuuta 2021. https://www.statista.com/statistics/735904/worldwide-x86-intel-amd-market-share/.

AMD. 2021. *AMD64 Architecture Programmer’s Manual: Volumes 1-5.* 467. Viitattu 22. helmikuuta 2021. https://www.amd.com/system/files/TechDocs/40332.pdf.

Analog Devices. 2007. “In-Circuit Programming of an SPI Flash with SHARC® Processors: Engineer-to-Engineer Note”. Viitattu 27. huhtikuuta 2021. https://www.analog.com/media/en/technical-documentation/application-notes/EE.231.Rev.2.08.07.pdf.

Appere, Elouan. 2019. “Analysis of Qualcomm Secure Boot Chains”. Viitattu 7. huhtikuuta 2021. https://blog.quarkslab.com/analysis-of-qualcomm-secure-boot-chains.html.

ARM. 2021. *Arm® Architecture Reference Manual: Armv8, for Armv8-A architecture profile.* 2436, 5956. Viitattu 10. huhtikuuta 2021. https://documentation-service.arm.com/static/60119835773bb020e3de6fee.

Augusto, Otávio. 2018. “SIOCtl: Simple IOCTL dispatcher”. Viitattu 28. maaliskuuta 2021. https://github.com/otavioarj/SIOCtl.

Bazhaniuk, Oleksandr, Yuriy Bulygin, Andrew Furtak, Mikhail Gorobets, John Loucaides, Alexander Matrosov ja Mickey Shkatov. 2015. “A New Class of Vulnerabilities in SMI Handlers”. Viitattu 28. maaliskuuta 2021. https://cansecwest.com/slides/2015/A%5C%20New%5C%20Class%5C%20of%5C%20Vulnin%5C%20SMI%5C%20-%5C%20Andrew%5C%20Furtak.pdf.

Belman-Flores, J.M., J.M. Barroso-Maldonado, A.P. Rodríguez-Muñoz ja G. Camacho-Vázquez. 2015. “Enhancements in domestic refrigeration, approaching a sustainable refrigerator – A review”. *Renewable and Sustainable Energy Reviews* 51:955–968. ISSN: 1364-0321. https://doi.org/https://doi.org/10.1016/j.rser.2015.07.003. https://www.sciencedirect.com/science/article/pii/S1364032115006504.

Carey, Mark, ja Rob Bathurst. 2013. “Hacking Embedded Devices: Doing Bad Things to Good Hardware”. Viitattu 10. maaliskuuta 2021. https://www.defcon.org/images/defcon-21/dc-21-presentations/Phorkus-Evilrob/DEFCON-21-Phorkus-Evilrob-Hacking-Embedded-Devices-Bad-things-to-Good-hardware.pdf.

Chen, Tien-He, ja Che-Min Chen. 2020. Power-up control circuit and mobile power bank. Yhdysvaltalainen patentti US010545550B2, haettu 28. tammikuuta 2020.

Chen, Yue, Yulong Zhang, Zhi Wang ja Tao Wei. 2017. *Downgrade Attack on TrustZone*. arXiv: 1707.05082 [cs.CR].

CodeAuroraForum. 2021. “index : abl/tianocore/edk2”. Viitattu 12. huhtikuuta 2021. https://source.codeaurora.org/quic/la/abl/tianocore/edk2/.

Coreboot. 2021a. “Dell OptiPlex 9010”. Viitattu 26. huhtikuuta 2021. https://doc.coreboot.org/mainboard/dell/optiplex_9010.html.

———. 2021b. “Flashing firmware externally supplying no power”. Viitattu 28. huhtikuuta 2021. https://doc.coreboot.org/flash_tutorial/no_ext_power.html.

———. 2021c. “Flashing firmware tutorial”. Viitattu 28. huhtikuuta 2021. https://doc.coreboot.org/flash_tutorial/index.html.

Corina, Jake, Aravind Machiry, Christopher Salls, Yan Shoshitaishvili, Shuang Hao, Christopher Kruegel ja Giovanni Vigna. 2017. “DIFUZE: Interface Aware Fuzzing for Kernel Drivers”. Teoksessa *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security,* 2123–2138. CCS ’17. Dallas, Texas, USA: Association for Computing Machinery. ISBN: 9781450349468. https://doi.org/10.1145/3133956.3134069. https://doi.org/10.1145/3133956.3134069.

Costin, Andrei, Jonas Zaddach, Aurélien Francillon ja Davide Balzarotti. 2014. “A large scale analysis of the security of embedded firmwares”. Teoksessa *USENIX Security 2014, 23rd USENIX Security Symposium, August 20-22, 2014, San Diego, USA,* toimittanut Usenix. Copyright Usenix. Personal use of this material is permitted. The definitive version of this paper was published in USENIX Security 2014, 23rd USENIX Security Symposium, August 20-22, 2014, San Diego, USA and is available at : San Diego.

David, Yaniv, Nimrod Partush ja Eran Yahav. 2018. “FirmUp: Precise Static Detection of Common Vulnerabilities in Firmware”. *SIGPLAN Not.* (New York, NY, USA) 53, numero 2 (maaliskuu): 392–404. ISSN: 0362-1340. https://doi.org/10.1145/3296957.3177157. https://doi-org.ezproxy.jyu.fi/10.1145/3296957.3177157.

Dell. 2008. “Integrated Dell™ Remote Access Controller Firmware Version 1.00 User Guide”. Viitattu 9. maaliskuuta 2021. %7Bhttps://downloads.dell.com/manuals/all-products/esuprt_electronics/esuprt_software/esuprt_remote_ent_sys_mgmt/integrated-dell-remote-access-cntrllr-6-for-blade-srvr-v1.0_user%5C%27s%5C%20guide_en-us.pdf%7D.

Dent, Alexander W. 2019. “Secure Boot and Image Authentication: Technical Overview (v2.0)”. Viitattu 7. huhtikuuta 2021. https://www.qualcomm.com/media/documents/files/secure-boot-and-image-authentication-technical-overview-v2-0.pdf.

Eclypsium. 2019. “SCREWED DRIVER — SIGNED, SEALED, DELIVERED: Common Design Flaw In Dozens of Device Drivers Allows Widespread Windows Compromise”. Viitattu 28. maaliskuuta 2021. https://eclypsium.com/wp-content/uploads/2019/08/Screwed-Drivers.pdf.

———. 2020a. “Anatomy of a Firmware Attack”. Viitattu 8. maaliskuuta 2021. https://eclypsium.com/wp-content/uploads/2020/09/Anatomy-of-a-Firmware-Attack-2020.pdf.

Eclypsium. 2020b. “MosaicRegressor: PROTECT YOUR ORGANIZATION FROM MOSAICREGRESSOR AND OTHER UEFI IMPLANTS”. Viitattu 30. maaliskuuta 2021. https://eclypsium.com/wp-content/uploads/2020/10/Protecting-Your-Organizations-From-MosaicRegressor.pdf.

———. 2020c. “Perilious peripherals: The Hidden Dangers Inside Windows Linux Computers”. Viitattu 1. maaliskuuta 2021. https://eclypsium.com/wp-content/uploads/2020/02/Eclypsium-Unsigned-Peripheral-Firmware-Research.pdf.

———. 2020d. “There’s a Hole in the Boot”. Viitattu 6. maaliskuuta 2021. https://eclypsium.com/wp-content/uploads/2020/08/Theres-a-Hole-in-the-Boot.pdf.

Eclypsium ja ADVINTEL. 2020. “TRICKBOT NOW OFFERS ‘TRICKBOOT’: PERSIST, BRICK, PROFIT: Researchers discover a new module in the TrickBot toolset aimed at detecting UEFI / BIOS firmware vulnerabilities”. Viitattu 30. maaliskuuta 2021. https://eclypsium.com/wp-content/uploads/2020/12/TrickBot-Now-Offers-TrickBoot-Persist-Brick-Profit.pdf.

Embleton, Shawn, Sherri Sparks ja Cliff C Zou. 2013. “SMM rootkit: a new breed of OS independent malware”. *Security and Communication Networks* 6 (12): 1590–1605.

Ermolov, Mark, ja Maxim Goryachy. 2017. “How to Hack a Turned-Off Computer, or Running Unsigned Code in Intel Management Engine”. https://www.blackhat.com/docs/eu-17/materials/eu-17-Goryachy-How-To-Hack-A-Turned-Off-Computer-Or-Running-Unsigned-Code-In-Intel-Management-Engine.pdf.

———. 2018. “Intel Management Engine JTAG Proof of Concept”. Viitattu 28. huhtikuuta 2021. https://github.com/ptresearch/IntelTXE-PoC.

ESET. 2018. “LOJAX: First UEFI rootkit found in the wild, courtesy of the Sednit group”. Viitattu 30. maaliskuuta 2021. https://www.eset.com/fileadmin/ESET/US/resources/datasheets/ESETus-datasheet-lojax.pdf.

Fish, Andrew. 2012. “Multithreading in EDK2.0”. Viitattu 25. huhtikuuta 2021. https://edk2-devel.narkive.com/hBHaeJVo/multithreading-in-edk2-0.

Frazelle, Jessie. 2020a. “Opening up the Baseboard Management Controller”. *Commun. ACM* (New York, NY, USA) 63, numero 2 (tammikuu): 38–40. ISSN: 0001-0782. https://doi.org/10.1145/3369758. https://dl.acm.org/doi/pdf/10.1145/3371595.3378404.

———. 2020b. “Securing the Boot Process”. *Commun. ACM* (New York, NY, USA) 63, numero 3 (helmikuu): 38–42. ISSN: 0001-0782. https://doi.org/10.1145/3379512. https://doi-org.ezproxy.jyu.fi/10.1145/3379512.

Geiselbrecht, Travis. 2018. “Travis Geiselbrecht’s Home Page”. Viitattu 26. huhtikuuta 2021. http://tkgeisel.com/.

Giri, Aparna, Doug Iler ja Jeff Krebs. 2020. “In-band or out-of-band: Advantages of iDRAC and iSM compared to OMSA”. Viitattu 26. huhtikuuta 2021. https://downloads.dell.com/manuals/common/dell-emc-sysmgmt-inband-outofband-idrac-and-ism-vs-omsa.pdf.

Google. 2012. “FastBoot Version 0.4”. Viitattu 9. maaliskuuta 2021. https://android.googlesource.com/platform/system/core/+/9bfecb0e3444306ec57d7fafe4a99a47d3848c17/fastboot/fastboot_protocol.txt.

Grassi, Marco, Muqing Liu ja Tianyi Xie. 2018. “Exploitation of a Modern Smartphone Baseband”. *BlackHat US.*

Guilbon, Joffrey. 2018a. “Attacking the ARM’s TrustZone”. Viitattu 28. huhtikuuta 2021. https://blog.quarkslab.com/attacking-the-arms-trustzone.html.

———. 2018b. “Introduction to Trusted Execution Environment: ARM’s TrustZone”. Viitattu 26. huhtikuuta 2021. https://blog.quarkslab.com/introduction-to-trusted-execution-environment-arms-trustzone.html.

Harley, Eugene Rodionov Alexander Matrosov David. 2014. “Bootkits: Past, Present Future”. *Virus Bulletin.—-2014.*

Hay, Roee. 2017. “fastboot oem vuln: Android Bootloader Vulnerabilities in Vendor Customizations”. Teoksessa *11th USENIX Workshop on Offensive Technologies (WOOT 17).* Vancouver, BC: USENIX Association, elokuu. https://www.usenix.org/system/files/conference/woot17/woot17-paper-hay.pdf.

Heiland, Deral. 2019. “[IoT Security] Introduction to Embedded Hardware Hacking”. Viitattu 26. huhtikuuta 2021. https://www.rapid7.com/blog/post/2019/02/20/iot-security-introduction-to-embedded-hardware-hacking/.

IEEE. 2013. *IEEE Standard for Test Access Port and Boundary-Scan Architecture.* 1–20. https://doi.org/10.1109/IEEESTD.2013.6515989.

Intel. 2005. “Intel® Active Management Technology Basics”. Viitattu 9. maaliskuuta 2021. https://www.supermicro.com/manuals/other/AMT_Basics.pdf.

———. 2009. *Intel® X58 Express Chipset: Datasheet.* 39. Viitattu 26. huhtikuuta 2021. https://www.intel.com/content/dam/doc/datasheet/x58-express-chipset-datasheet.pdf.

———. 2013. *Intel® C600 Series Chipset and Intel® X79 Express Chipset: Datasheet.* 52, 78, 256–260, 414–415. Viitattu 10. maaliskuuta 2021. https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/c600-series-chipset-datasheet.pdf.

———. 2020a. *Intel® 64 and IA-32 Architectures Software Developer’s Manual: Combined Volumes:1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D and 4.* 4165–4194. Viitattu 22. helmikuuta 2021. https://software.intel.com/content/dam/develop/external/us/en/documents-tps/325462-sdm-vol-1-2abcd-3abcd.pdf.

———. 2020b. “Intel® AMT Basic Concepts”. https://software.intel.com/content/www/us/en/develop/documentation/amt-developer-guide/top/getting-started/basic-concepts.html.

———. 2020c. “Intel® Converged Security and Management Engine (Intel® CSME): Security White Paper”. Viitattu 9. maaliskuuta 2021. https://www.intel.com/content/dam/www/public/us/en/security-advisory/documents/intel-csme-security-white-paper.pdf.

———. 2020d. *Intel® Server Products and SolutionsIntel® Integrated Baseboard Management Controller Embedded Web Server - User Guide: Guide to Intel® Integrated Baseboard Management Controller Embedded Web Server for Intel® Server Boards and Systems based on Intel® Xeon® Scalable processor family.* 9–11, 77–78. Viitattu 29. huhtikuuta 2021. https://www.intel.com/content/dam/support/us/en/documents/server-products/server-boards/Purley_RMM4_BMC_User_Guide.pdf.

Intel. 2020e. *Intel® Server System R1000WF Product Family Technical Product Specification: An overview of product features, functions, architecture, and support specifications.* 15. Viitattu 10. maaliskuuta 2021. https://www.intel.com/content/dam/support/us/en/documents/server-products/server-boards/R1000WF_TPS.pdf.

Kaczmarek, Śebastien. 2013. "UEFI and Dreamboot". Viitattu 26. huhtikuuta 2021. https://conference.hitb.org/hitbsecconf2013ams/materials/D2T1%5C%20-%5C%20Sebastien%5C%20Kaczmarek%5C%20-%5C%20Dreamboot%5C%20UEFI%5C%20Bootkit.pdf.

Kallenberg, Corey, Xeno Kovah, John Butterworth ja Sam Cornwell. 2014. "Extreme Privilege Escalation on Windows 8/UEFI Systems". Viitattu 26. huhtikuuta 2021. https://www.mitre.org/sites/default/files/publications/14-2221-extreme-escalation-presentation.pdf.

Kaspersky. 2020. "MosaicRegressor: Lurking in the Shadows of UEFI: Technical details". Viitattu 30. maaliskuuta 2021. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2020/10/07080558/MosaicRegressor_Technical-details.pdf.

Kovah, Xeno, ja Corey Kallenberg. 2020. "Advanced x86: BIOS and System Management Mode Internals Input/Output". Viitattu 28. huhtikuuta 2021. https://opensecuritytraining.info/IntroBIOS_files/Day1_04_Advanced%5C%20x86%5C%20-%5C%20BIOS%5C%20and%5C%20SMM%5C%20Internals%5C%20-%5C%20IO.pdf.

Laittice Semiconductor. 2017. "Programming External SPI Flash through JTAG for ECP5/ECP5-5G". Viitattu 28. huhtikuuta 2021. http://www.latticesemi.com/-/media/LatticeSemi/Documents/ApplicationNotes/PT2/FPGATN02050ProgrammingExtSPIFlashJTAGECP55G.ashx?document_id=52228.

Mahate, Shakeel. 2016. "Introduction". Viitattu 9. maaliskuuta 2021. https://github.com/littlekernel/lk/wiki/Introduction/cc107fff050145a440bc041e3ae50f85c7d425fb.

Makkaveev, Slava. 2019. "The Road to Qualcomm TrustZone Apps Fuzzing". Viitattu 28. huhtikuuta 2021. https://research.checkpoint.com/2019/the-road-to-qualcomm-trustzone-apps-fuzzing/.

Matrosov, Alex. 2017. “UEFI Ransomware: Full Disclosure at Black Hat Asia”. Viitattu 28. huhtikuuta 2021. https://blogs.blackberry.com/en/2017/03/uefi-ransomware-full-disclosure-at-black-hat-asia.

Mazuera-Rozo, Alejandro, Jairo Bautista-Mora, Mario Linares-Vásquez, Sandra Rueda ja Gabriele Bavota. 2019. “The Android OS stack and its vulnerabilities: an empirical study”. *Empirical Software Engineering* 24 (4): 2056–2101.

Microsoft. 2017. “BCD System Store Settings for UEFI”. Viitattu 25. huhtikuuta 2021. https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/bcd-system-store-settings-for-uefi.

———. 2020. “Secured-core PCs: A brief showcase of chip-to-cloud security against kernel attacks”. Viitattu 28. maaliskuuta 2021. https://www.microsoft.com/security/blog/2020/03/17/secured-core-pcs-a-brief-showcase-of-chip-to-cloud-security-against-kernel-attacks/.

Monroe, Bruce, Rodrigo Rubira Branco ja Vincent Zimmer. 2017. “Firmware is the new Black –Analyzing Past 3 years of BIOS/UEFI Security Vulnerabilities”. Viitattu 11. huhtikuuta 2021. https://raw.githubusercontent.com/rrbranco/BlackHat2017/master/BlackHat2017-BlackBIOS-v0.13-Published.pdf.

Morley, Connor. 2021. “Shining a light on UEFI – the hidden memory space being exploited in attacks”. *Network Security* 2021 (1): 14–17. ISSN: 1353-4858. https://doi.org/https://doi.org/10.1016/S1353-4858(21)00009-X. http://www.sciencedirect.com/science/article/pii/S135348582100009X.

Mrazek, Deborah, ja Colin Bay. 2020. “INTEL vPro® vs. AMD® PRO* OUT-OF-BAND MANAGEMENT”. Viitattu 26. huhtikuuta 2021. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/vpro-vs-amd-pro-out-of-band-management-white-paper.pdf.

Murray, Alex. 2021. “GRUB2 Secure Boot Bypass 2021”. Viitattu 7. huhtikuuta 2021. https://ubuntu.com/blog/grub2-secure-boot-bypass-2021.

Ning, Zhenyu, Fengwei Zhang, Weisong Shi ja Weidong Shi. 2017. “Position Paper: Challenges Towards Securing Hardware-Assisted Execution Environments”. Teoksessa *Proceedings of the Hardware and Architectural Support for Security and Privacy.* HASP ’17. Toronto, ON, Canada: Association for Computing Machinery. ISBN: 9781450352666. https://doi.org/10.1145/3092627.3092633. https://doi.org/10.1145/3092627.3092633.

NIST. 2013. “CVE-2013-4783 Detail”. https://nvd.nist.gov/vuln/detail/CVE-2013-4783.

Nixawk. 2017. “Intel Active Management Technology - System Privileges”. Viitattu 7. huhtikuuta 2021. https://www.exploit-db.com/exploits/43385.

Nystrom, Magnus, Martin Nicoles ja Vincent Zimmer. 2011. “UEFI Networking and Pre-OS Security”. *Intel Technology Journal* 15 (lokakuu): 80–1.

Oleksiuk, Dmytro. 2015. “Building reliable SMM backdoor for UEFI based platforms”. Viitattu 26. huhtikuuta 2021. http://blog.cr4.sh/2015/07/building-reliable-smm-backdoor-for-uefi.html.

———. 2016. “PEI stage backdoor for UEFI compatible firmware”. Viitattu 26. huhtikuuta 2021. https://github.com/Cr4sh/PeiBackdoor.

Oshana, Robert. 2013. *Software Engineering for Embedded Systems: Methods, Practical Techniques, and Applications.* 540.

Papp, Dorottya, Zhendong Ma ja Levente Buttyan. 2015. “Embedded systems security: Threats, vulnerabilities, and attack taxonomy”. Teoksessa *2015 13th Annual Conference on Privacy, Security and Trust (PST),* 145–152. https://doi.org/10.1109/PST.2015.7232966.

Perla, Enrico, ja Massimiliano Oldani. 2010. *A guide to kernel exploitation: attacking the core.* 3–19. Elsevier.

Positive Technologies. 2017. “Disabling Intel ME 11 via undocumented mode”. http://blog.ptsecurity.com/2017/08/disabling-intel-me.html.

Qualcomm. 2016. “DragonBoard™ 410c based on Qualcomm® Snapdragon ™ 410E processor: Little Kernel Boot Loader Overview”. Viitattu 7. huhtikuuta 2021. https://developer.qualcomm.com/qfile/28821/lm80-p0436-1_little_kernel_boot_loader_overview.pdf.

Rapid7. 2013. “A Penetration Tester’s Guide to IPMI and BMCs”. Viitattu 7. huhtikuuta 2021. https://blog.rapid7.com/2013/07/02/a-penetration-testers-guide-to-ipmi/.

Redini, Nilo, Aravind Machiry, Dipanjan Das, Yanick Fratantonio, Antonio Bianchi, Eric Gustafson, Yan Shoshitaishvili, Christopher Kruegel ja Giovanni Vigna. 2017. “BootStomp: On the Security of Bootloaders in Mobile Devices”. Teoksessa *26th USENIX Security Symposium (USENIX Security 17),* 781–798. Vancouver, BC: USENIX Association, elokuu. ISBN: 978-1-931971-40-9. https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-redini.pdf.

Regenscheid, Andrew. 2018. *Platform Firmware Resiliencyy Guidelineds: NIST Special Publication 800-193.* Viitattu 12. huhtikuuta 2021. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-193.pdf.

Rose, G. A. 1967. ““Intergraphic,” A Microprogrammed Graphical-Interface Computer”. *IEEE Transactions on Electronic Computers* EC-16 (6): 773–784. https://doi.org/10.1109/PGEC.1967.264723.

Rosenfeld, K., ja R. Karri. 2010. “Attacks and Defenses for JTAG”. *IEEE Design Test of Computers* 27 (1): 36–47. https://doi.org/10.1109/MDT.2010.9.

Rutkowska, Joanna. 2006. “Subverting Vista Kernel For Fun And Profit”. Viitattu 28. huhtikuuta 2021. https://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf.

Senrio. 2016. “JTAG Explained (finally!): Why "IoT", Software Security Engineers, and Manufacturers Should Care”. Viitattu 28. huhtikuuta 2021. https://blog.senr.io/blog/jtag-explained.

Skorobogatov, Sergei. 2012. “Physical Attacks and Tamper Resistance”. Teoksessa *Introduction to Hardware Security and Trust,* toimittanut Mohammad Tehranipoor ja Cliff Wang, 143–173. New York, NY: Springer New York. ISBN: 978-1-4419-8080-9. https://doi.org/10.1007/978-1-4419-8080-9_7. https://doi.org/10.1007/978-1-4419-8080-9_7.

Sloss, Andrew N., Dominic Symes ja Chris Wright. 2004. “CHAPTER 10 - FIRMWARE”. Teoksessa *ARM System Developer’s Guide,* toimittanut ANDREW N. SLOSS, DOMINIC SYMES ja CHRIS WRIGHT, 366–379. The Morgan Kaufmann Series in Computer Architecture and Design. Burlington: Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-155860874-0/50011-3. https://www.sciencedirect.com/science/article/pii/B9781558608740500113.

Supermicro. 2013. Viitattu 26. huhtikuuta 2021. https://www.supermicro.com/support/faqs/faq.cfm?faq=16699.

Swetland, Brian. 2015. “Introduction”. Viitattu 26. huhtikuuta 2021. https://github.com/littlekernel/lk/wiki/Introduction/85fffd964e6befd61368bcc38bdc6ac64f474324.

Szefer, Jakub, ja Ruby B. Lee. 2012. “Architectural Support for Hypervisor-Secure Virtualization”. Teoksessa *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems,* 437–450. ASPLOS XVII. London, England, UK: Association for Computing Machinery. ISBN: 9781450307598. https://doi.org/10.1145/2150976.2151022. https://doi.org/10.1145/2150976.2151022.

Tian, Dave Jing, Grant Hernandez, Joseph I Choi, Vanessa Frost, Christie Raules, Patrick Traynor, Hayawardh Vijayakumar, Lee Harrison, Amir Rahmati, Michael Grace ym. 2018. “Attention spanned: Comprehensive vulnerability analysis of {AT} commands within the android ecosystem”. Teoksessa *27th {USENIX} Security Symposium ({USENIX} Security 18),* 273–290.

Tianocore. 2018. “EDK II UEFI Driver Writer’s Guide”. Viitattu 26. huhtikuuta 2021. https://tianocore-docs.github.io/edk2-ModuleWriteGuide/draft/#edk-ii-module-writers-guide.

———. 2020a. “EDK II Module Writer’s Guide”. Viitattu 25. huhtikuuta 2021. https://tianocore-docs.github.io/edk2-ModuleWriteGuide/draft/#edk-ii-module-writers-guide.

———. 2020b. “What is TianoCore?” Viitattu 25. huhtikuuta 2021. https://www.tianocore.org/.

UEFIForum. 2013. “An Introduction to Platform Security”. Viitattu 8. maaliskuuta 2021. https://uefi.org/sites/default/files/press_releases/UEFI_Specifications_Expand_in_Mobile_Devices_and_Non-PC_Markets_May_8_2013.pdf.

———. 2018. “An Introduction to Platform Security”. Viitattu 1. maaliskuuta 2021. %7Bh ttps://uefi.org/sites/default/files/resources/Intel_An%20Introduction%20to%20Platform%20.pdf%7D.

———. 2019. “UEFI Secure Boot In Modern Computer Security Solutions”. Viitattu 25. huhtikuuta 2021. https://uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2019.pdf.

———. 2020a. “UEFI Platform Initialization (PI) Specification”. Viitattu 22. helmikuuta 2021. https://uefi.org/sites/default/files/resources/PI_Spec_1_7_A_final_May1.pdf.

———. 2020b. “UEFI Specification Version 2.8 (Errata B)”. Viitattu 22. helmikuuta 2021. https://uefi.org/sites/default/files/resources/UEFI%20Spec%202.8B%20May%202020.pdf.

———. 2021. “Membership List”. Viitattu 9. maaliskuuta 2021. https://uefi.org/members.

Welton, Ryan, ja Marco Grassi. 2015. “Secured-core PCs: A brief showcase of chip-to-cloud security against kernel attacks”. Viitattu 28. maaliskuuta 2021. https://gsec.hitb.org/materials/sg2015/D2%5C%20-%5C%20Ryan%5C%20Welton%5C%20and%5C%20Marco%5C%20Grassi%5C%20-%5C%20Current%5C%20State%5C%20of%5C%20Android%5C%20Privilege%5C%20Escalation.pdf.

Wu, Songyang, Xiong Xiong, Yong Zhang, Yang Tang ja Bo Jin. 2017. “A general forensics acquisition for Android smartphones with qualcomm processor”. Teoksessa *2017 IEEE 17th International Conference on Communication Technology (ICCT),* 1984–1988. https://doi.org/10.1109/ICCT.2017.8359976.

Xiao, Zihang, Qing Dong, Hao Zhang ja Xuxian Jiang. 2014. “Oldboot: the first bootkit on Android”. https://blogs.360.cn/post/oldboot-the-first-bootkit-on-android.html.

Yao, Jiewen, ja Vincent Zimmer. 2015. “A Tour Beyond BIOS Launching Standalone SMM Driversin the PEI Phase using the EFI Developer Kit II”. Viitattu 25. huhtikuuta 2021. https://raw.githubusercontent.com/vincentjzimmer/Documents/master/A_Tour_Beyond_BIOS_Launching_Standalone_SMM_Drivers_in_PEI_using_the_EFI_Developer_Kit_II.pdf.

Zhang, Zhuowei. 2018. “Comparing Qualcomm’s XBL UEFI bootloaders on Snapdragon 820, 835, and 845”. Viitattu 8. huhtikuuta 2021. https://worthdoingbadly.com/qcomxbl.

Zimmer, Vincent, Michael Rothman ja Suresh Marisetty. 2017. *Beyond BIOS: developing with the unified extensible firmware interface.* 24. Walter de Gruyter GmbH & Co KG.